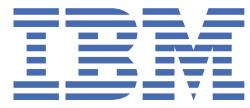


IBM Sterling Transformation Extender



Tables of Contents

Introduction

Introduction	1
The business role of IBM Sterling Transformation Extender	1
Accessibility features for IBM Sterling Transformation Extender	1
Globalization	2
To change application locale (Windows)	2
To change the application locale (Windows) for the Design Studio application	2
To change the application locale (Windows) for tools and runtime applications	3
To change the application locale (UNIX)	3
To change the Launcher Administration locale	3
To change the Management Console locale	4
To change the Resource Registry locale	4
Locale values	5
Product overview	5
Product architecture	5
Runtime services	6
Design environment	6
Projects	6
Schema design	6
Map design	7
Input and output cards	8
Transformation, routing, and business logic rules	8
Adapter configuration for input and output	8
Resource adapters	8
Connection and action design	9
Configuration variables	9
Database Interface Designer	9
Integration Flow Designer	9
Developing application interfaces	10
Modeling the data of business objects	10
Defining transformation rules	11
Defining data sources and targets	11
Defining process models	11
Formulating business process models	12
Defining resource connectivity	12
Defining event rules	12
Configuring map settings	12
Managing systems	12
Type Tree Maker	12
Server components	12
Management and monitoring tools	13
Launcher Administration	13
Simple Network Management Protocol (SNMP) support	13
Management Console	13
Launcher Monitor	13
Snapshot Viewer	14
Software Development Kit (SDK)	14
Software Development Kit Application Programming Interfaces (APIs)	14
Software Development Kit options	14
Adapter toolkit	14
B2B management (Trading Manager)	14
Partner Manager	15
Message Manager	15
Other IBM Sterling Transformation Extender solutions	15
Integration maps	15
Map behavior during execution	16
Input and output card settings	16
Map settings	16
Configuration variables	17
Reserved words and symbols	17
Quotation marks	18
Hex, decimal, and symbol values	18
File name extensions	20
Execution	20
Executing in test and production environments	21
Executing on multiple platforms	21
Command-driven execution model	21
Execution command overriding a database target example	21
Event-driven execution model	22
Launcher	22

Triggers and watches: asynchronous communication	22
Synchronous coordination	22
Launcher Architecture	22
Launcher methodology	23
Business Advantages of the Launcher	23
Audit and transaction logging	23
Application-embedded execution model	23
Examples	24
Tivoli License Manager	24
What's new	
What's new in V11.0.1.0	24
What's new in Packs V10.2.3.0	26
What's new in Packs V10.2.0.1	28
Configuration	
Configuration	28
Data Directory	29
The IBM Sterling Transformation Extender configuration file config.yaml	29
Configuration file syntax validation	29
Configuration file expansion expressions	29
Environment variable expansion	29
Master Key File (MKF) expansion	29
Customizing the Trace	30
Customizing the Trace for IBM Sterling Transformation Extender Rest Execution	30
How to verify the Config.yaml file	31
Enhancing Security, preventing Server Side Forgery Requests attacks	31
Design Server	
Installation	32
Windows installer	32
IBM Sterling Transformation Extender Server Components	32
Installation prerequisites	33
Verify that the following ports are available on your machine	33
Data folders used by IBM Sterling Transformation Extender	33
Additional data folders	34
Installation procedures	34
Installing IBM Sterling Transformation Extender Server with Install Shield	34
Manually Installing IBM Sterling Transformation Extender Server and Runtime	34
Post installation action	34
Modify an existing installation	35
Starting and Stopping IBM Sterling Transformation Extender application	35
Accessing IBM Sterling Transformation Extender	35
Customizing the installation	35
Customizing the Redis and IBM Sterling Transformation Extender Databases	35
Customizing IBM Sterling Transformation Extender URL and Selecting HTTPS protocol	35
Uninstalling IBM Sterling Transformation Extender	36
Troubleshooting the installation	36
Manually Uninstalling the IBM Sterling Transformation Extender	36
Troubleshooting IBM Sterling Transformation Extender installation process	37
Manually updating Apache Tomcat 9 for the IBM Sterling Transformation Extender application	37
Linux installation	37
IBM Sterling Transformation Extender Utility	38
CONFIGURE command	38
INSTALL, UNINSTALL, START, STOP commands	38
STATUS command	38
LOGS command	39
Native installation	39
Prerequisites	39
System Configuration	39
Temporary directory mounting options	40
Changing the mount option for /tmp	40
Changing the IBM Sterling Transformation Extender temporary directory	40
Installation procedures	40
Manually updating Apache Tomcat 9 for the IBM Sterling Transformation Extender application	41
Docker installation	41
Docker configuration	41
Installation	42
Podman installation	42
Troubleshooting	42
Using the OneDB wire listener as a MongoDB server	43
IBM Sterling Transformation Extender design data	43
Design data storage	43
Design data backup and migration	43
MongoDB database operations	43

Design databases	44
Additional accounts	44
Backing up and restoring databases	44
MongoDB migration example	44
Filesystem data backup and restore	45
Filesystem data migration example	45
Administration	45
User Management	45
Managing user accounts	46
Add a user	46
Introduction	46
Installation	46
User Administration	46
Default User Administration	46
Creating a Realm	47
Configuring Clients	47
Connecting to Design Server	47
LDAP User Authentication	48
Selecting LDAP provider in Keycloak	48
Required settings for a successful connection to your LDAP/AD provider	48
User synchronization	49
Mappers	49
Runtime REST API	49
Connecting Runtime REST API server	49
Outbound connections to Runtime Servers	49
Design Server concepts	50
Projects	50
Importing and exporting	50
Connections and actions	50
Project-level connections and actions	51
Map-level connections and actions	51
Connection testing	51
Files	51
Schemas	51
Schema design	51
About schemas	51
Subtypes	52
Schema hierarchy	52
Defining data	52
Objects, types, and classes	52
Testing part of a schema	53
Invalid input message from test data generation	53
Importers	53
COBOL Copybook Importer	53
Running the COBOL copybook importer in Design Server UI	54
Runtime options	54
COBOL copybook Specification	54
Type tree Mapping	54
Clause	55
CSV Importer	55
Running the CSV Importer in Design Server UI	55
CSV importer Properties	55
JSON Importer	56
Importing JSON Schema in Design Server UI	56
Importing JSON Template in Design Server UI	57
The representation of JSON types in schemas	57
Using native JSON schemas in a map	57
JSON cards and text encodings	57
Type properties	57
Defining type properties	58
Basic type properties	58
Name	58
Class	59
Description	59
Intent	59
Partitioned	59
Order subtypes	59
Initiator	59
Terminator	60
Release characters	60
Building release characters for output data	60
Guidelines for using release characters	60
Release character example	60

Empty	61
Empty type property example	61
Document Type	61
Where used	61
Symmetric swapping	62
Orientation	62
Shaping	62
Item properties	63
Item subclass	63
Number item subclass properties	63
Interpret as binary	64
Binary integer presentation	64
Binary float presentation	64
Binary packed presentation	64
Binary BCD presentation	64
Length (bytes)	64
Byte order	64
Interpret as character	65
Character integer presentation	65
Character decimal presentation	65
Zoned character presentation	65
Places > implied	65
Size (digits)	65
Excluded from min and max size	66
Size example	66
Separators	66
Integer separators	66
Separator > format	66
1000's syntax > value	67
1000's syntax (literal) > value	67
1000's syntax (variable) > default	67
1000's syntax (variable) > item	67
1000's syntax (variable) > find	67
Decimal separators	67
Separators > format	68
Separators > 1000's syntax	68
Separators > fraction syntax	68
Sign	68
Pad	69
Pad > value	69
Pad > Padded to	69
Padded to > length	70
Padded to > SizedAs	70
Padded to > CountsTowardMinContent	70
Padded to > Allow Excess Trailing Pads	71
Pad > justify	71
Pad > apply pad	71
Pad > fill	72
Restrictions	72
Restrictions > ignore case	72
Restrictions > rule	72
National language	72
National language > data language	72
Supported code pages	72
NONE and Zero	76
NONE > Special Value and Zero > Special Value	76
NONE > Required on input and Zero > Required on input	76
Places	76
E-Notation	76
Data parsing expectations	76
Generating a data item defined with E-Notation	77
E-Notation output	77
Text item subclass properties	77
Interpret as binary	64
Interpret as character	65
Size (content)	78
Pad	69
Restrictions	72
National language	72
NONE	79
NONE > Special Value	79
NONE > Required on input	79

Bidirectional	80
Date & Time item subclass properties	80
Date	81
Date > format	81
Time	81
Time > format	81
Format	81
Use of format elements	82
Custom date format	82
Custom Time Format	82
Time zones	83
Time zone format string for XML	83
Optional time segments of the time format string	84
Date and time format examples	84
NONE and Zero	84
NONE > Special Value and Zero > Special Value	84
NONE > Required on input and Zero > Required on input	85
Syntax item subclass properties	85
Syntax objects with variable values	85
Example of variable syntax object as an item type	85
Syntax objects as data	85
Example of syntax objects as components of a type	86
Delimiter > find	86
Group properties	86
Group subclass	86
Properties of group subclasses	87
Choice group components	87
Unordered group components	87
Sequence group formats	87
Explicit format	87
Track	88
Fixed syntax	88
Guidelines for defining a fixed group	88
Explicit delimited syntax	88
Delimiter	88
Implicit format	89
Floating component	89
Implicit whitespace syntax	89
Build as	89
Character set	89
Implicit delimited syntax	89
Delimiter	90
No syntax	90
Distinguishable components of an implicit group	90
Specifying a delimiter	90
Literal	90
National language	72
Data language	90
Variable	91
Location	91
Delimiter value appears as data	91
Allow Excess Trailing Delimiters	91
Defining exclude characters	91
Escaping an excluded character in the data content	92
XML properties in the schema	92
Support for XML constructs	92
XML schema datatypes	93
Simple types	93
Complex types	93
Elements	93
Attributes	94
Groups and substitution groups	94
Identity constraints	94
Namespaces	94
Element type declarations	94
Element and attribute wildcards	94
Type propagation for Category, Group, and Item Properties	94
To Propagate in Category/Group/Item property	95
To Propagate Restrictions in Category/Group/Item property	95
Item restrictions	95
Restrictions settings	95
Value restrictions	95

Character descriptions	96
To add character restrictions to an item	96
Range restrictions	96
Value not in range	96
Value not in range	96
To add range restrictions to an item	97
Inserting symbols	97
Ignoring restrictions	97
Components	97
Components are required for group types	97
Components must be in the same schema	97
Importance of component order	98
Component range	98
Indefinite number	98
Single occurrence	98
Defining components	98
Guidelines for defining components	99
Complete type name	99
Relative type names	99
Moving types with the same relative type name	99
Ambiguous type names	99
Specifying minimum and maximum consecutive occurrences in the component list	99
Fixed and variable ranges	100
Specifying a component range	100
Variable component names	100
Required and optional data	100
Significance of required data	100
Defining component rules	101
Examples of component rules	101
Component rule syntax	101
Entering object names in component rules	101
Shorthand notation	102
Component rules are context-sensitive	102
Special characters in component rules	102
Comments in component rules	102
Component attributes	102
Identifier attribute	102
Restart attribute	102
Sized attribute	103
Include self in size	103
Partitioning	103
Determining when to partition	103
Required partitioning	103
Partitioning for convenience	104
Benefits of partitioning	104
Partitioning types	104
Partitioning items	104
Partitioning an item type using initiators	104
Partitioning an item type using restrictions	104
Example of using restrictions	105
Partitioning an item type by format	105
Partitioning groups	105
Partitioning a group type using initiators	105
Partitioning a group type using identifiers	105
Partitioning a group type using component rules	106
Type inheritance	106
Inheritance of item properties and restrictions	106
Inheritance of category properties and components	106
Organizing types under a category	106
Using categories for inheritance	106
When not to use categories	107
Propagation of type properties	107
Propagation of type properties common to all types	107
Propagation of type properties for specific types	107
Error detection	108
How error detection works	108
Existence indicators	108
Existence versus presence of components	109
Error recovery	109
Using the Restart attribute	110
How the Restart attribute works	110
Mapping invalid data	110

Schema analyzer	110
Mapping effects	110
Internal consistency	111
Logical analysis	111
Structural analysis	111
Error and warning messages	111
Distinguishable objects	111
Objects in a data stream	111
Schema analyzer and distinguishable objects	112
Bound types	112
Bound components	112
Component of a fixed group	112
Component of an explicit delimited group	112
Component of an implicit group	113
Component of a choice group	113
Component of an unordered group	113
Group starting set	113
Group unbound set	113
Unbound set of a sequence group	113
Initiator-distinguishable types	114
Determining if a component is initiator-distinguishable from its following set	114
Determining if a partition is initiator-distinguishable from its following set	114
Determining if two types are initiator-distinguishable	114
Distinguishable objects of the same component	115
Content-distinguishable components	115
Content-distinguishable types	116
Ending-distinguishable types	118
Distinguishable data objects of an implicit group	119
Guidelines for defining an implicit delimited sequence	119
Guidelines for defining an implicit sequence that has no delimiter	119
Guidelines for defining an implicit unordered group that is delimited	119
Guidelines for defining an implicit unordered group that has no delimiter	120
Distinguishable data objects of an explicit group	120
Guidelines for defining an explicit fixed group	120
Guidelines for defining an explicit delimited group	120
Objects of a choice group	120
Objects of a partitioned type	120
Distinguishable syntax objects	120
Return codes and error messages	121
Type Tree Analyzer errors and warnings	121
Schema analysis logic error messages	121
Logic error and warning messages	127
Schema analysis structure error messages	127
Schema analysis structure warning messages	127
Maps	129
Map design	129
Introduction to the Map Designer	129
Using the Map Designer	130
Define data objects and properties	130
Create maps to specify sources and targets	130
Auto Map preferences	130
Mapping Scope	130
Rule Creation Option	131
Name Matching Options	131
Mapping Criteria	131
Match percent examples	131
Profiler preferences	132
Where Used view	132
Search options	133
Search and replace	133
Search For	133
Scope	134
Using search and replace	134
Find and replace	134
Using find and replace	135
Maps and map source files	135
Map name guidelines	135
Map source file differences	136
Troubleshooting tip	136
Input and output cards	136
Card overview	136
Card specifics	137

Compositional hierarchy	137
Native XML schema support	137
Input-card type requirements for XML documents	137
Parsing native XML schemas	138
Running the Native XML Schema Customization wizard task flow	138
Support for XSDL hints in output cards	138
Native JSON schema support	138
Static input cards	139
Static file validation and tracing	139
Automatically mapping source to target	140
Auto Map overview	140
Running the Auto Map task flow	140
Setting the Auto Map options	141
Selecting the rules to create	141
Editing rule proposals	141
Renaming the map	142
Card settings	142
Schema	142
CardName	142
CardName guidelines	142
Schema	143
NativeXMLSchemaCustomization	143
Identifier	143
GlobalElements	143
XSITypes	143
MixedContent	144
Type	144
Metadata	144
NameSpaces	144
SourceRule input card settings	144
SourceRule Setting overrides	145
Input data retrieval	145
FetchAs	146
FetchAs > WorkArea	146
Guidelines for reusing work areas	146
FetchAs > FetchUnit	146
FetchUnit details	147
FetchUnit with multiple input cards	147
GET	147
Source	147
Source > FilePath	147
Source > Command	147
DatabaseQueryFile	148
DatabaseQueryFile > Database	148
DatabaseQueryFile > Query	148
DatabaseQueryFile > File	148
Transaction	148
OnSuccess	148
OnFailure	148
Scope	149
FetchAs and Scope actions	149
TargetRule output card settings	149
TargetRule Setting overrides	150
PUT	150
FilePath	150
Target > Command	150
DatabaseQueryFile	150
DatabaseQueryFile > File	150
DatabaseQueryFile > Database	150
DatabaseQueryFile > Table	150
Transaction	151
OnSuccess	151
OnFailure	151
Scope	151
Input and output card settings	151
Retry	152
Retry > Switch	152
Retry > MaxAttempts	152
Retry > Interval	152
DocumentVerification	152
Use of DocumentVerification	152
Use of DocumentVerification for output data	153

Backup settings	153
Backup > Switch	153
Backup > When	153
Backup > BackupLocation	154
BackupLocation > Directory	154
Directory > Value	154
BackupLocation > Filename	154
BackupLocation > FileName > Action	154
BackupLocation > FileName > Value	154
Effects of modifying a schema	155
Changing a type that is used for a card	155
SyntaxCard	155
Editing a card	155
Changing a card name	155
Changing the type of an input card	156
Changing the type of an output card	156
Changing the schema of a card	156
Map events	156
Map rules	156
Map rules overview	157
Rule editor	157
Color coding of map rules	157
Configuring colors for map rules	157
Map rules as expressions	157
Map rules define how to generate data objects	158
Map rules are evaluated independently in sequential order	159
Generating no output	159
Text literals as NONE	159
Special characters in map rules	159
Syntax errors in map rules	159
Data object names in map rules	159
Card names in map rules	160
Component names in map rules	160
Index in map rules	160
Partition in map rules	160
Comments in map rules	160
Unavailable rule cells	161
Use ellipses	161
Search functionality	161
To search in a Map using Search icon	161
Formulating map rules	161
When input data is missing	161
When portions of an input series are missing	162
When an output evaluates to NONE	162
Mapping an input item to an output item	162
Automatic item conversions	163
Intermediate item conversions	163
Concatenating text strings	163
Mapping a group to a group	163
Automatic conversion of syntax items	163
Mapping to a group with a maximum range of 1	163
Outputs with maximum range greater than 1	164
Indexing an output	164
Mapping to multiple occurrences of a group	164
Indexing an Input	164
Mapping from a floating component type	164
Functional map basics	164
When to use a functional map	165
Similarities of input and output data	165
Using a functional map	165
Example scenario for using a functional map	166
Comparing executable and functional maps	166
Syntax of a functional map expression	166
Determining the arguments of a functional map	167
Input argument to a functional map evaluates to NONE	167
Entering the map rule that references the functional map	167
Input card types of a functional map	168
Input arguments as data objects	168
Output card type of a functional map	168
Entering map rules	169
Using multiple inputs in a functional map	169
Input objects are triggers	169

Rename a functional map	169
Map settings	169
MapAudit settings	170
MapAudit > Switch	170
MapAudit > BurstAudit	170
BurstAudit > Data	170
Data > SizeValidation	171
BurstAudit > Execution	171
MapAudit > SummaryAudit	171
SummaryAudit > Execution	171
MapAudit > SettingsAudit	172
SettingsAudit > Data	172
SettingsAudit > Map	172
MapAudit > AuditLocation	172
AuditLocation > Directory	172
Directory > Value	173
AuditLocation > FileName	173
FileName > Action	173
FileName > Value	173
AuditLocation > Sized	173
MapTrace settings	174
MapTrace > Switch	174
MapTrace > TraceLocation	174
TraceLocation > Directory	174
Directory > Value	174
TraceLocation Filename	174
MapTrace > InputContentTrace	175
InputContentTrace > CardNumber	175
InputContentTrace > StartObject	175
InputContentTrace > EndObject	175
Using ranges for InputContentTrace	175
MapTrace > RulesTrace	175
RulesTrace > CardNumber	175
MapTrace > SummaryContentTrace	176
MapTrace Format	176
WorkSpace settings	176
WorkSpace > PageSize	176
WorkSpace > PageCount	177
WorkSpace > Directory	177
Directory > Value	177
WorkSpace > WorkFilePrefix	177
WorkFilePrefix > Action	177
Century settings	177
Century	177
Century > CCLookup	178
Validation settings	178
Validation	178
Validation > OnValidationError	178
Validation > RestrictionError	178
Validation > SizeError	179
Validation > PresentationError	179
Validation Ignore Component Rules	179
Retry settings	179
Retry > Switch	179
Retry > MaxAttempts	180
Retry > Interval	180
Warnings settings	180
Other Warnings settings	180
Warnings > Return	180
CodePageFallback	181
MapRuntime setting	181
Configuring bursts	181
Batching logical messages	181
Using bursts for large files	182
Configuring bursts for error recovery	182
Configuring the RUN function and bursts	182
Logical or adapter-specified FetchUnits	182
Building maps	183
Referenced maps	183
Multi-platform maps	183
Including a static input file in a compiled map	183
Building a map	183

Rebuilding a map that has changed	184
Building maps for a single platform	184
Platform-specific file name extensions	184
Sample JCL	185
Schema errors	186
Map build analysis errors	186
Map build warning messages	186
Map build compile errors	186
Compiled map files	186
Running a map	186
Running in a multi-threaded environment	187
Thread-safe map	187
Verifying that the map meets basic thread-safe checks	187
Map settings to optimize concurrent performance	187
Card settings to optimize concurrent performance	187
Run results	188
Map Profiler	188
Configuring the Map Profiler in the Design Server UI	188
Map Profile Settings	188
Map performance	189
Map Profiler overview	189
Map Profiler output	189
Type names	190
Function times	190
RUN maps	190
Configuring the Map Profiler	190
Profiler preferences	132
Command line	191
Using the Map Profiler	191
Profiler output example	191
Profiler summary report example	192
For best results...	193
Managing invalid data	193
Using the Restart Attribute	193
Using the REJECT function	193
Using the ISERROR function	193
Using the CONTAINSERRORS function	194
Map audit overview	194
Audit log contents	194
Controlling audit log content	194
Execution summary in the audit log	195
Elapsed execution time in the audit log	195
Source data report by card	195
Validation Errors in the Source Report Section of the Audit Log	195
Target data report by card	196
Work area information by card	196
Execution summary audit log sample	196
BurstAudit execution in the audit log	197
BurstAudit Data in the audit log	197
Data content information in the audit log	197
Data settings in the audit log	197
Data settings in the audit log sample	198
Map settings in the audit log	198
Mapping from the audit log	198
ExecutionSummary in the Tryaudit schema	199
Configuring data audit	199
Detail	199
Item data	200
Status codes in the audit log	200
Troubleshooting	200
Overview of debugging aids	200
Command Server window	200
Run results	201
Audit log	201
Customized validation settings	201
Map Debugger	201
Breakpoints	201
Saving breakpoints	202
Conditional BreakPoints	202
Rule execution	202
Viewing object values at a rule level	202
Running the Map Debugger	202

Remote debugging	203
Debugging a map remotely	203
Trace files	203
Trace for general data	204
Creating a trace file	204
Trace file contents	204
When data is valid	205
Group validation	205
Valid card objects	205
Invalid data	205
Invalid card objects	205
Finding the important messages	205
Existence of data	206
Validation for invalid objects	206
Unknown data	206
Negative messages not fatal	206
Example of optional occurrences	207
Static file validation and tracing	139
Trace for XML data	208
Creating an XML trace file	208
XML validation log	208
Validation sequence	208
Validation results	208
Trace for XML parser errors only	209
Error and warning messages	209
Map execution error and warning messages	209
Map execution warning messages	213
Map build error messages	213
Map build warning messages	215
Map compile error messages	216
Audit log status codes	217
Valid audit log status codes	217
Warning audit log status codes	218
Error audit log status codes	218
Flows	218
Introduction to Flows	218
Nodes	219
Map	219
Flow	220
Source and Target	220
Introduction to nodes	220
Source Node	220
Target Node	220
Node settings	220
Request node	221
Cache Read and Write	221
Cache Read and Cache Write nodes configuration	222
Cache Read	222
Settings	223
Cache Scope	223
Data Format	223
JSON data format	223
Delimited data format	224
Key Delimiter	224
Record Delimiter	224
Include Key	224
Key Prefix	225
Delete Key	225
Default Key Value	225
Key Pattern	225
Node Logging	225
Cache Write	225
Settings	226
Cache Scope	226
Data Format	226
Key Delimiter	224
Record Delimiter	227
Key Prefix	227
Node Logging	227
Clone	227
Decision	227
Settings	227

Flow Variable	227
Operator	227
Case Sensitive	228
Value	228
Fail	228
Settings	228
Error Message	228
Format Converter Node	228
Converting from XML to CSV or Delimited	229
Using the Format Converter Node	229
CSV input format - available properties	230
CSV input format Delimited - available properties	230
CSV output format - available property	230
CSV output format delimited - available properties	230
JSON or YAML input, CSV output - available properties	230
JSON or YAML output, CSV input or delimited- available properties	230
XML input, CSV output or delimited- available properties	231
JSON Read Node	231
Settings	231
Paths	232
Variable Scope	232
Output Content	232
Examples	232
JSON Transform Node	233
JSON Template Property	233
Path to a Primitive	233
Path to an Object	234
Path to an Array	234
Examples	234
JAVA	235
Settings	235
Class name	235
Properties	236
Copying Java nodes zip file under the non-docker environment	236
Copying Java nodes zip file under the docker environment	236
Join	236
Settings	237
Write To File	237
File Path	237
Record Delimiter	237
Output Header	237
Header Flow Variable	237
Header	238
Header Delimiter	238
Log	238
Log File Name	238
Passthrough	238
REST Client	238
Functionality	239
Terminals	239
Settings	239
Authentication	239
Input Data Request Mode	239
Request Assignments	239
Response Assignments	240
Logging	240
Proxy server support	240
Proxy server support in REST Client Node	240
Proxy configuration using environment variables	240
Proxy configuration using flow variables	241
Proxy configuration using properties	241
Proxy configuration using the command line	241
Troubleshooting of proxy server	241
Route	241
Settings	242
Sleep	242
SOAP Client Node	242
Functionality	243
Terminals	243
Settings	243
Authentication	243
Logging	243

Split	244
Configuring a Split node	244
Consuming data through an input terminal	244
Consuming data from a file	244
Defining number of Split instances	244
Settings	244
Split example	245
Status Node	245
Functionality	246
Terminals	246
Terminals	246
Error and Success terminals	246
Error and Success terminals example	246
Log terminals	247
Flow terminals	247
Scheduled flows	247
Listener	247
Listener example	247
Amazon SQS Listener	247
Listener Properties and Commands	248
To continuously listen to a message from a queue	248
Flow audits	249
Flow variables	249
Flow variables example	249
Deployment	250
Deployment overview	250
Server groups	250
Server definition	250
Packages	251
Package definitions	251
Deployment	251
Design Server UI	251
Design workspace	252
Schema Designer	252
Schema Designer dictionary	252
Schema Designer structure	252
Schema Designer properties	252
Schema Designer toolbar	252
Map Designer	252
Map structure	252
Map Designer workspace	252
Build and run	252
Map Design navigator	252
Map Designer toolbar	252
Flow Designer UI	252
Palette	253
Canvas	253
Nodes	253
Context menu	253
Links	253
Structure pane	254
Information pane	254
Toolbar	254
Design Server Properties	254
Where Used	254
Dashboard	254
Server Dashboard	255
Functionality of the Server Dashboard	255
Using Design Server	255
Add a user	255
Create a project	255
Import a project	256
Export a project	256
Selective Export	256
Create a flow	256
Define mapping between source and target	257
Running flows in the Designer	257
Preparing to run a flow from the Designer	257
Server section options	258
Variables section options	258
Input Data section options	258
Running the flow	258
Viewing logs	258

Viewing flow terminal data	258
Displaying link data	259
Including Maps and Files that are Dependencies of a Flow	259
Design Studio	
Introduction to Design Studio	259
Design Studio overview	259
Design Studio configuration	259
Design Studio design tasks	260
Design Studio runtime tasks	260
Execution settings	260
Design Studio customization	260
Customizing Design Studio toolbars	260
Customizing Design Studio shortcut keys	261
Map Designer	261
Map Designer overview	262
Map Designer Eclipse overview	262
Map deployment overview	262
Database Interface Designer	262
Database Interface Designer unique key combinations	262
Database Interface Designer configuration	263
Database Interface Designer adapters	263
Integration Flow Designer	263
Integration Flow Designer unique key combinations	263
Integration Flow Designer components	264
Design Studio reference	264
Design Studio quotation mark usage	264
Design Studio reserved words and symbols	264
Design Studio hex, decimal, and symbol values	265
Type Designer	268
Getting started	268
Starting the Type Designer	268
Opening an existing type tree	268
Introduction to the Type Designer	269
About type trees	269
Subtypes	269
Type tree hierarchy	269
Type Designer basics	270
Type Designer files	270
Type Designer icons	270
Opening or changing to the Transformation Extender Development perspective	271
Customizing the Transformation Extender Development perspective	271
Customizing the Type Designer environment	271
Confirmation message preferences	272
Group view preferences	272
Font preferences	273
Search options	273
Search and replace	273
Search For	274
Limit To	274
Scope	275
Using search and replace	275
Find and replace	276
Using the find and replace operation	276
Bookmarks and Tasks	276
Adding a bookmark	277
Removing a bookmark	277
Adding a task	277
Removing a task	277
Working with type trees	278
Creating type trees	278
To create a type tree	278
To add types to the type tree	278
Viewing type trees side by side	279
Viewing subtypes	279
Selecting subtypes	279
To change the placement order of subtypes	279
Defining data	280
Objects, types, and classes	280
To define a literal or variable value	280
To add a component to the target tree	280
To view a specific component/item restriction	281
Type tree differences	281

Exporting a type tree	281
Managing types	282
Selecting subtypes	279
Drag-and-drop operations in the Type Designer	282
Moving and copying objects	283
Moving a type	283
Copying a type	283
Type names	283
Reordering objects	284
Reordering existing subtypes	284
Merging types	284
Merging a type	284
Supertypes	285
Existing types	285
Invalid types	285
Renaming types	285
Generating a type definition report	285
Testing part of a type tree	286
Generating test data	286
Invalid input message from test data generation	286
Type properties	286
Defining type properties	286
Basic type properties	287
Name	288
Class	288
Description	288
Intent	288
Partitioned	288
Order subtypes	289
Initiator	289
Terminator	289
Release characters	289
Building release characters for output data	289
Guidelines for using release characters	290
Release character example	290
Empty	290
Empty type property example	290
Document Type	291
Where used	291
Symmetric swapping	291
Orientation	291
Shaping	292
Item properties	292
Item subclass	292
Number item subclass properties	293
Interpret as binary	293
Binary integer presentation	293
Binary float presentation	293
Binary packed presentation	293
Binary BCD presentation	294
Length (bytes)	294
Byte order	294
Interpret as character	294
Character integer presentation	294
Character decimal presentation	294
Zoned character presentation	294
Places > implied	295
Size (digits)	295
Excluded from min and max size	295
Size example	295
Separators	295
Integer separators	296
Separator > format	296
1000's syntax > value	296
1000's syntax (literal) > value	296
1000's syntax (variable) > default	296
1000's syntax (variable) > item	296
1000's syntax (variable) > find	297
Decimal separators	297
Separators > format	297
Separators > 1000's syntax	297
Separators > fraction syntax	297

Sign	298
Pad	298
To specify a pad character	299
Pad > value	299
Pad > Padded to	299
Padded to > length	299
Padded to > SizedAs	299
Padded to > CountsTowardMinContent	300
Padded to > Allow Excess Trailing Pads	301
Pad > justify	301
Pad > apply pad	301
Pad > fill	301
Restrictions	301
Restrictions > ignore case	301
Restrictions > rule	302
National language	302
National language > data language	302
Supported code pages	302
NONE and Zero	305
NONE > Special Value and Zero > Special Value	305
NONE > Required on input and Zero > Required on input	305
Places	306
Text item subclass properties	306
Interpret as binary	293
Interpret as character	294
Size (content)	307
Pad	298
Restrictions	301
National language	302
NONE	308
NONE > Special Value	308
NONE > Required on input	308
Bidirectional	308
Date & Time item subclass properties	309
Date	309
Date > format	309
Time	309
Time > format	309
Format	310
Use of format elements	310
Custom date format	310
Custom Time Format	311
Time zones	311
Time zone format string for XML	312
Optional time segments of the time format string	312
Date and time format examples	312
NONE and Zero	313
NONE > Special Value and Zero > Special Value	313
NONE > Required on input and Zero > Required on input	313
Syntax item subclass properties	313
Syntax objects with variable values	313
Example of variable syntax object as an item type	314
Syntax objects as data	314
Example of syntax objects as components of a type	314
Delimiter > find	314
Group properties	315
Group subclass	315
Properties of group subclasses	315
Choice group components	315
Unordered group components	315
Sequence group formats	316
Explicit format	316
Track	316
Fixed syntax	316
Guidelines for defining a fixed group	316
Explicit delimited syntax	317
Delimiter	317
Implicit format	317
Floating component	317
Implicit whitespace syntax	318
Build as	318
Character set	318

Implicit delimited syntax	318
Delimiter	318
No syntax	318
Distinguishable components of an implicit group	318
Specifying a delimiter	319
Literal	319
National language	302
Data language	319
Variable	319
Location	319
Delimiter value appears as data	320
Allow Excess Trailing Delimiters	320
Defining exclude characters	320
Adding literal	321
Adding syntax type	321
Escaping an excluded character in the data content	321
XML properties in the type tree	322
Support for XML constructs	322
XML schema datatypes	322
Simple types	323
Complex types	323
Elements	323
Attributes	323
Groups and substitution groups	323
Identity constraints	323
Namespaces	324
Element type declarations	324
Element and attribute wildcards	324
Item restrictions	324
Defining item restrictions	324
Restrictions settings	325
Value restrictions	325
Character restrictions	325
To add character restrictions to an item	326
Range restrictions	326
Value not in range	326
To add range restrictions to an item	327
Inserting symbols	327
Ignoring restrictions	327
Components	327
Components are required for group types	328
Components must be in the same type tree	328
Importance of component order	328
Component range	328
Indefinite number	329
Single occurrence	329
Group views	329
Viewing nested components	329
Defining components	329
Complete type name	330
Relative type names	330
Moving types with the same relative type name	330
Ambiguous type names	330
Manual entry of types with same relative type names	330
Always drag components	331
Viewing the component number	331
Specifying minimum and maximum consecutive occurrences in the component list	331
Fixed and variable ranges	331
Specifying a component range	331
Viewing the range column	332
Types that can be components	332
Guidelines for defining components	332
Variable component names	332
Opening a component view	333
Required and optional data	333
Significance of required data	333
Defining component rules	333
Examples of component rules	334
Component rule syntax	334
Entering object names in component rules	334
Shorthand notation	335
Component rules are context-sensitive	335

Special characters in component rules	335
Inserting functions into component rules	335
Formatting a component rule	335
Comments in component rules	335
Syntax errors	336
Managing components	336
Component attributes	336
Identifier attribute	336
Restart attribute	336
Sized attribute	336
Include self in size	337
Partitioning	337
Determining when to partition	337
Required partitioning	337
Partitioning for convenience	338
Benefits of partitioning	338
Partitioning types	338
Partitioning items	338
Partitioning an item type using initiators	338
Partitioning an item type using restrictions	339
Example of using restrictions	339
Partitioning an item type by format	339
Partitioning groups	339
Partitioning a group type using initiators	339
Partitioning a group type using identifiers	340
Partitioning a group type using component rules	340
Type inheritance	340
Inheritance of item properties and restrictions	340
Inheritance of category properties and components	341
Organizing types under a category	341
Using categories for inheritance	341
When not to use categories	341
Propagation of type properties	341
Propagation of type properties common to all types	341
Propagation of type properties for specific types	342
Propagating type properties	342
Error detection and recovery	343
Error detection	343
How error detection works	343
Existence indicators	343
Existence versus presence of components	344
Error recovery	344
Using the Restart attribute	345
How the Restart attribute works	345
Mapping invalid data	345
Type Tree Analyzer	346
Mapping effects	346
Internal consistency	346
When to Analyze Structure or Logic	346
Logical analysis	346
Structural analysis	346
To analyze a type tree	346
Error and warning messages	347
Distinguishable objects	347
Objects in a data stream	348
Type tree analyzer and distinguishable objects	348
Bound types	348
Bound components	348
Component of a fixed group	349
Component of an explicit delimited group	349
Component of an implicit group	349
Component of a choice group	349
Component of an unordered group	349
Group starting set	349
Group unbound set	350
Unbound set of a sequence group	350
Initiator-distinguishable types	350
Determining if a component is initiator-distinguishable from its following set	350
Determining if a partition is initiator-distinguishable from its following set	350
Determining if two types are initiator-distinguishable	350
Distinguishable objects of the same component	351
Content-distinguishable components	352

Content-distinguishable types	352
Ending-distinguishable types	354
Distinguishable data objects of an implicit group	355
Guidelines for defining an implicit delimited sequence	355
Guidelines for defining an implicit sequence that has no delimiter	356
Guidelines for defining an implicit unordered group that is delimited	356
Guidelines for defining an implicit unordered group that has no delimiter	356
Distinguishable data objects of an explicit group	356
Guidelines for defining an explicit fixed group	356
Guidelines for defining an explicit delimited group	356
Objects of a choice group	357
Objects of a partitioned type	357
Distinguishable syntax objects	357
Utilities for XML	357
XML Type Tree/Schema Synchronization utility	357
XML type tree compatibility utility	358
Modifying the target type tree	358
Any-2-XML	359
Using Any-2-XML	359
Return codes and error messages	359
Type Tree Analyzer errors and warnings	359
Type tree analysis logic error messages	360
Logic error and warning messages	365
Type tree analysis structure error messages	366
Type tree analysis structure warning messages	366
Type Tree Importer	367
Introduction to the Importer Wizard	367
List of importers	368
Application-specific importers	368
Configuring JVM options from the dtx.ini file	368
JVM options	368
JNI layer trace	369
Configuring JVM options example	369
Accessing the Importer Wizard	369
CORBA IDL Importer	370
Java Class Importer	370
Java Message Service (JMS) Importer	370
Text File Importer	370
Type Library Importer	370
Type Tree Maker Importer	370
COBOL Copybook (Deprecated) Importer	370
Running the COBOL Copybook (Deprecated) Importer	371
Wrapped and unwrapped copybook format	371
Wrapped format	371
Unwrapped format	372
Copybook specifications	372
Troubleshooting	372
Modifying a copybook file	372
Modifying a generated type tree	372
REDEFINES clause	372
COBOL Occurs Depending On	373
COBOL Occurs Depending On component rules	373
When component is numeric	373
When component is not numeric	373
COBOL features not supported	374
Bytes assigned for COMP data	374
Bytes assigned for COMP-3 data	374
COBOL Copybook Importer	374
Running the COBOL Copybook Importer	375
Selecting communication data structures	375
COBOL Copybook Importer compiler preferences	376
Troubleshooting	376
COBOL Copybook Importer error message types	376
COBOL Copybook Importer log and trace files	376
COBOL and type tree mapping	377
OpenAPI importer	378
Mapping to a REST API with the OpenAPI importer	378
Hosting a REST API with the OpenAPI importer via the Launcher	379
JavaScript Object Notation (JSON) parser	380
JSON importer	380
Running the JSON Importer	380
PL/I Importer	381
Running the PL/I Importer	381

PL/I Importer preferences	382
Enable IMS Support	383
PL/I Language Support	383
Troubleshooting	384
Error message types	384
PL/I and type tree mapping	384
XML Schema and DTD Importers	386
XML Schema Importer	386
Running the XML Schema Importer	386
About classic validation	387
XML type tree description	388
Misc category	388
Prolog category	388
Attr category	388
Type category	388
Element category	388
Doc group	389
Global	389
XML schema elements and attributes	389
XML schema elements	389
Primitive datatypes	390
Derived datatypes	390
XML schema constraining facets	392
Global elements	393
xsi:type attribute	393
xsi:type support for abstract and concrete data types	393
Use of derived types on base types by using xsi:type attributes	393
When XSDL hints are required in output	394
Example using XSDL hints	394
XML DTD Importer	395
Running the XML DTD Importer	395
DTD elements and attributes in the type tree	396
WSDL Importer	396
Support for SOAP arrays	396
WSDL document elements	396
WSDL document types	397
Running the WSDL Importer	397
Special symbols	397
ANYQUOTE and QUOTE	398
ANYQUOTE and QUOTE examples	398
IGNORE	398
IGNORE examples	398
Character restrictions	399
Using a configuration file for character restrictions	399
XML configuration file and schema	399
XML schema	399
XML configuration file	399
Customizing the configuration file	400
Using character restrictions	400
Understanding character restriction ordering	401
Validation of input data	401
Construction of output data	401
Summary of the ordering process	401
Multiple character restriction sets	401
WSDL example	401
Return codes and error messages	401
COBOL Copybook return codes and messages	401
XML parsing error messages	403
Type Tree Maker	403
Introduction	403
Creating document files	403
Example files	404
Command files	404
Accessing the Importer Wizard	404
Using the Type Tree Maker	404
Creating an XML document file	404
Creating a Type Tree	404
Type Tree Maker Importer	405
Running the Type Tree Maker Importer	405
Starting the Type Tree Maker from a command line	405
Type Tree Maker document file	406
File structure	406

Commands and comments	406
Prolog	406
Processing instructions	407
TTMAKER element	407
XML tag notation	407
Composition of the .mts file	408
Commands	408
AddComponent	408
AddRestriction	408
ASCTYPE_XML	409
BinaryDatetime	409
BinaryNumber	409
BinaryText	410
CATEGORY	410
CharDateTime	410
CharNumber	410
CharRestrictions	411
CharTextJapanese	411
CharTextWestern	411
Choice	412
ChoiceComponent	412
GROUP	412
ITEM	412
NEWTREE	413
PROPAGATE	413
Range	413
Range command	413
Range command element	413
RangeRestrictions	414
Sequence	414
SequenceComponent	414
SyntaxItem	414
Unordered	414
UnorderedComponent	414
Value	415
ValueRestrictions	415
Command elements	415
BCD	416
BinFloat	416
BinInt	416
CharInt	417
Decimal	417
Decimal format Strings	417
Leading-Sign	417
Whole#	418
Fraction	418
Trailing-Sign	418
Sign Sub-string formats	418
Delimited	419
Components of delimited	419
DelimiterLiteral	419
empty	419
ExcludeChar	420
ExcludeRange	420
ExcludeValue	420
Explicit	420
FractionItem	420
Implicit	421
IncludeChar	421
INITIATOR	421
Japanese	421
Literal	421
Modify	421
OneByteLiteral	422
Packed	422
PaddedToFixedSize	422
PadNumber	422
PadText	423
RELEASE	423
ROOT	423
Size	424
TERMINATOR	424

ThousandsItem	424
TotalDigits	424
TTMAKER	425
TypeSyntax	425
Variable	425
Western	425
WhiteSpace	425
Zero	426
Zoned	426
Type Tree Maker examples	426
Export example	426
Installed export example files	427
Files created	427
Generating a cross-reference table from a restriction list	427
Import example	427
Installed import example files	428
Files created	428
MyImport.mtt type tree	428
Mapping from metadata to a TTMAKER document	428
DTD example	429
Return codes and error messages	429
Type Tree generation error messages	429
Type tree generation logic warning messages	430
Errors and warnings	431
Integration Flow Designer	431
Integration Flow Designer overview	431
When to Use the Integration Flow Designer	431
Using the Integration Flow Designer	431
Stages of using the Integration Flow Designer	431
Stage 1: defining systems	432
Stage 2: verifying component relationships	432
Stage 3: preparing systems to run	432
Integration Flow Designer basics	432
Filename extensions	432
Starting the Integration Flow Designer	433
Converting files from previous versions	433
Integration Flow Designer user interface	433
Navigator	433
Navigator icons	434
Navigator: List view	434
Navigator: Composition view	434
Using the Navigator	434
Drag-and-drop functionality	434
Clipboard functionality in the Integration Flow Designer	435
System and subsystem components	435
Focus options	435
Resizing/moving options	435
Displaying components from the navigator	435
Dragging components	435
System windows	435
Toolbox	436
Menu Commands	436
Menu Bar	436
File menu	436
Edit menu	437
View Menu (Alt+V)	437
System menu	437
Tools menu	438
Window Menu (Alt+W)	438
Help menu	438
Customizing your work environment	438
Shortcut keys	438
Options	439
Maps	439
Miscellaneous	439
Navigator	439
Subsystem	439
Confirmations	439
Import and export	439
Finding and replacing in the Integration Flow Designer	439
Finding and replacing component attributes	440
Find and replace operations	440

Using the more find and replace option	440
To find component names	441
To find specific component names	441
To find card names	441
To find path names	441
To find specific path names	441
To replace	442
Text in a system definition diagram	442
Printing a system definition diagram	442
Using the grid	442
To view the grid	442
To cause new components to be aligned to a grid point	443
To align existing components to grid points	443
How the Integration Flow Designer works	443
Theory of operation	443
Map components	443
Command Server or Launcher	443
Command Servers	444
Launchers	444
Map component execution settings	444
Launcher settings	444
Source and target settings	444
Map settings	445
Systems	445
Single-server systems	445
Distributed systems	445
System definition files	446
Systems	446
System definition files	446
System definition files and map source files	446
About relocating system definition files	446
Working with system definition files	446
To create a new system definition file	447
To open a system definition file	447
To display the systems contained in a system definition file	447
To close a system definition file	447
To save a system definition file	447
To create a system	447
To copy a system to another system	447
To rename a system	448
To delete a system	448
Importing and exporting system definitions	448
Exporting system definitions	448
Importing system definitions	449
Import and export options	449
Audit log	449
Map components	450
Source map components	450
Modifying a map component name	451
Compiled map components	451
Pseudo map components	451
Defining sources and targets for a pseudo map	451
Editing input and output cards	452
To edit an input or output card	452
To delete an input or output card	452
Creating a source map from a pseudo map	452
Viewing sources and targets of a map component	453
Card tooltips	453
Viewing multiple tooltips	453
Working with map components	453
Map names	454
Execution settings	454
Viewing execution settings	454
MapServerLocation	454
MapSettings	454
MapDelay	455
PendingExpiration	455
Max Concurrent Map Instances setting	455
TimeEvent	455
Pending Instance Thresholds setting	456
Input and output settings	456
Overriding map settings	457

Overriding card settings	457
Subsystem components	458
Subsystem component name	458
Execution settings	458
ServerName	458
ExecutionType	459
Inputs and outputs	459
Default settings	459
Saving map card positions	459
Overriding card source or target	459
Working with subsystems	459
Overview of server definitions and execution modes	460
Server implementation examples	460
Server definitions	460
Working with system definition files	446
Assigning a server to a system	461
Modifying a server definition	461
Importing and exporting server definitions	462
To export a server definition	462
To import a server definition	462
System execution modes	462
Generating system files	462
Building maps	463
To build all maps referenced by all source map components in a system	463
To build one map referenced by a source map component in a system	463
To build a collection of maps referenced by a collection of source map components within a system	463
To view the last build results at any point in time	463
Analyzing a System	463
Generating executable system files	464
To generate a command file	464
To generate a Launcher system file	464
Unresolved settings	465
Processing unresolved settings	465
System definition file differences	466
Comparing system definition files	466
Comparing systems	466
Comparing servers	466
Comparing deploy scripts	466
Comparing components	466
Comparing compiled map components	466
Comparing source map components	467
Comparing a psuedo map component	467
Comparing subsystem components	467
Comparing document links	467
Comparing system definition files	467
To compare system definition source files	467
To view system definition file differences	467
Viewing and printing system definition file differences	468
Links	468
Internal links	468
External links	468
Pseudo links	469
Using a pseudo link	469
To add a pseudo link	469
To delete a pseudo link	469
To display or hide links within a system window	469
Doc links	469
Using doc links	470
To add a document link for a component	470
To view a document link	470
To delete a document link	470
Overriding a card source or target	470
Deploying systems	471
Using deploy scripts	471
To access deploy scripts	471
To configure a script command	471
To add a deploy script	471
To remove a deploy script	471
To rename a deploy script	472
Build and transfer maps	472
To build and transfer a compiled map	472
To use the Maps Referenced Below option	472

To drag from the List view	472
To drag from the system window	473
To modify the component list	473
To specify the directory where the compiled map (.mmc) is transferred	473
Editing a path	473
Generate and transfer files	473
To generate and transfer a Launcher system file (.msl)	474
To generate and transfer a command file	474
Moving a map component to the deploy dialog	474
Moving subsystems to the deploy dialog	474
Transfer additional files	475
Running a deploy script	475
To run a deploy script	475
To view deploy script results	475
Return codes and error messages	475
Return codes and error messages	475
System analysis structure error messages	476
Systems using wildcards analysis structure error messages	477
Troubleshooting tips for error messages	477
Not enough memory to execute error message	477
"Version Mismatch" error message	477
Database Interface Designer	477
Database Interface Designer overview	477
Basic steps for using database data	478
Database Interface Designer basics	478
Starting the Database Interface Designer	478
Database Interface Designer user interface	479
The Navigator	479
Changing the appearance of the navigator	479
To show or hide the Navigator	479
To float the Navigator in the main window	479
To dock the Navigator	479
Database Interface Designer icons	480
Menu commands and tools	480
Menu	480
Toolbar	480
File menu	480
Edit menu	481
View menu	481
Database menu	482
Query menu	482
Tools menu	482
Window menu (Alt+W)	482
Help menu	483
Configuring the environment	483
Tools > Options	483
General options	483
Navigator options	483
Trace window options	483
Tables/views option	483
Confirmations options	484
Shortcut keys	484
Database/query files	484
Creating database/query files	485
Defining a database	485
Defining a query	486
Editing or deleting a database or query	486
Understanding the MDQ XML format	487
XML Schema	487
Saving database/query files	487
XML prolog text	487
XML data from XML-generated MDQ files	487
Attribute formatting	487
Password encryption	488
Backup copies	488
Processing MDQ files	488
Runtime execution	488
Design-time execution	488
Comparing database/query files	488
Criteria for analyzing differences	488
Creating a database/query file comparison	489
Viewing database/query file differences	489

Database/query file difference results	489
Database definition setting differences	490
Query differences	490
Variable differences	490
Unique database information	490
Stored procedure differences	490
Table differences	490
Defining variables in SQL statements	490
Defining a query with variables	491
Specifying the values at runtime	491
Generating type trees	491
From a table or view	492
From a stored procedure	493
From a message queue (Oracle AQ)	494
From a query	495
Printing reports	496
Database Interface Designer trace files	496
Database Interface Designer trace	496
Viewing Database Interface Designer trace files	497
Finding text in trace files	497
Database type trees	498
Table and query type tree structure	498
Special characters in type names	498
Characteristics of the DBSelect or DBTable type	499
Components and format of the row type	499
Delimited row group format	499
Fixed row group format	499
Defining the column type(s)	499
Binary column types	500
Column types for expressions	500
Specifying column aliases	500
Stored procedure type tree structure	501
Oracle AQ Message type tree structure	501
Database sources and targets	501
Using a database as a source	501
Defining a database source in the Map Designer	501
Database GET> Source settings	502
GET> Source> Command setting	502
Using a database as a target	502
Defining a database target in the map designer	502
Database PUT > Target settings	503
PUT > Target > Command setting	503
Database connections and transactions	503
Transactional control	503
Database connection management	503
Connection factors	504
Connection rules	504
Connection example	504
Updating database tables	504
Using key and update columns	504
Defining key and update columns	505
Specifying update mode	505
Using the Map Designer or Integration Flow Designer	505
Using an adapter command at execution time	505
Update key columns	505
Example using update columns	506
Database functions	506
Accessing database information in a map rule	506
Using DBLOOKUP and DBQUERY	506
Syntax1 - using a static MDQ file	507
Syntax2 - using dynamic adapter commands	507
Syntax2 formatting issues	508
Examples	508
Example 1 - obtaining a single column value	508
Example 2 - using DBQUERY to obtain multiple columns or rows	509
Example 3 - using DBQUERY to provide map input to RUN function	509
Example 4 - using WORD to parse multi-column output from DBQUERY	509
Example 5 - using DBQUERY with adapter commands to obtain a single column value	510
Uses	510
Using bind values in database functions	510
Using stored procedures	511
Calling stored procedures	511

Database-independent syntax for calls	511
Examples using stored procedures	512
Returning the Value from a stored function	512
Using a stored procedure as an input	512
Using a query to execute a stored procedure	512
Using a stored procedure as an output	513
Stored procedures with object type parameters	513
Database triggers	514
Database triggers overview	514
Database support	514
Installation requirements	514
Column-based triggering	515
Tables created for triggering	515
Maintaining triggering tables	515
Handling unexpected shutdowns	515
Handling truncated tables	516
Table-based triggering	516
Row-based triggering	516
Column-based triggering	516
Issues for both row- and table-based triggering	516
Issues for row-based triggering only	517
Defining a trigger using a database source	517
Using a database as a map trigger	517
Defining a trigger for a query	517
Defining events	518
Specifying a combination of different event classes	518
Specifying AND or OR	518
Specifying when	519
Format of the when expression	519
Specifying triggers on the command line	519
Using the Integration Flow Designer to enable triggers	519
Debugging and viewing results	520
Troubleshooting tools	520
Database trace files	520
Format of database trace files	520
Producing the database trace in the Database Interface Designer	520
Producing the database trace during map execution	521
Using the Trace adapter command	522
Database trace for a valid source	522
Database trace for a source with errors	522
Database trace for a valid target	523
Database trace for a target: missing required value	524
Database trace for a target - using the bad data adapter command (-BADDATA)	525
Database trace for a target with -UPDATE off	526
Database trace for a target: DBLOOKUP/DBQUERY functions	527
Tracing database errors only	528
Viewing database source and target data	529
Using backup settings	529
Capturing data not processed	529
Database audit files	530
DBMS trace utilities and SQL command tools	530
Trace utilities	530
SQL command tools	530
REST API	
REST API V2.0	531
Deployment APIs	531
Map and Flow run APIs	531
PUT /v2/run/url	531
POST /v2/run/url	532
Flow Schedule APIs	532
POST /v2/schedule/url	532
DELETE /v2/schedule/url	532
Flow variables	
Flow functions	533
Command Server	
Using a Command Server	533
Command Server Overview	533
Building a platform-specific map	533
Executing a map	534
Using execution commands	534
Input data validation	534

Output objects built	534
Troubleshooting tools	534
Command Server display	535
Command Server return codes and messages	535
Map audit logs and trace files	535
Map audit log	535
Map trace files	538
External interfaces for applications, EXIT and RUN	538
Windows platforms	538
Running the map examples	538
Using the Command Server	539
Help	539
z/OS Batch	539
Running the map examples	539
Using the Command Server	540
Execution command line (JCL PARM statement)	540
Example 1	540
Example 2	540
Example 3	540
Example 4	540
Identifying data sources and targets	541
Names to avoid	541
Temporary datasets	541
JCL	542
Overriding Language Environment (LE) runtime options	543
z/OS Batch execution commands	543
Input Dataset Override (-I)	544
No DTXLOG (-NL) or (/NL)	545
Output Dataset Override (-O)	545
Workfiles in Memory (-WM)	545
Print Uppercase (-+)	545
Command Dataset (-@)	545
Problem Resolution (-\$)	545
Record Separators (/V)	546
Troubleshooting	546
EBCDIC-encoded XML trace files	546
VSAM considerations	546
Using the EXIT function	547
Using the RUN function	547
Reusing work files	547
Output file same as input file	548
Appended output	548
Resource names	548
z/OS CICS	548
Running the map examples	549
Using the CICS Execution Option	549
z/OS CICS execution commands	549
Audit (-A)	550
Input Dataset Override (-I)	551
Execution Log (-L)	551
Output Dataset Override (-O)	552
Enqueuing on Data Files (-Q)	552
Trace (-T)	553
Workfiles in Memory (-WM)	553
Problem Resolution (-\$)	553
Print Uppercase (-+)	553
Eliminate AICA (ICVR Timeout) Abends (-!n)	553
Specifying storage medium	554
Specifying output fixed format record length	554
Variable length records	555
The :W option	555
The :P option	555
Using EXEC CICS LINK passing storage buffers	556
Using the EXIT function	556
Using the RUN function	557
Preloading IBM Transformation Extender maps and .mdq files in a z/OS CICS environment	557
Configuring the CICS preloader utility	557
The preloader control file	557
Identifying the maps and .mdq files in the DTXnnnIN preloader control file	558
Map and .mdq file DD statements in the CICS startup JCL	558
Map or .mdq files in a z/OS flat file	558
Map or .mdq files in a PDS or PDSE	558
Running the IBM Transformation Extender CICS preloader utility	558
Abnormal termination	559

The DTXMPULD utility	559
Resource names	559
Multiple IBM Transformation Extender versions	559
Enabling the multiple IBM Transformation Extender feature	559
Using the multiple installations	560
Resource names for z/OS platform	560
Resource name aliases	560
Resource name limitations for files and DDNAMES	560
Virtual server names	560
Specifying resource name file for batch	561
DDNAMES	561
Specifying resource name file for CICS	561
VSAM	561
Resource configuration files	561
Launcher	
Launcher introduction	561
Installation considerations	562
Optional local z/OS administration interfaces	562
Launcher overview	562
Launcher HTTP Listener overview	562
Using the Launcher	563
Basic steps	563
Launcher architecture	563
Triggers: Asynchronous communication	564
Synchronous coordination	564
System execution consistency	564
Map function and requirements	564
Defining a system	564
Getting started	565
Creating systems	565
Launcher Administration	565
Accessing Launcher Administration	565
Using Launcher Administration	565
Automatic startup option	566
Configuring automatic restart	566
Connection ports	566
User profiles	567
Deployment directories	567
Resource resolution	567
Cluster support	568
Firewall configuration	568
HTTP Listener startup configuration	568
Automatic restart for launchers and HTTP Listeners	569
About multiple Launchers	569
Preparing to execute defined systems	569
Starting and stopping the Launcher	569
Modifying systems while the Launcher is running	570
Configuration parameters	570
HTTP Listener settings in config.yaml	570
Enabling or disabling settings	571
M4File	571
Logging	571
FileListenCount	571
ListenSleepTime	571
Launcher	571
Logging categories descriptions	572
ResourceCfgFile	572
LoadPDH	572
MaxThreads	572
TriggerTime	573
Launcher logging	573
InitPendingHigh	575
InitPendingLow	575
WatchMaxThreads	575
DisableMaxThreads	575
InitPendingIdleMS	575
RunMapCacheMaxNum	576
LogFilePermissions	576
MapAdapterTraceOff	576
CircularLogSize	576
CircularLogFileNum	577
Resource Manager	577
Logging categories	577
Disable	577

TransManager	577
Stream configuration options	577
Connection Manager	578
How the Connection Manager works	578
Connection Manager settings	578
Logging categories	578
Configuring Java-based adapters	582
Trace and log files naming conventions	582
Result codes in log files	582
Launcher settings	582
Accessing Launcher settings	583
Launcher settings window	583
MapServerLocation	583
MapSettings > MapAudit	583
Unique names for audit log files	584
MapSettings > WorkSpace	584
MapDelay	584
PendingExpiration	585
Max concurrent map instances	585
TimeEvent	585
TimeEvent > Switch	585
TimeEvent > BeginAfter	585
TimeEvent > Trigger	586
Trigger > Interval	586
Interval > OmitDays	587
Interval > OmitHours	587
TimeEvent > SkipIfBusy	587
TimeEvent > EventCoordination	587
Pending instance thresholds	588
MapInstance Timeout	588
Trigger event settings	588
Time events	588
Source events	588
Time and source event coordination	589
Launcher input trigger files	589
Source and target settings	589
Adapter classes	590
Wildcards in adapter names	590
File adapters	590
Event triggers	591
Wildcards	591
File names, path names, directories, and overrides	591
OnFailure and Retry settings	591
Effect of OnSuccess card setting on file event triggers	591
Troubleshooting tip	592
Database adapters	592
Characteristics of database adapters	592
Application adapters	593
Characteristics of application adapters	593
Message adapters	593
Characteristics of message adapters	593
Adapters that support event triggers	594
Connection testing adapters	594
Subsystem component execution settings	594
Launcher settings dialog	595
Server	595
ExecutionType	595
Inputs and outputs	595
Map initiation scenarios	595
Coordinating and using wildcards	596
Single source name	596
Matching multiple source names	596
Matching source to target names	596
Multiple wildcards in a source name	596
Threading wildcards across multiple maps	596
Multiple map instances in the same watch	597
Using time events	597
Case 1: Different map instances can run at the same time	597
Case 2: Different map instances can run sequentially	597
Case 3: Skip If Busy can be used to omit map instances from starting	597
Using source events	597
Case 1: Different map instances run at the same time, if all sources and targets are different.	598
Case 2: Different map instances run at same time or sequentially based on source and target settings	598

Same map in different watches	598
Cooperative file listening: Multiple watches monitor a single trigger directory	598
Horizontal scaling	599
Cooperative file listening and fault tolerance	599
Cooperative file listening and load balancing	599
Cooperative file listening configuration	599
Cooperative file listening requirements	599
Cooperative file listening config.yaml file and Launcher settings	600
Error directory configuration	600
Keep directory configuration	600
Balance the workload across watches and scale for large data	601
Different maps that use the same source	602
Impact of Delete settings	602
Different maps use the same target	602
One map's target is another map's source	602
HTTP requests trigger maps	602
HTTP Listener map execution results overview	603
Status code command (-STATUS_CODE)	603
Status page commands (-STATUS_PAGE and -STATUS_PAGE_DEFAULT)	603
HTTP Listener map performance and throughput	603
HTTP Listener connection performance	604
HTTP Listener browser access	604
HTTP Listener logging	604
Supported cryptographic protocols for the HTTP Listener	605
Map initiation states	605
Initiation pending state	605
Resource pending state	605
Map initiation tip	605
Management Tools	605
Overview	606
Management Tools for Windows platforms	606
Management Tools for UNIX platforms	606
Unique keys in the Launcher Management Tools	606
HTTP Listener default port settings	606
Automatically-generated log files	607
Launcher log file	607
UNIX signal-generated codes in the Launcher log file	607
Management Console log file	608
Communication log file	608
Java Management Tool log file permissions	608
Management Console	608
Features of the Management Console	608
Using the Management Console	608
Management Console quick start list	609
Management Console icon descriptions	609
Required steps to perform outside of the Management Console	609
Getting started	609
Controlling systems	610
Monitor option	610
Retry option	611
Snapshot option	611
Launcher options	611
Override Configuration options	612
Management Console statistics	612
System statistics	612
Summary statistics	612
Performance	612
Status	612
Historical	613
Status statistics	613
Pending	614
Adapter connections	614
Historical information	614
Map failures	614
Adapter connections	614
Map function failures	615
Configuration information	615
System tab	615
Adapter connections tab	615
Launcher HTTP Listener summary, status, and configuration	615
Dynamic adapter tracing	617
Using the dynamic adapter trace feature	617
Enabling dynamic adapter tracing	617

Configuration pane	618
Configuration pane elements	618
Using wildcards	619
Output files	619
Dynamic logging	619
Using the dynamic logging feature	620
Enabling dynamic logging	620
Configuration pane	620
Configuration pane elements	620
Output files	621
Logging messages for Trace	621
Launcher trace 01xx	621
Resource Manager trace 03xx	622
Connection Manager trace 05xx	622
Logging messages for Debug	623
Launcher debug 11xx	623
Resource Manager debug 13xx	624
Connection Manager debug 15xx	624
Logging messages for Info	625
Launcher info 21xx	625
Resource Manager info 23xx	626
Connection Manager info 25xx	627
Logging messages for Warning	627
Launcher warning 31xx	627
Resource Manager warning 33xx	627
Connection Manager warning 35xx	628
Logging messages for Error	628
Launcher error 41xx	628
Resource Manager error 43xx	629
Connection Manager error 45xx	629
Logging messages for Fatal	630
Launcher fatal 51xx	630
Resource Manager fatal 53xx	630
Connection Manager fatal 55xx	631
Logging option	631
Launcher-created log file permissions	631
Example	631
Log file types	631
Launcher activity log files	632
Debug log files	632
Diagnostic log files	632
Environmental debug (UNIX)	632
System and watch status in JSON format	633
Launcher Monitor	634
How the Monitor works	634
Using the Launcher Monitor	634
Configuring the Monitor display	635
Watches	635
Adjusting colors	635
Snapshots	636
Snapshots on demand	636
Snapshot settings	636
Snapshot Viewer	636
How are Snapshots created?	636
Command line options	637
Syntax conventions	637
Running the setup script	638
Net pause and net continue commands	638
Launcher command line syntax	639
Status command	639
Snapshot commands	640
Using Snapshot commands	640
File storage commands	641
Display commands	641
Return codes for system-based command options	641
Starting, stopping, pausing, and resuming the Launcher service	642
Starting, stopping, pausing, and resuming systems	643
Trace command	643
Logging command	643
List system information command	644
Launcher Administration command line syntax	644
List the configured administration options	645
General options	646

Access options	646
Deployment directories options	647
Advanced options	648
HTTP Listener options	648
HTTP Listener startup configuration	568
Import and export properties options	649
Management Console command line syntax	650
HTTP Listener command line syntax	651
Multiple processes	652
Activities summary	652
Installation	653
Using resource management	653
Global resource management	653
Launcher systems run options	653
Using Launcher Administration to run systems as separate processes	653
Benefits of using separate Launcher processes	654
General tab	654
Selecting separate Launcher processes	654
Connection ports	654
Access tab - User Properties dialog	655
Adding and removing processes and system files with the Deployment Directories tab	655
Configuring Launcher processes in the File List dialog	656
Property (Directory)	656
Property (File)	656
Configuring Launcher processes to start automatically	657
Modifying systems while the Launcher is running	657
Changing file configurations	657
Restrictions when modifying deployment directories and processes	657
Modifying administration user access	658
Configuring the initialization file	658
Managing the Launcher process	658
Starting the Launcher service or Launcher daemon	659
Monitoring systems through the Management Console	659
Shell script and batch file (launcher.sh and launcher.bat)	659
UNIX shell script	659
Windows batch file	659
Using the Launcher Monitor	660
Debug trace and log files	660
z/OS Launcher	660
Overview	660
Referenced terms	660
Architecture	660
Utilities and tools	661
JCL tools	661
z/OS UNIX System Services Command Server	661
Native file system	661
File types	662
File access	662
Allocation	662
Map names	662
Compiled map name specifications	662
Related data file locations	662
DEFAULT	663
CUSTOM	663
UNIQUE	663
Map sources and targets	663
Record Terminators (/V)	664
Allocation parameters	664
Map sources	664
Map targets	665
File triggering	665
File-creation triggering	665
File-update triggering	665
Wildcards in native file system names	665
File pre-allocation	666
Incomplete file creation or update	666
Local z/OS administration interfaces	666
Overview	666
Activities summary	667
Using the Launcher Management Tools	667
Using the Management Console	667
Starting the Launcher Monitor	667
Default port settings	667

Environment settings	668
ISPF panel interface	669
Commands	669
Menu options	669
Server information	669
Managing multiple Launchers	670
z/OS operator console interface	670
Commands	670
Managing multiple Launchers	671
Multiple processes in the native administration interfaces	671
Troubleshooting tip	671
SNMP Agent	
SNMP Agent	671
SNMP overview	671
What is MIB?	672
MIB data objects	672
MIB variables	672
How an SNMP Agent works	672
How It all fits together	672
Configuration	672
Installation	672
To change the SNMP Agent locale	673
Configuring the SNMP Agent	673
SNMP administration utility	673
General tab	673
Startup Settings tab	674
Disabling the logging of messages to the output console	675
SNMP security	675
SNMP communities	675
Defining SNMP communities	675
SNMP user-based security model	676
Defining SNMPv3 users	676
Adding users to the security file	676
Starting the SNMP Agent	677
SNMP operations and thresholds	677
GET	677
SET	677
Traps	678
Defining SNMP traps	678
Supported traps	678
Thresholds	680
Setting threshold limits	680
Threshold notification	680
Troubleshooting	680
DTX MIB	680
Listener properties	681
Map properties	681
Adapter properties	681
System objects	682
Table objects	682
Resource Adapters	
Resource Adapters	683
Adapters overview	683
Importers	684
System requirements	684
General rules for adapter commands	684
Command syntax summaries	684
Adapter command aliases	685
Configuring Java-based adapters	685
UNIX only	685
API usage	685
Encoding and decoding data	686
Supported adapters	686
Limitations	686
Encoding	686
Decoding	686
Transporting	686
Encode/decode scenarios	687
Input card or GET	687
Output card or PUT	687
GET function	687
Transport encode/decode examples	687
SOAP Input card	687

SOAP Output card	688
SOAP request/reply in a GET function	688
Using resource adapters	688
Using an adapter in the map designer	688
Specifying the FTP adapter in the map designer	688
Using an adapter in a map or component rule	689
Examples of database adapter commands	689
Using an adapter in the Integration Flow Designer	689
Specifying the TIBCO RV adapter in the Integration Flow Designer	689
On the command line	690
Database-specific adapter commands on the command line	690
Command line examples	690
For Windows sources and targets on the command line	691
For UNIX sources and targets on the command line	691
Functions	691
Using the GET function	691
Using the PUT function	691
Using the RUN function	692
Using wildcards	692
Scope settings and FetchUnit interpretation	692
Using global transaction management	693
File Adapter	693
Text File Importer	693
Supported character sets	693
Western character sets, international and non-international versions)	693
Japanese character sets (international version)	693
Sink adapter	693
Using the Sink adapter	694
Application adapter	694
Source and target card settings	694
Source settings	694
Target settings	694
Communication adapters	694
VAN access through the FTP adapter	695
Specifying the FTP adapter for GXS VAN access	695
GXS VAN command syntax	695
Messaging adapters	695
Message flow	695
Adapter transactions	696
Transactional control	696
Connection management	696
Messages as a data source	696
Example of data processing modes	696
Messages as a data target	696
Database adapters	696
Supported database adapters	697
Restrictions and limitations	697
Using database-specific adapters	697
Adapter usage in functions	698
Adapters in Input Data and Output Data Settings	698
List of commands	698
Audit (-A or -AUDIT)	699
Bad Data (-BD or -BADDATA)	699
Connect String (-C or -CONNECT)	699
Commit by Card (-CC or -CCARD)	699
Commit by Statement (-CS or -CSTMT)	700
Delete (-D or -DELETE)	700
Database Name (-DN or -DBNAME)	700
Database Adapter Type (-DT or -DBTYPE)	700
Data Source (-DS or -SOURCE)	701
File (-F or -FILE)	701
Global Transaction Management (-GTX)	701
Database/Query File (-M or -MDQ)	701
Password (-PW or -PASSWORD)	702
Stored Procedure Name (-PR or -PROC)	702
Query (-Q or -QUERY)	702
Row Count (-RC or -ROWCNT)	702
SQL statement (-S or -STMT)	702
Table Name (-TB or -TABLE)	703
Trace (-T or -TRACE)	703
Trace Error (-TE or -TRACEERR)	703
Trigger (-TR or -TRIG)	703
Update (-UP or -UPDATE)	704

User ID (-US or -USER)	704
Variable (-V or -VAR)	705
Generating schemas using mtmaker	705
Overview	705
Generating a schema script	705
Using mtmaker with a database/query file	705
Using mtmaker with an .mdq File	706
Using mtmaker without a database/query file	706
Using mtmaker without an .mdq File	706
General parameter rules	706
List of parameters	706
Database Name (-B)	707
Data Source (-D)	707
Schema File (-F)	707
Stored Procedure Name (-G)	708
Database/Query File (-M)	708
Category Type (-N)	708
Schema Script File (-O)	708
Password (-P)	708
Query Name (-Q)	708
SQL statement (-S)	708
Table Name (-T)	708
User ID (-U)	708
Variable Name (-V)	709
Format (-X)	709
Database Adapter Type (-Z)	709
Return codes and error messages	710
General adapter messages	710
Database-specific adapter messages	711
Apache ActiveMQ Adapter	714
Authentication connections	714
Adapter properties and commands	715
Examples	718
Apache HDFS Adapter	718
System requirements	718
HDFS Adapter commands overview	718
HDFS Adapter commands	719
Protocol scheme -S (-SCHEME)	720
WebHDFS host name -H (-HOST)	720
WebHDFS port number -P (-PORT)	721
Username -U (-USER)	721
Path prefix -PP (-PATHPREFIX)	721
Filename -F (-FILE)	721
Chunk size -CS (-CHUNKSIZE)	721
Single chunk mode -SCM (-SINGLECHUNKMODE)	721
Immediate write mode -IWM (-IMMEDIATEWRITEMODE)	721
Block size -BLS (-BLOCKSIZE)	721
Buffer size -BUS (-BUFFERSIZE)	721
Overwrite -OW (-OVERWRITE)	722
Permission flags -PER (-PERMISSION)	722
Replication factor -REP (-REPLICATION)	722
Cross-Site Request Forgery Prevention Header -CSRHF (-CSRHF)	722
Extended attributes -XATTR (-XATTR)	722
Truststore file full path -TSL (-TRUSTSTORELOCATION)	722
Truststore password -TSP (-TRUSTSTOREPASSWORD)	722
Keystore file path -KSL (-KEYSTORELOCATION)	723
Keystore password -KSP (-KEYSTOREPASSWORD)	723
Key password -KP (-KEYPASSWORD)	723
Authenticate server -AS (-AUTHSERVER)	723
Authenticate client -AC (-AUTHCLIENT)	723
Verify host name -VH (-VERIFYHOSTNAME)	723
Kerberos mode -KM (-KERBEROSMODE)	723
Login configuration file location -LCFL (-LOGINCONFIFILELOCATION)	723
Kerberos configuration file location -KCFL (-KERBEROSCONFIGFILELOCATION)	724
Kerberos authentication user -KAU (-KERBEROSAUTHUSER)	724
Kerberos authentication password -KAP (-KERBEROSAUTHPASSWORD)	724
Kerberos configuration file -KCN (-KERBEROSCONFIGNAME)	724
Basic authentication user -BAU (-BASICAUTHUSER)	724
Basic authentication password -BAP (-BASICAUTHPASSWORD)	724
Adapter log file full path -T, -T+, -TV, -TV+, -TE, -TE+	724
HDFS Adapter transaction settings	724
Apache Kafka Adapter	725
Authenticating Connection	725

Adapter properties and commands	725
Adapter commands for consumers and producers	726
Adapter commands for producers	729
Adapter commands for consumers	730
Amazon S3 Adapter	733
Introduction	733
Adapter Properties and Commands	734
Examples	737
Amazon SQS Adapter	737
Amazon SNS Adapter	739
Azure Service Bus Adapter	741
Introduction	741
Defining Connections	741
Azure Service Bus adapter as a source	741
Azure Service Bus adapter as a target	741
Adapter Properties	741
Endpoint	742
Shared Access Key Name	742
Shared Access Key	742
Queue Name	742
Content Type	742
Session Identifier	742
Receive Mode	743
Message Limit	743
Message Timeout	743
Logging	743
Append Log	743
Log File Path	743
Examples	743
Azure SQL Adapter	744
Introduction	744
Azure SQL Adapter commands	744
Server Name	745
Port	745
Database	745
User	745
Password	745
Encrypt	745
Trust Server Certificate	745
Hostname in Certificate	745
Login Timeout	745
Autocommit	745
Table	746
Query	746
Procedure	746
DML	746
Schema	746
Logging	746
Append Log	746
Log File Path	746
Pre SQL	746
Post SQL	747
Write Mode	747
-QTY and -LSN adapter commands	747
-ARRAYSIZE adapter command	747
Examples	747
Azure Blob Storage Adapter	747
Introduction	748
Adapter Properties and Commands	748
Account Name	748
Access Key	748
Endpoint Suffix	748
Endpoints Protocol	749
Container Name	749
Write Mode	749
Block Size	749
Read Mode	749
Overwrite Blob	749
Blob Name	749
Logging	749
Log File Path	750
Convert Data	750
Data Format	750
Header	750

Quote	750
Delimiter	750
Escape	750
Separator	750
Character Set	750
Limit	750
Sample Size	751
Worksheet Index	751
Select Source Tables	751
Find Array	751
Array Path	751
Examples	751
B2B Advanced Communications Storage Adapter	751
Batch File and Shell Script Adapters	753
System requirements	753
Command aliases	753
Commands for the Batch File and Shell Script adapters	754
Audit (-A or -AUDIT)	754
Command line interpreter (-CL or -CLI)	754
Batch files in Windows environments	754
Shell scripts in UNIX environments	755
Command file (-C or -CMD)	755
Transfer file (-F or -FILE)	755
Inline output (-I or -INLINE)	755
Poll process (-P or -POLL)	756
Show mode (-S or -SHOW)	756
Standard input (-< or -STDIN)	756
Trace (-T or -TRACE)	756
Syntax summary for Batch File and Shell Script adapters	757
Data sources	757
Data targets	757
Return codes and error messages	757
CICS Adapter	758
System requirements	758
Command alias	758
CICS Adapter commands	758
Bridge Facility Like (-BFL)	759
Bridge Keep Time (-BKT)	759
Driver Transaction Name (-DT)	759
Host (-HOST)	759
Password (-PW)	759
Port (-PORT)	759
Service (-SERVICE)	760
Timeout (-TIMEOUT)	760
Trace (-T)	760
Transaction Name (-TRAN)	760
User ID (-UID)	761
Syntax summary	761
Data sources	761
Data targets	761
Example 1	761
Example 2	761
Using the CICS adapter	762
Adapter capabilities	762
Adapter limitations	762
Defining a bridge transaction for use with the CICS adapter	762
Cipher Adapter	762
IBM GSKit requirements	763
Cipher adapter commands	763
COM Adapter	764
System requirements	764
COM Adapter command alias	764
COM Adapter commands	764
Input card configuration file (-CFG)	764
Prequel (-PRE) command for COM	765
Post (-POST) command for COM	765
Program ID (-PROGID) command for COM	765
Set Property (-PROP) command for COM	766
Trace (-T) command for COM	766
Syntax summary	766
COM adapter data sources	766
COM adapter data targets	766
Determining the dispatch ID for a method or property	767
Viewing the IDL file	767

Running the oleview utility	767
Using the adapter	767
COM Automation advantages	768
Adapter capabilities	768
Supported data types	768
Object persistence	768
DCOM server considerations	769
Standard COM Automation terms	769
Troubleshooting	769
Error code propagation	769
Type Library Importer	769
Using the Type Library Importer	769
Structures	770
Arrays	770
SAFEARRAY Arrays	770
COM Automation objects	771
Connect:Direct Adapter	771
System requirements	771
Configuration requirements	771
Connect:Direct adapter command overview	771
Command aliases	772
Primary Node (-PNODE) command	773
Primary node user (-PUSER) command	773
Primary node password (-PPASS) command	773
Secondary node (-SNODE) command	773
Secondary node user (-SUSER) command	774
Secondary node password (-SPASS) command	774
File name (-FILENAME) command	774
Process name (-PROCNAME) command	774
File disposition (-DISPOSITION) command	774
Binary transfer (-BINARY) command	775
Restart checkpoint (-CKPT) command	775
Wait interval (-WAIT) command	775
Directory restriction (-DIRRESTRICT) command	776
Create trace file (-TRACE) command	776
Append trace file (-TRACE+) command	776
Trace adapter errors (-TRACEERR) command	777
Append adapter errors (-TRACEERR+) command	777
Use translation table (-Xlate) command	777
DB2 (Windows/UNIX) Adapter	777
System requirements	777
Database columns and types	778
Item type properties	778
Date and Time Formats	779
Database Interface Designer settings	779
Stored procedures native call syntax	779
DB2 (Windows/UNIX) Adapter commands	780
Adapter-specific command list	780
Database adapter type (-DBTYPE) adapter command for DB2 (Windows/UNIX)	780
Row count (-ROWCNT) adapter command for DB2 (Windows/UNIX)	780
Data Source (-SOURCE) adapter command for DB2 (Windows/UNIX)	780
Adapter commands for a source	781
Adapter commands for a target	781
Binding values in DBLOOKUP and DBQUERY	781
Limitation when using DB2	781
Specifying the data type	781
Bulk copy example files	782
Restrictions and limitations	782
Return codes and error messages	782
DB2 (z/OS ODBC) Adapter	782
System requirements	782
Database columns and types	783
How DB2 data types convert to item types in a type tree	783
Date and time formats	783
Database Interface Designer settings	784
DB2 (z/OS ODBC) Adapter commands	784
DB2 (z/OS ODBC) Adapter-specific commands	785
Database Adapter Type (-DBTYPE) adapter command	785
Row count (-ROWCNT) adapter command for DB2 (z/OS ODBC)	785
Data Source (-SOURCE) adapter command	785
Adapter commands for a source	785
GET > Source > Command setting	785
Database execution command: input source override (-ID)	786
DBLOOKUP and DBQUERY functions	786

GET function	787
Adapter commands for a target	787
PUT > Target > Command setting	787
Database execution command: output source override (-OD)	788
PUT function	788
Using tdfmaker to define update keys	789
Using tdfmaker on z/OS	789
Sample tdfmaker.JCL file	789
Using mtssmaker on z/OS	789
mtssmaker parameters for z/OS	790
Sample JCL files for mtssmaker on z/OS	790
DB2 (z/OS ODBC) Adapter limitations	791
Return codes and error messages	791
DB2 (z/OS) Adapter	791
System requirements	791
Binding the application plan	791
Database columns and types	791
Database Interface Designer settings	792
DB2 (z/OS) Adapter commands	792
DB2 (z/OS) Adapter-specific commands	792
Database adapter type adapter command for DB2 (z/OS)	792
Database/query file adapter command for DB2 (z/OS)	793
DB2 (z/OS) Adapter source commands	793
GET > source > command for DB2 (z/OS) Adapter	793
DB2 (z/OS) Adapter database execution command (input source override)	793
DBLOOKUP or DBQUERY functions	794
GET function	794
DB2 (z/OS) Adapter commands for a target	794
PUT > target > command for DB2 (z/OS) Adapter	795
DB2 (z/OS) Adapter database execution command (output source override)	795
PUT function	795
DB2 (z/OS) Adapter limitations	795
Return codes and error messages	796
Extreme Scale Adapter	796
System requirements	796
Extreme Scale adapter command overview	796
Command aliases	797
Connection string (-CONNSTRING) command	797
Grid (-GRID) command	797
Hash map (-HASHMAP) command	798
Key (-KEY) command	798
Reference (-REFERENCE) command	798
Command (-COMMAND) adapter command	798
Now (-NOW) command	799
SQL statement (-SQL) command	799
Create trace file (-T) command	799
Trace adapter errors (-TE) command	799
Email Adapters	800
System requirements	800
Command alias	800
Using adapter commands	800
Commands for email adapters	800
Attachment (-ATT or -ATTACH)	801
Multiple attachments	802
Attachment name (-AN or -ATTNAME)	802
Audit (-A or -AUDIT)	802
BCC recipient (-BCC)	802
CC recipient (-CC)	802
From (-FROM)	803
Header (-HDR)	803
Login (-L or -LOGIN)	803
Listen (-LSN)	803
Markread (-MARKREAD)	804
Password (-P or -PASS)	804
Port (-PORT)	804
Post office (-PO)	804
Priority (-PRI or -PRIORITY)	804
Protocol (-PR or -PROTO)	805
Raw (-RAW)	805
Receipt (-RCPT or -RECEIPT)	805
Server (-SVR or -SERVER)	805
SSL protocol (-SPROTO)	805
SSL encryption strength (-STR)	806
Subject (-SUB or -SUBJECT)	806

Text (-TXT or -TEXT)	806
To recipient (-TO)	807
Trace (-T or -TRACE)	807
Type (-TYPE)	807
Userid (-UID or -USERID)	807
Username (-U or -USER)	808
Syntax summary	808
Lotus Notes adapter commands	808
Data sources for LNOTES	808
Data targets for LNOTES	808
Internet email adapter commands	809
Data sources for INET	809
Data targets for INET	809
Email adapters example using data sources and targets	809
Example: send mail using the Lotus Notes protocol	809
Example: send mail using internet email protocol	810
Command line example: Lotus Notes override	810
Command line example: internet email override	810
Troubleshooting email adapters	810
Trace log file	810
Internet email	810
Lotus Notes	810
Return codes and error messages	811
Messages	811
Excel Adapter	811
System requirements	812
Excel Adapter commands	812
Transaction scope and document persistence in memory	813
data_from_map option	813
Add document (-ADOC) command	813
Remove document (-RDOC) command	813
Replace document (-UDOC) command	814
Get document data (-GDOC) command	814
Excel workbook (-FILE) command	814
Worksheet (-WRKSHEET) command	814
Read Mode (-READMODE) command	814
Worksheet Identifier (-IDENTIFIER) command	814
Worksheet Index (-WRKSHEETINDEX) command	815
Template (-TEMPLATE) command	815
Excel transformation (-XLS) command	815
Google Cloud Storage Adapter	815
Authenticating Connections	816
Adapter properties and commands	816
Examples	818
Google Pub / Sub Adapter	818
Authenticating Connections	819
Adapter properties and commands	819
Examples	821
FTP adapter overview	821
FTP adapter introduction	821
Adapter Properties and Commands	822
FTP Adapter command examples	829
Syntax summary	829
Troubleshooting	831
Return codes and error messages	832
GZIP/ZLIB Adapter	833
GZIP/ZLIB Adapter overview	833
System requirements	833
Command alias	833
GZIP/ZLIB Adapter commands	834
Action adapter command for GZIP/ZLIB	834
File adapter command for GZIP/ZLIB	834
Stream adapter command for GZIP/ZLIB	834
Trace adapter command for GZIP/ZLIB	835
Syntax summary	835
Command behavior	835
HL7 MLLP Adapter	836
System requirements	836
How to use the HL7 MLLP adapter in a map	836
Conversation support	837
HL7 MLLP adapter command overview	837
Command alias	838
Client connection (-CCON) command	838
Host (-H) command	839
Listen (-LSN) command	839

Number of event messages (-NEM) command	839
Port (-P) command	840
Quantity (-QTY) command	840
Service (-S) command	840
Trace (-T) command	841
HTTP Adapter	841
HTTP Adapter Introduction	841
HTTP Adapter commands	842
Syntax summary	848
Troubleshooting	849
Return codes and error messages	850
IBM MQ Adapter	850
IBM MQ message content	851
Transmission queues	851
IBM MQ client or server adapter	851
System requirements	851
Client system	851
Server system	852
Command aliases	852
GET > Source settings	852
GET > Source > Command setting	852
Source > Transaction > OnSuccess setting	852
Source > Transaction > OnFailure setting	853
Source > Transaction > Scope setting	853
Source > Transaction > Warnings setting	853
PUT > Target settings	853
PUT > Target > Command setting	853
Target > Transaction > OnSuccess setting	854
Target > Transaction > OnFailure setting	854
Target > Transaction > Scope setting	854
Target > Transaction > Warnings setting	854
PUT > Target > Retry settings	854
Target > Retry > Switch setting	855
Target > Retry > MaxAttempts setting	855
Target > Retry > Interval setting	855
IBM MQ adapter commands	855
Channel definition (-CD)	856
Cluster queue manager name (-CQMN)	857
Examples	857
Coded character set identifier (-CCSID)	857
Complete message (-CMSG)	858
Convert (-CVT)	858
Correlation ID (-CID)	858
Default header (-DH)	858
Error queue manager name (-EQMN)	859
Error queue name (-EQN)	859
Message expiration (-EXPIRY)	859
Global transaction management (-GTX)	859
Group message (-GRP)	860
Group message 2 (-GRP2)	860
Example 1	861
Example 2	861
Header (-HDR)	861
Hex correlation ID (-HCID)	862
Hex group message ID (-HGRP)	862
Hex message ID (-HMID)	862
Ignore group message (-XGRP)	863
Listen (-LSN)	863
Message buffer size (-MBS)	863
Message format (-FORMAT)	863
Message ID (-MID)	864
MQOPEN options (-MQOO)	864
Packet (-PKT)	865
Password (-P)	865
Quantity (-QTY)	865
Queue manager name (-QMN)	866
Queue name (-QN)	866
Refresh message cursor (-REFRESH)	866
Require all messages (-ALLMSG)	867
Require all segments (-ALLSEG)	867
Trace (-T)	867
Transmission Error queue name (-XEQN)	868
Transmission queue name (-XQN)	868

User (-U)	868
Version 2 (-V2)	869
Wait Interval (-WI)	869
Syntax summary	869
Data sources	869
Data targets	870
Wildcard support	870
Using the messaging adapter	870
Threading for IBM MQ	871
Synchronizing multiple watches on the same queue	871
Group message support	871
Physical message	871
Logical message	872
Message group	872
Incomplete message segments	872
Reassembly of segments	872
Retrieval of group messages	872
Example files	872
Using the sample type tree	873
Mapping message data only	873
Mapping the message descriptor	873
Mapping the IBM MQ Link for R/3 header	873
Troubleshooting	873
Trace log	873
Return codes and error messages	874
Messages	874
IBM MQ and error code relationships	874
Informix Adapter	875
System requirements	875
Database columns and types	875
Item type properties	876
Date and time formats	876
Database Interface Designer settings	877
Database Definition dialog box	877
Stored procedures native call syntax	877
Informix Adapter commands	878
Adapter-specific command list	878
Database adapter type (-DBTYPE)	878
Row count (-ROWCNT)	878
Data source (-SOURCE)	878
Adapter commands for a source	878
GET > Source > Command setting	879
Database execution command: input source override (-ID)	879
DBLOOKUP or DBQUERY functions	880
GET function	880
Adapter commands for a target	881
PUT > Target > Command setting	881
Database execution command: output source override (-OD)	881
PUT function	882
Binding values in DBLOOKUP/DBQUERY	882
Limitation when using ODBC	883
Specifying the data type	883
Configuring DSN for Launcher startup	883
Restrictions and limitations	883
Return codes and error messages	883
Java Class Adapter	883
System requirements	884
Adapter capabilities	884
Java Class Adapter limitations	884
Command alias	884
Java Class Adapter commands	884
Deserialize (-DS)	885
Serialize (-S)	885
Trace (-T)	885
Trace error (-TE)	885
Syntax summary	885
Data sources and targets	885
Using the Java Class Adapter	885
Simple Java class example	886
Type tree elements	886
Adapter action sequence	886
Complex Java class example	887
Return values	887
Return value rules	888

Java Class Importer	888
Configuring JVM options in the config.yaml file	888
Running the Java Class Importer	888
Return codes and error messages	889
Java Class Adapter messages	889
JAXB Adapter	889
System requirements	889
Java class requirements	889
JAXB adapter command overview	890
Command alias	890
Class name (-C) command	891
File location (-U) command	891
Object ID (-I) command	891
Encoding (-jaxb.encoding) command	891
Schema location (-jaxb.schemaLocation) command	891
No namespace schema location (-jaxb.noNamespaceSchemaLocation) command	892
Generate document-level events (-jaxb.fragment) command	892
Create trace file (-T) command	892
Trace adapter errors (-TE) command	892
JDBC Adapter	893
Introduction	893
JDBC adapter command overview	893
Authenticating Connections	894
Adapter properties and commands	894
Examples	899
JDBC adapter -KEY command examples	899
JDBC adapter -BIND command examples	899
JMS Adapter	900
System requirements	901
Messaging modes	901
Point-to-Point (PTP)	901
Publish and subscribe (Pub/Sub)	901
Command Alias	901
JMS Adapter commands	902
ConnectionFactoryName (-CFN)	902
CorrelationID (-CID)	903
DeliveryMode (-DM)	903
DurableSubscriber (-DS)	903
Expiration (-E)	903
GroupID (-GID)	904
Header (-HDR)	904
InitContextFactory (-ICTXF)	904
InitContextFactoryURL (-ICTXFURL)	905
JNDI Method (-JM)	905
Listen (-LSN)	905
Priority (-PRI)	905
Property (-PRP)	906
Password (-PW)	906
Queue (-QN)	906
Quantity (-QTY)	906
Reply (-R)	907
Selector (-SEL)	907
Trace (-T)	907
Text (-TT)	908
Topic (-TN)	908
UserName (-UN)	908
Syntax summary	908
Data sources	909
Example 1	909
Data targets	909
Example 2	909
Using the Adapter	910
Message content	910
Header	910
Header Fields Table	910
Properties	910
Application-specific Properties	910
JMS-specific Properties	911
Vendor-specific Properties	911
Using Message Properties	911
Body or Payload	912
Message	912
Bytes message	912
Stream message	913

MapMessage	913
TextMessage	913
ObjectMessage	913
Message Selection	913
Example Files	913
JMS Importer	913
Configuring JVM Options in the config.yaml File	913
Running the JMS Importer	914
JMS Adapter dependent jars	914
Return Codes and Error Messages	914
Messages	914
JNDI Adapter	914
System requirements	915
Command alias	915
JNDI commands	915
List of commands	915
Authentication Method (-AM)	915
InitContextFactory (-ICTXF)	915
InitContextFactoryURL (-ICTXFURL)	916
Max Hits (-MAXHITS)	916
Timeout (-TIMEOUT)	916
Trace (-T)	916
Syntax summary	917
Data sources	917
Data targets	917
Example 1	917
Example 2	917
Using the adapter	918
Theory of operation	918
Adapter capabilities	918
Adapter limitations	918
Adapter scenarios	918
Return values	919
Schemas	919
JNDI service providers	919
LDAP	920
Operations supported	920
Command line options supported	920
URL format	920
Initial Context factory	921
LDAP schema	921
COS naming	921
Operations supported	922
Command line options supported	922
URL format	922
Initial Context factory	922
Generic naming type tree (COS Naming)	922
RMI	923
Operations supported	923
Command line options supported	923
URL format	923
Initial Context factory	923
Generic naming type tree (RMI)	923
DNS	924
Operations supported	924
Command line options supported	924
URL format	924
Initial Context factory	925
DNS Type Tree	925
File system	925
Operations supported	925
Command line options supported	926
URL format	926
Initial Context factory	926
Generic naming schema (file system)	926
WebLogic	926
Operations supported	927
Command line options supported	927
URL format	927
Initial Context factory	927
Generic naming type tree (WebLogic)	927
Interoperability between JNDI adapter and other adapters	928
Binary data	928

Example files	928
Memory Link Adapter	928
MIME Adapter	929
MIME Adapter overview	929
System requirements	930
Command alias	930
MIME commands	930
List of commands	930
Decode (-DECODE)	930
Encode (-ENCODE)	931
Generate ID (-GENID)	931
Message Digest (-DIGEST)	931
Session (-SESSION)	931
Example	932
Trace (-T)	932
Transport (-TRANSPORT)	932
Syntax summary	932
Data sources	933
Data targets	933
Example	933
Using the adapter	933
MIME structure	933
MIME header fields	934
MIME version header field	934
Content-type header field	934
Content-transfer-encoding header field	934
Content-ID header field	935
Content-description header field	935
Multipart messages	935
Representing MIME data in type trees	935
Message	936
Header	936
Message Parts	937
Using the type tree example	937
MongoDB Adapter	937
Type tree requirements	938
Instance document formats	938
MongoDB adapter commands	939
MSMQ Adapter	940
Overview	940
MSMQ message content	940
System and configuration requirements	940
Command alias	940
GET > Source settings	941
GET > Source > Command setting	941
Source > Transaction > OnSuccess setting	941
Source > Transaction > OnFailure setting	941
Source > Transaction > Scope setting	941
Source > Transaction > Warnings setting	942
PUT > Target settings	942
PUT > Target > Command setting	942
Target > Transaction > OnSuccess setting	942
Target > Transaction > OnFailure setting	942
Target > Transaction > Scope setting	942
Target > Transaction > Warnings setting	943
MSMQ commands	943
MSMQ Adapter commands	943
Wildcard support	943
Correlation ID (-CID)	944
Global Transaction Management (-GTX)	944
Header (-HDR)	944
Label (-LABEL)	944
Listen (-LSN)	945
Queue Name (-QNAME)	945
Quantity (-QTY)	946
Trace (-T)	946
Transactional Queue (-TQ)	947
Syntax summary	947
Data sources	947
Data targets	947
MSMQ maps	948
Map1	948
Results1	948
Change Map1	948

Results2	948
Map2	948
Results1	949
Change Map2	949
Results2	949
Map3	949
Results	950
Map4	950
Results	950
Map5	950
Results1	951
Put messages on the queue	951
Results2	951
Map6	951
Results	952
Map7	952
MSMQ type trees	952
Using the MSMQ type trees	953
MSMQ message header properties	953
MSMQ property variant type and values	953
VT_CAB	953
VT_CLSID	953
VT_LPSTR	954
VT_UI1	954
VT_UI4	954
Message property definitions and values	954
PROPID_M_ACKNOWLEDGE	955
PROPID_M_ADMIN_QUEUE	955
PROPID_M_APPSPECIFIC	955
PROPID_M_BODY	955
PROPID_M_BODY_TYPE	955
PROPID_M_CONNECTOR_TYPE	956
PROPID_M_CORRELATIONID	956
PROPID_M_DELIVERY	956
PROPID_M_DEST_QUEUE	956
PROPID_M_EXTENSION	957
PROPID_M_JOURNAL	957
PROPID_M_LABEL	957
PROPID_M_PRIORITY	957
PROPID_M_RESP_QUEUE	957
PROPID_M_TIME_TO_BE_RECEIVED	957
PROPID_M_TIME_TO_REACH_QUEUE	957
PROPID_M_TRACE	958
PROPID_M_XACT_STATUS_QUEUE	958
Troubleshooting	958
Trace log	958
Error returned to engine (-5)	958
Return codes and error messages	958
Messages	959
Microsoft SQL Server Adapter	959
Overview	959
System requirements	959
DTX_LOCALE_CHARSET	960
DTX_TEMPDB	960
DTX_SQLSVR_CONN_LIMIT	960
Database columns and types	960
Item type properties	960
Date and time formats	961
Database Interface Designer settings	961
Database definition dialog	961
Stored procedures native call syntax	961
Microsoft SQL Server Adapter commands	962
Adapter command summary	962
Adapter-specific command list	962
Database Adapter Type (-DBTYPE)	962
Data Source adapter command (-SOURCE)	962
Trigger adapter command (-TR or -TRIG)	963
Adapter commands for a source	963
GET> Source> Command setting	963
-ID execution command	963
The source in the compiled map is a database	963
The source in the compiled map is not a database	964
DBLOOKUP or DBQUERY functions	964

GET Function	964
Adapter commands for a target	965
PUT> Target> Command setting	965
-OD execution command	965
The target in the compiled map is a database	965
The target in the compiled map is not a database	966
PUT function	966
Extensions for the Launcher	966
Installing Microsoft SQL Server Extensions	967
Adding extended stored procedures	967
Bulk copy example files	967
Restrictions and limitations	967
Return codes and error messages	967
OData Adapter	967
Authenticating connections	967
OData Adapter properties and commands	968
Root URL (-RU or -ROOTURL)	968
Authentication Type (-AT or -AUTHTYPE)	968
OData Version	969
Username	969
Password	969
Entity Set	969
Headers	969
Format	969
Filter	969
Select	969
Top	969
Skip	970
Order By	970
Read Mode	970
Logging	970
Append Log	970
Log File Path	970
GET function example	970
PUT function example	970
ODBC Adapter	971
Overview	971
System requirements	971
Database columns and types	971
Item type properties	971
Date and time formats	972
Database Interface Designer settings	972
Database definition dialog	972
Stored procedures native call syntax	973
ODBC Adapter commands	973
Adapter command summary	973
Adapter specific commands	973
Database Adapter Type (-DBTYPE)	973
Row Count (-ROWCNT)	974
Data Source (-SOURCE)	974
Adapter commands for a source	974
GET> Source> Command setting	974
-ID Execution command	975
The source in the compiled map is a database	975
The source in the compiled map is not a database	975
DBLOOKUP or DBQUERY functions	975
GET function	976
Adapter commands for a target	976
PUT> Target> Command setting	976
-OD execution command	976
The target in the compiled map is a database	977
The target in the compiled map is not a database	977
PUT function	977
Overriding column attributes	978
Defining column attribute overrides	978
Binding values in DBLOOKUP/DBQUERY	979
Limitation when using ODBC	979
Specifying the data type	979
Configuring DSN for Launcher startup	979
Restrictions and limitations	980
Return codes and error messages	980
OLE DB Adapter	980
System requirements	980
Creating a universal data location file	981

Creating a .udl file	981
Database columns and types	981
Item type properties	981
Date and time formats	982
Database interface designer settings	982
Database Designer dialog	982
Stored procedures native call syntax	983
OLE DB Adapter commands	983
Adapter command summary	983
Adapter specific commands	983
Database Adapter Type (-DBTYPE)	983
Row Count (-ROWCNT)	984
Data Source (-SOURCE)	984
Adapter commands for a source	984
GET> Source> Command setting	984
-ID Execution command	984
The source in the compiled map is a database	985
The source in the compiled map is not a database	985
DBLOOKUP or DBQUERY functions	985
GET function	986
Adapter commands for a target	986
PUT> Target> Command setting	986
-OD Execution command	986
The target in the compiled map is a database	986
The target in the compiled map is not a database	987
PUT function	987
Restrictions and limitations	987
Return codes and error messages	988
OpenPGP Adapter	988
Pipes	988
System requirements	988
Command alias	988
OpenPGP Adapter commands	988
List of commands	989
Action (-ACTION)	989
Additional Command (-ADDCMD)	989
Comment (-COMMENT)	990
Compression Algorithm (-COMALG)	990
Decode (-DECODE)	990
Encode (-ENCODE)	990
Encryption Algorithm (-ENCALG)	991
Encryption Type (-ENCTYPE)	991
Message Digest (-DIGEST)	991
Message File Name (-FILE)	991
Input card	992
Output card	992
GET Map function	992
PUT map function	992
Message Recipient (-R)	992
Signature Type (-SIGTYPE)	992
Signing Algorithm (-SIGALG)	993
Trace (-T)	993
User Login (-U)	993
User Password (-PW)	993
Syntax summary	994
Data sources and targets	994
Example 1	994
Example 2	994
Oracle Adapter	994
System requirements	995
Database columns and types	995
Item type properties	995
Date and time format	996
XML type	997
URIType	997
Database Interface Designer settings	997
Database definitions	997
Specifying connect strings	998
Stored procedures native call syntax	998
Oracle client result cache support	998
SQL SELECT query requirements	998
Tuning Oracle adapter database connections	999
Enabling the Oracle adapter to use the Oracle client result cache	999

Oracle Adapter commands	999
Adapter properties and commands	999
Adapter commands for a source	1000
GET > Source > Command setting	1001
Database execution command: input source override (-ID)	1001
DBLOOKUP and DBQUERY functions	1002
GET function	1002
Adapter commands for a target	1003
PUT > Target > Command setting	1003
PUT function	1004
Database execution command: output source override (-OD)	1004
Database triggers using Oracle	1005
Bulk copy example files	1005
Restrictions and limitations	1005
Return codes and error messages	1005
Oracle AQ Adapter	1005
Oracle AQ message content	1006
System requirements	1006
Command alias	1006
GET > Source settings	1007
GET > Source > Command setting	1007
Source > Transaction > OnSuccess setting	1007
Source > Transaction > OnFailure setting	1007
Source > Transaction > Scope setting	1007
Source > Transaction > Warnings setting	1007
PUT > Target settings	1008
PUT > Target > Command setting	1008
Target > Transaction > OnSuccess setting	1008
Target > Transaction > OnFailure setting	1008
Target > Transaction > Scope setting	1008
Target > Transaction > Warnings setting	1008
Oracle AQ Adapter commands	1009
List of commands	1009
Connect string (-C)	1010
Correlation ID (-CID)	1010
Database name (-DBNAME)	1010
Database/query file (-MDQ)	1010
Delay (-D)	1010
Exception queue (-EQ)	1011
Expiration (-E)	1011
Global transaction management (-GTX)	1011
Header (-HDR)	1011
Immediate (-I)	1012
IP address (-IP)	1012
Listen (-LSN)	1012
No data (-NODATA)	1012
Password (-PW)	1012
PORT number (-PORT)	1013
Priority (-P)	1013
Quantity (-QTY)	1013
Queue name (-Q)	1013
Recipient (-R)	1014
Sender (-S)	1014
Trace (-T)	1014
User ID (-US)	1014
Syntax summary	1015
Data sources	1015
Data targets	1015
Oracle AQ Adapter example files	1015
Connection sharing and transactions	1015
Data format	1015
Trace log	1016
Return codes and error messages	1016
Oracle AQ Adapter messages	1016
PDF Adapter	1016
System requirements	1016
PDF Adapter commands	1017
Command aliases	1017
Transaction scope and document persistence in memory	1017
The data_from_map option	1018
Add document (-ADOC) command	1018
Remove document (-RDOC) command	1018
Replace document (-UDOC) command	1018

Get document data (-GDOC) command	1019
Filename (-URL) command	1019
Schema (-SCHEMA) command	1019
Page (-PAGE) command	1019
PDF Template (-TEMPLATE) command	1019
Name lookup file (-NAMES) command	1019
Append data (-APPEND) command	1019
PDF transformation (-PDF) command	1020
RabbitMQ Adapter	1020
Overview	1020
Introduction	1020
Prerequisites	1021
Authenticating connections	1021
RabbitMQ adapter as a source	1021
RabbitMQ adapter as a target	1021
Adapter properties and commands	1022
Design Time Operations	1026
Testing Connections	1026
Listing queues and exchanges	1026
Generating Schema	1026
Examples	1027
Example of the GET map function	1027
Example of the PUT map function	1027
Redis Adapter	1028
Overview	1028
Introduction	1028
Redis adapter as a source	1028
Redis adapter as a target	1028
Supported Redis adapter	1028
Adapter properties and commands	1030
Examples	1031
ServiceNow Adapter	1032
Introduction	1032
Authentication Connections	1032
Adapter Properties and Commands	1032
Instance Name	1033
Authentication	1033
Username	1033
Password	1033
Access Token	1033
Access Token Expiry Action	1034
Consumer Key	1034
Consumer Secret	1034
Refresh Token	1034
Security Token	1034
Token Storage File	1034
Encryption Key	1034
Version	1034
Endpoint	1035
Table Name	1035
System Id	1035
Query Parameter	1035
Logging	1035
Log File Path	1035
Examples	1035
GET examples	1036
PUT examples	1036
REST Adapter	1036
Authenticating connections	1036
REST Adapter properties and commands	1036
Configuration mode	1037
URL	1037
Method	1037
Headers	1037
Script Path	1037
Endpoint	1037
Package Id	1038
Authentication	1038
Access Token	1038
Access Token Expiry Action	1038
Access Token URL	1038
Consumer Key	1038
Consumer Secret	1038
Refresh Token	1038

Security Token	1039
Token Storage File	1039
Encryption Key	1039
Logging	1039
Append Log	1039
Log File Path	1039
SSL/TLS support in REST adapter	1039
Authenticate Server	1039
Truststore file full path	1040
Truststore Password	1040
Authenticate Client	1040
Keystore file path	1040
Keystore Password	1040
Key Password	1040
Verify host name	1041
Pagination support in REST adapter	1041
Pagination Type	1041
Records per Call	1041
Limit	1042
First Offset	1042
Number of Records Location	1042
Number of Records Path	1042
First Page	1042
Number of Pages Location	1042
Number of Pages Path	1042
Next Page Link Location	1042
Next Page Link Path	1043
Base URL	1043
Next Page Token Location	1043
Next Page Token Path	1043
Pagination Token Location	1043
Pagination Token Name	1043
Header and Parameter Special Values	1043
Examples	1044
No Authentication Examples	1044
Basic Authentication examples	1044
OAuth 2.0 authentication examples	1044
Proxy server support	1045
Proxy server support in REST Adapter	1045
Proxy configuration using environment variables	1045
Proxy configuration using flow variables	1045
Proxy configuration using properties	1045
Proxy configuration using the command line	1046
Troubleshooting of proxy server	1046
Salesforce Adapter	1046
Authenticating connections	1046
Salesforce adapter as source	1047
Salesforce adapter as a target	1047
Properties and commands	1047
Instance URL	1047
Access token	1047
Version	1047
Endpoint	1048
Access Token Expiry Action	1048
Consumer key	1048
Consumer secret	1048
Refresh token	1048
Security Token	1048
Token Storage File	1048
Encryption key	1049
Object Name	1049
Id	1049
SOQL Query	1049
Search string	1049
Logging	1049
Append Log	1049
Log File Path	1049
Examples	1049
GET map function	1050
PUT map function	1050
Secure File Transfer Protocol (SFTP) Adapter	1050
SNMP Adapter	1053
System requirements	1054

Command alias	1054
SNMP Adapter commands	1054
List of commands	1054
Context name (-CONTEXTNAME or -N)	1055
Port (-PORT or -P)	1055
Security level (-SECURITYLEVEL or -L)	1055
Trace (-TRACE or -T)	1055
User (-USER or -U)	1055
Version (-VERSION or -V)	1056
XML (-XML)	1056
Example	1056
Example files	1056
Return codes and error messages	1056
SNMP Adapter messages	1056
SOAP (WSDL) Adapter	1057
System requirements	1057
Command alias	1057
SOAP Adapter commands	1057
List of commands	1057
Decode (-DECODE)	1058
Encode (-ENCODE)	1058
Header (-HDR)	1058
Raw (-RAW)	1058
Return (-RETURN)	1059
Return option and XPath	1059
SOAPAction (-SA)	1060
Trace (-T)	1060
Transport (-TRANSPORT)	1061
Syntax summary	1061
Data sources	1061
Example	1061
Data targets	1061
Example	1062
Source options behavior	1062
Using the adapter	1062
Consumer scenario	1062
Provider scenario	1063
Response	1063
Example files	1063
Return codes and error messages	1063
Messages	1063
Socket Adapter	1064
System requirements	1064
How the adapter works	1064
Client or server mode	1064
Conversation support	1064
Command alias	1065
Socket Adapter commands	1065
List of commands	1065
Client Connection (-CCON)	1066
Data header (-DHDR)	1066
End Of File (-EOF)	1066
End Of Message (-EOM)	1067
Fixed (-FIXED)	1067
Host (-H or -HOST)	1067
Inclusive sized (-ISIZED)	1067
Listen (-LSN)	1068
Mode (-M or -MODE)	1068
Number of event messages (-NEM)	1068
Port (-P or -PORT)	1068
Quantity (-QTY)	1068
Service (-S or -SERVICE)	1069
Sized (-SIZED)	1069
Trace (-T or -TRACE)	1069
Window Size (-WNDSIZE)	1069
Syntax summary	1069
Data sources	1069
Data targets	1070
Examples	1070
Troubleshooting	1070
Standards Processing Engine Adapter	1070
System requirements	1070
SPE adapter commands and data harness functions	1070
Data harness functions	1071

Data harness function syntax	1071
SPE adapter action command overview	1073
Command aliases	1074
The SPE database configuration file	1074
Database connection persistence	1075
The data_from_map option	1075
Add document (-ADOC) command	1075
Get document data (-GDOC) command	1075
Remove document (-RDOC) command	1075
Replace document (-UDOC) command	1076
Run a map (-TRANSFORM) command	1076
De-envelope a document (-DEENVELOPE) command	1077
Envelope a document (-ENVELOPE) command	1078
Archive (Tar) Adapter	1080
Overview	1080
System requirements	1080
Command alias	1080
Archive (Tar) Adapter commands	1081
List of commands	1081
Tar File Archive (-ARCHIVE)	1081
Archive File (-FILE)	1081
Group ID (-GID)	1082
Tar File Member (-MEMBER)	1082
File Permissions (-PERM)	1082
Trace (-T)	1082
User ID (-UID)	1083
Syntax summary	1083
Data sources	1083
Data targets	1083
Examples	1083
Troubleshooting	1084
Adapter trace file	1084
Sample adapter trace file	1084
Return codes and error messages	1084
Messages	1084
TIBCO Rendezvous Adapter	1086
System Requirements	1086
TIBCO Rendezvous Message Content	1086
Simple Messages	1087
Complex messages	1087
Data format for TIBCO Rendezvous messages	1087
Binary Datum item types	1087
Tuple Array of Binary Datum Item Types	1088
TIBCO Rendezvous	1088
High Data Rates	1088
Distributed queuing	1088
TIBCO Rendezvous Command Aliases	1088
Wildcard support	1089
Wildcard usage with the Launcher	1089
Using wildcards to generate unique message IDs	1089
GET▶ Source Settings	1089
GET▶ Source▶ Command Setting	1090
Source▶ Transaction▶ OnSuccess Setting	1090
Source▶ Transaction▶ OnFailure Setting	1090
Source▶ Transaction▶ Scope Setting	1090
Source▶ Transaction▶ Warnings Setting	1090
PUT▶ Target Settings	1090
PUT▶ Target▶ Command Setting	1091
Target▶ Transaction▶ OnSuccess Setting	1091
Target▶ Transaction▶ OnFailure Setting	1091
Target▶ Transaction▶ Scope Setting	1091
Target▶ Transaction▶ Warnings Setting	1091
TIBCO Rendezvous Adapter commands	1091
Certified Session Name (-CMN)	1092
Confirmation (-CONFIRM)	1093
Daemon (-DAEMON)	1093
Distributed Queue (-DQ)	1093
Error Subject Name (-ERN)	1093
Header (-HDR)	1094
Listener Certified Session Name (-LCMN)	1094
Ledger File Name (-LFN)	1094
Listen (-LSN)	1095
Message (-MSG)	1095

Network (-NETWORK)	1095
No Old Messages (-NOM)	1095
Opaque Message Format (-OPAQUE)	1096
Point-to-Point (-PP)	1096
Queue Options (-QO)	1096
Quantity (-QTY)	1097
Reply (-REPLY)	1097
Request Listen (-RQSTLSN)	1098
Request Reply (-RQSTRPL)	1098
Subject Name (-SBN)	1098
Service (-SERVICE)	1099
Trace (-T)	1099
Syntax summaries	1099
Data Sources	1099
Data Targets	1099
Example Files	1100
Type Tree Example	1100
Map Example	1100
Troubleshooting	1100
Trace Log	1100
Advisory messages	1100
Return Codes and Error Messages	1100
Messages	1101
Archive (Zip) Adapter	1101
Overview	1101
System requirements	1101
Command alias	1102
Archive (Zip) Adapter commands	1102
List of commands	1102
Archive File (FILE)	1102
Archive Data (MEMORY)	1102
Archive Restore	1103
Syntax summary	1103
Data sources	1103
Data targets	1103
Examples	1103
Return codes and error messages	1104
Messages	1104

Resource Registry

Resource Registry overview	1104
Resource Registry files	1105
Resource Registry interface overview	1105
Resource Registry navigator overview	1105
Resource Registry menu commands	1105
Resource Registry map deployment overview	1106
Creating a Resource Registry resource name alias	1106
Access to the Resource Registry	1106
Resource name files	1106
Creating Resource Registry name files	1106
Defining Resource Registry names	1107
Defining resource names with environment variables	1107
Defining Resource Registry virtual servers	1107
Defining Resource Registry values	1107
Implementing Resource Registry resource names	1108
Resource Registry alias overview	1108
Resource Registry alias usage overview	1108
Resource Registry alias expansion in external interface functions	1109
Resource Registry aliases in audit and debug files	1109
Resource Registry trace file usage overview	1109
Resource Registry encrypted alias usage with masked command lines	1109
Resource Registry encryption overview	1110
Resource Registry name files and encryption keys	1110
Resource Registry master key file and encrypted value display	1111
Encrypting and resetting Resource Registry values	1111
Encrypting a Resource Registry value and generating a master key file	1111
Converting a Resource Registry name file to AES encryption	1111
Resetting encrypted Resource Registry values	1112
Regenerating a Resource Registry master key file	1112
Resource configuration files overview	1112
Creating resource configuration files	1112
Creating a Launcher resource configuration file	1113
Creating a Command Server resource configuration file	1113
Transformation Extender Programming Interface and Java API resource configuration files	1113
Saving resource configuration files	1114

Resource configuration files for map servers	1114
Map Designer resource configuration file	1114
Launcher resource configuration file	1114
Transformation Extender Programming Interface resource configuration file	1114
Java API resource configuration file	1114
Initialization resource configuration file	1115
Execution Commands	
Execution maps	1115
Command Server overview	1115
Coordinating execution commands with compiled settings	1115
Map settings	1115
Card settings	1116
Compiled map files	1116
Map Designer	1116
Integration Flow Designer	1116
Command files	1117
Using Command Server commands	1117
General rules for execution commands	1117
From a command line	1117
Command line help	1118
From the Platform API	1118
From a RUN() function in a map rule	1118
Execution commands	1118
Command availability	1119
MapAudit (-A)	1119
Examples using MapAudit (-A)	1121
Close Window (-B)	1121
Century (-D)	1121
Fail on Warnings (-F)	1122
Examples using Fail on Warnings (-F)	1122
Ignore Validation Options (-G)	1122
Input Source Override - Application (-IA)	1123
Input Source Override - Database (-ID)	1124
Input Source Override - Echo (-IE)	1125
Input Source Override - File (-IF)	1127
Input Source Override - Message (-IM)	1128
Input Source Override - Pointer (-IP)	1129
List (-L)	1130
Initialization File Override (-N)	1130
Output Target Override - Application (-OA)	1130
Output Target Override - Database (-OD)	1131
Output Target Override - Echo (-OE)	1132
Output Target Override - File (-OF)	1133
Output Target Override - Message (-OM)	1134
Output Target Override - Pointer (-OP)	1135
WorkSpace PageSize (-P)	1135
Refresh Status (-R)	1136
MapTrace (-T)	1136
Examples using MapTrace (-T)	1137
Stop Validation (-V)	1137
WorkSpace (-W)	1137
Retry (-Y)	1138
Ignore Warnings (-Z)	1138
Examples using Ignore Warnings (-Z)	1138
Examples	1139
Running a map located in a different directory	1139
Creating work files in an alternate directory	1139
Executing multiple maps from a single command	1140
Overriding a database target	1140
Overriding the schema location at run time	1140
Executing a command file	1141
Using a run function and echoed outputs in memory	1141
Audit Log in memory	1141
INI commands for debugging maps	1141
Logging	1142
Logging categories	1142
Debug	1143
DebugName	1143
DebugAppend	1143
Functions and Expressions	
Expressions and evaluations	1143
Expressions	1144
Component rule expressions evaluate to true or false	1144
Map rule expressions evaluate to data	1144
Literals	1144

Data object names	1145
Object names in map rules	1145
Object names in component rules	1145
Card name	1145
Local type name	1145
Partition list	1146
Component path	1146
Indexed object names	1146
Using LAST in a map rule to reference an input	1146
Using LAST in a component rule	1146
Using LAST in a map rule to reference an output	1146
Component paths separated by a colon	1147
Component paths separated by IN	1147
Referring to all occurrences with IN COMPONENT	1147
Comment object name	1147
Shorthand notation	1147
Using the dollar sign (\$)	1148
Using ellipses	1148
Evaluating expressions	1148
Card order can influence the order of evaluation sets	1148
Functions influence the number of evaluation sets	1149
Object names influence the number of evaluation sets	1150
Using object names to coordinate evaluation sets	1150
Using IN to decouple object name coordination	1150
Using IN to decouple different partitions in the same rule	1150
Operators	1151
Arithmetic operators	1151
Text operators	1151
Logical operators	1151
Comparison operators	1151
Order of operator evaluation	1152
Operands	1152
Using functions in expressions	1152
Function arguments	1152
Input arguments	1153
Nested input arguments	1153
Output arguments	1153
Function arguments and evaluation	1153
Map names in expressions	1154
Evaluation of functional maps in an expression	1154
Expressions that evaluate to NONE	1155
When an operand evaluates to NONE	1155
When an input argument of a function evaluates to NONE	1155
When an input argument of a functional map evaluates to NONE	1155
When an input of an executable map evaluates to NONE	1156
Impact of track setting on order of output	1156
Evaluated expressions assigned to output items	1156
Automatic item format conversions	1156
Numeric precision	1157
Using functions	1157
Functions in a component rule	1157
Functions in a map rule	1157
Syntax of a function	1157
Function argument syntax	1157
Functions that convert character-set encoding to the native code page	1158
General functions	1158
ASFUNCTION	1159
CLONE	1160
DEFAULT	1160
ECHOIN	1160
HANDLEIN	1161
PARSE	1161
REFORMAT	1163
RESOLVETYPE	1163
Bidirectional functions	1164
SETLOGICALORDER	1164
SETORIENTATIONLTR	1165
SETORIENTATIONRTL	1165
SETTEXTSHAPING	1165
SETTEXTSHAPINGOFF	1165
SETVISUALORDER	1165
Bit manipulation and testing functions	1166
SETOFF	1166
SETON	1166
TESTOFF	1166

TESTON	1167
Cache functions	1167
CACHEDELETE	1167
CACHEREAD	1168
CACHEWRITE	1168
Conversion functions	1168
BASE64TOTEXT	1169
BCDTOHEX	1169
BCDTOINT	1170
BCDTOTEXT	1170
CONVERT	1171
DATETONUMBER	1171
DATETOTEXT	1172
FROMBASSETN	1172
FROMDATETIME	1172
FROMNUMBER	1173
HEXTXTTOSTREAM	1173
INT	1174
NUMBERTODATE	1174
NUMBERTOTEXT	1175
PACK	1175
PACKAGE	1176
QUOTEDTOTEXT	1176
SERIESTOTEXT	1177
STREAMTOHEXTXT	1177
SYMBOL	1178
TEXTTOBASE64	1178
TEXTTOBCD	1179
TEXTTODATE	1179
TEXTTONUMBER	1180
TEXTTOQUOTED	1180
TEXTTOTIME	1181
TIMETOTEXT	1181
TOBASSETN	1182
TODATETIME	1182
TONUMBER	1183
UNPACK	1183
UNZONE	1184
ZONE	1184
Date/time functions	1185
ADDDAYS	1186
ADDHOURS	1186
ADDMINUTES	1187
CURRENTDATE	1187
CURRENTDATETIME	1188
CURRENTTIME	1188
DATETONUMBER	1171
DATETOTEXT	1172
FROMDATETIME	1172
MAX	1190
MIN	1191
NUMBERTODATE	1174
TEXTTODATE	1179
TEXTTOTIME	1181
TIMETOTEXT	1181
TODATETIME	1182
Error handling functions	1194
CONTAINERRORS	1194
FAIL	1194
GETXMLERRORMSG	1195
ISERROR	1195
ONERROR	1196
REJECT	1196
QUIT	1197
VALID	1197
External interface functions	1198
DBLOOKUP	1198
DBQUERY	1200
DDEQUERY	1201
EXIT	1202
Implementing a library EXIT function	1203
EXIT function's library interface	1203
Using the EXITPARAM Structure	1203
GET	1204
JEXIT	1205
JEXIT Examples	1206
PUT	1207
RUN	1208
RENAMEFILE	1209

HARDLINKFILE	1209
SOFTLINKFILE	1209
COPYFILE	1209
SLEEP	1210
Flow functions	1210
DECVARIABLE	1210
DELETEVARIABLE	1211
GETVARIABLE	1211
INCVARIABLE	1211
RESOLVEVARIABLE	1212
SETVARIABLE	1212
Inspection functions	1212
ABSENT	1213
CONTAINERRORS	1194
DELETEFILE	1213
ISALPHA	1214
ISERROR	1195
ISLOWER	1215
ISNUMBER	1215
ISUPPER	1216
MEMBER	1216
NOT	1217
OFFSET	1217
OR	1217
PARTITION	1218
PRESENT	1218
SIZE	1219
TESTOFF	1166
TESTON	1167
VALID	1197
Logical functions	1220
ALL	1220
BINDABS	1221
EITHER	1221
IF	1222
ISALPHA	1214
ISLOWER	1215
ISNUMBER	1215
ISUPPER	1216
NOT	1217
OR	1217
WHEN	1225
Lookup and reference functions	1225
CHOOSE	1226
DBLOOKUP	1198
DBQUERY	1200
DDEQUERY	1201
EXTRACT	1230
GETANDSET	1230
GETDIRECTORY	1231
GETITXUID	1231
GETLOCALE	1232
GETFILENAME	1232
GETPARTITIONNAME	1232
GETRESOURCEALIAS	1233
GETRESOURCENAME	1233
GETTXINSTALLEDIRECTORY	1234
GETDATADIRECTORY	1234
INDEX	1234
INDEXABS	1235
LASTERRORCODE	1235
LASTERRORMSG	1236
LOOKDOWN	1236
LOOKUP	1237
MEMBER	1216
SEARCHDOWN	1238
SEARCHUP	1238
SORTDOWN	1239
SORTDOWNBY	1240
SORTUP	1240
SORTUPBY	1241
UNIQUE	1241
GETENV	1242
GETOSNAME	1242
Math and statistics functions	1242
ABS	1243
ACOSINE	1243
ASIN	1244
ATAN	1244
ATAN2	1244

COSINE	1244
COSINEH	1244
COUNT	1244
COUNTABS	1245
EXP	1245
FACTORIAL	1245
FROMBASESETEN	1172
INT	1174
LOG	1246
LOG10	1247
MAX	1190
MIN	1191
MOD	1247
POWER	1248
RAND	1248
ROUND	1248
SEED	1249
SIN	1249
SINH	1249
SQRT	1249
SUM	1250
TAN	1250
TANH	1250
TOBASESETEN	1182
TRUNCATE	1251
Text functions	1251
BCDTOTEXT	1170
COUNTSTRING	1252
CPACKAGE	1253
CSERIESTOTEXT	1253
CSIZE	1254
CTEXT	1254
DATETOTEXT	1172
FILLEFT	1255
FILLRIGHT	1255
FIND	1256
HEXTEXTTOSTREAM	1173
LEAVEALPHA	1257
LEAVEALPHANUM	1257
LEAVENUM	1258
LEAVEPRINT	1258
LEFT	1259
LOWERCASE	1259
MAX	1190
MID	1260
MIN	1191
NORMXML	1261
NUMBERTOTEXT	1175
PACKAGE	1176
PROPERCASE	1262
REGEXMATCH	1263
REGEXPARSE	1263
REGEXPOSITION	1263
REGEXREPLACE	1264
REGEXREPLACEALL	1264
RANDDATA	1264
REVERSEBYTE	1265
RIGHT	1265
SEED	1249
SERIESTOTEXT	1177
SQUEEZE	1266
SUBSTITUTE	1267
TEXT	1267
TEXTTOBCD	1179
TEXTTONUMBER	1180
TEXTTOTIME	1181
TIMETOTEXT	1181
TODATETIME	1182
TONUMBER	1183
TRIMLEFT	1271
TRIMRIGHT	1272
UPPERCASE	1272
WORD	1273
XML functions	1273
VALIDATE	1274
VALIDATEEX	1274
XVALIDATE	1274
XPATH functions	1275
XPATH	1275

XPATH used with XMLLIB	1275
XPATHEX	1275
XSLT functions	1276
XSLT	1276
XSLT used with XMLLIB	1276
XSLTEX	1276
Custom functions	1277
Creating a custom function	1277
Date and time format strings	1278
Time units	1278
Date units	1279
Binary date and time format strings	1279
Japanese date and time format strings	1279
Japanese date format strings	1280
Japanese time format strings	1280
Western date and time format strings	1281
Western date format strings	1281
Western time format strings	1281
Number format strings	1282
Leading sign format strings	1283
Trailing sign format strings	1283
Substring format strings	1283
Whole number and fraction format strings	1283
RUN function return codes	1283
Character set codes for CPACKAGE, CSERIESTOTEXT, and CTEXT	1285
Sterling B2B Integrator	
Sterling B2B Integrator in Design Server	1289
Sterling B2B Integrator in Design Server overview	1289
Supported Transformation Extender map functionality on Sterling B2B Integrator	1289
Differences when running Transformation Extender maps on Sterling B2B Integrator	1289
Well-behaved map	1290
Data harness	1290
Getting started with Sterling B2B Integrator	1291
Adding a Sterling B2B Integrator server	1291
Configuring an HTTPS connection to the Sterling B2B Integrator server	1291
Configuring the Check In Map service options	1291
Configuring troubleshooting options	1292
Building Transformation Extender map	1292
Deploying Transformation Extender maps	1292
Making the checked-in map the default	1292
Sterling B2B Integrator in the Design Studio	1293
Sterling B2B Integrator in Design Studio overview	1293
Supported Transformation Extender map functionality on Sterling B2B Integrator	1293
Differences when running Transformation Extender maps on Sterling B2B Integrator	1293
Well-behaved map	1294
Data harness	1294
Getting started with Sterling B2B Integrator	1295
Configuring Design Studio for Sterling B2B Integrator	1295
Configuring the connection properties	1295
Configuring an HTTPS connection to the Sterling B2B Integrator server	1296
Configuring the Check In Map service options	1296
Configuring troubleshooting options	1296
Building Transformation Extender maps	1297
Testing Transformation Extender maps	1297
Run operation details	1297
Result details from test running maps	1298
Deploying Transformation Extender maps	1298
Making the checked-in map the default	1299
Deploy operation details	1299
Result details from deploying maps	1299
Troubleshooting Transformation Extender maps	1299
Troubleshooting for the Test Map service	1300
Sterling B2B Integrator session log file	1300
Sterling B2B Integrator audit log file	1300
Troubleshooting for the Check In Map service	1300
Troubleshooting for the Test Map service	1300
IBM DataPower SOA Appliances	1301
WebSphere DataPower SOA Appliances	1301
Limitations of DataPower SOA appliances	1301
Project settings for DataPower maps	1303
Configuring your project to create DataPower maps by default	1303
Configuring your project to update the MapRuntime setting in all of the executable maps	1304
Configuring for WebSphere DataPower SOA Appliance services	1304
Configuring for the Interoperability Test Service	1304
Configuring an HTTPS connection	1304

Save Messages options	1305
Appliance Configuration options	1305
Working Directory option	1305
Retry map setting	1306
Configuring for the XML Management Interface service	1306
Setting Custom File name	1306
Generating WebSphere DataPower maps	1306
Building a WebSphere DataPower map	1307
Testing DataPower maps	1307
Running on a WebSphere DataPower SOA Appliance	1307
Running locally	1307
Deploying WebSphere DataPower maps	1308
Setting password for XML Management Interface service	1309
Overwriting a deployed file	1309
Troubleshooting DataPower maps	1309
Troubleshooting for the Interoperability Test Service	1310
DataPower session log file	1310
Viewing the DataPower session log file	1310
DataPower audit log file	1310
Troubleshooting for the XML Management Interface service	1310
Standards Processing Engine	
Standards Processing Engine	1311
Configuring Map Designer for SPE	1311
Creating a project and importing maps	1311
SPE database connection	1311
Configuring SPE database connections	1312
SPE database deployment options	1312
SPE test options	1313
Configuring default input and output files	1313
Setting Sterling B2B Integrator translator properties	1314
SPE Logging options	1314
Testing a map on SPE and displaying results	1314
Deploying a map to SPE	1314
Global Transaction Management	
Global transaction management	1315
Components	1315
Client application	1315
Transaction manager	1315
Transaction specification types	1315
Supported Third-party Transaction Managers	1316
Supported adapters	1316
Resource managers	1316
Configuration requirements	1316
Installing and running transaction managers	1317
TransManager setting	1317
Input and output card settings	1317
Global transactions in multiple threads	1317
Examples	1318
Flow example	1318
Legend for the example	1318
Trace log file examples	1319
MQS TM trace file example	1319
MSDTC TM trace file example	1319
Performance Recommendations	
Performance overview	1320
General performance overview	1320
Performance measurement factors	1320
Load and performance measurements	1321
Throughput and CPU measurements	1321
Command Server operation overview	1322
Command Server execution overview	1322
Command Server initialization	1322
Command Server validation	1322
Command Server transformation	1322
Command Server finalization steps	1323
Command Server execution time	1323
Command Server work and data files	1323
Reusing work files	1323
Command Server paging settings	1324
Paging and I/O usage	1324
Scenario: configure paging by input file size	1325
Configuring optimal page subsystem settings	1325

WorkSpace in Memory option	1326
Launcher operation overview	1326
Launcher and Command Server performance comparison	1326
Map performance and tuning best practices	1326
Data search performance best practices	1327
Data search functions	1327
Search functions reference	1328
Routing	1328
Data routing usage	1328
Comparisons between the routing functions	1329
Map Profiler (dtxprof)	1329
Using the Map Profiler	1330
Page Setting Assistant for Maps (dtxpage)	1330
Merging and Pruning Schemas and Type Trees	1330
Prune transactions sets from a schema or type tree	1330
Merge pruned schema or type tree to destination schema or type tree	1330
Mainframe performance	1331
Overview	1331
Understanding mainframe performance	1331
Hardware expectations and MIPS/MSU ratings	1332
Performance case	1332
Performance case after normalizing	1332
File formats	1333
Record separators (/V) execution command	1333
Pre-allocating work files (temporary data sets)	1333
Batch	1333
CICS	1333
Paging in memory	1333
WorkSpace Work Files in Memory	1334
WorkSpace Work Files in Memory - Hiperspace	1334
External factors	1334
Batch execution environment	1334
IBM Language Environment (LE) runtimes	1334
Performance considerations - adjusting LE runtime options	1335
Input Data Sets (Source)	1335
Output data sets (Target)	1335
Temporary data sets (work files)	1336
Execution command settings	1336
WorkSpace Pagesize (Paging) (-Psize:count) or (/Psize:count)	1336
WorkSpace (Work Files in Memory) (-WM) or (/WM)	1336
WorkSpace (Work Files in Memory - Hiperspace) (-WH) or (/WH)	1336
Striped data sets	1337
XPLINK performance build	1337
CICS execution environment	1337
IBM Language Environment (LE) runtimes	1337
Temporary data sets (work files)	1337
Enqueuing on data files (-Q)	1338
Execution command settings	1338
WorkSpace Pagesize (Paging) (-Psize:count) or (/Psize:count)	1338
WorkSpace (Work Files in Memory) (-WM) or (/WM)	1338
Striped data sets	1337
CICS shared data tables	1338
Troubleshooting	1338
Batch troubleshooting	1338
Excessive EXCPs (I/Os)	1339
Excessive CPU time used	1339
External factors	1339
CICS troubleshooting	1340
Excessive CICS I/Os, transaction CPU time	1340
External factors	1340
Glossary	1340

Utility Commands

Introduction	1341
Utility commands for troubleshooting	1342
Utility Commands for schema designer	1342
tanalyze utility command	1342
Syntax summary for tanalyze	1343
Utility command options for tanalyze	1343
-L tanalyze option	1343
-S tanalyze option	1343
-R tanalyze option	1343
-LOG tanalyze option	1344

-FAIL tanalyze option	1344
-SAVE tanalyze option	1344
-VERBOSE tanalyze option	1344
-APPEND tanalyze option	1344
-NOLOG tanalyze option	1344
Command line help for tanalyze	1344
Using the tanalyze command	1345
timport utility command	1345
Syntax summary for timport	1345
Utility command options for timport	1345
-APPEND timport option	1346
-BYTEORDER timport option	1346
-CHARSET timport option	1346
-CICS timport option	1346
-FAIL timport option	1346
-HELP timport option	1346
-HINTS timport option	1346
-IMP timport option	1347
-KEEPMTS timport option	1347
-LANG timport option	1347
-LOG timport option	1347
-ND timport option	1347
-NO timport option	1347
-NOLOG timport option	1347
-O timport option	1348
-OPT timport option	1348
-VALIDATION timport option	1348
-VERBOSE timport option	1348
IMPORTER OPTIONS	1348
Command line help for timport	1349
Using the timport command	1349
Import a schema file example	1349
Import a copybook schema example	1350
Import a copybook schema with ND option example	1350
texport utility command	1350
Syntax summary for texport	1350
Utility command options for texport	1350
-T texport option	1351
-O texport option	1351
-NO texport option	1351
-LOG texport option	1351
-FAIL texport option	1351
-APPEND texport option	1352
-VERBOSE texport option	1352
-NOLOG texport option	1352
Command line help for texport	1352
Using the texport command	1352
Example 1	1352
Example 2	1352
dtxmlconv Utility command	1353
Syntax summary for dtxmlconv	1353
Utility command options for dtxmlconv	1353
-V dtxmlconv option	1353
-L dtxmlconv option	1354
-S dtxmlconv option	1354
-B dtxmlconv option	1354
-O dtxmlconv option	1354
-LOG dtxmlconv option	1354
-FAIL dtxmlconv option	1354
-APPEND dtxmlconv option	1354
-VERBOSE dtxmlconv option	1354
-NOLOG dtxmlconv option	1355
Command line help for dtxmlconv	1355
Using the dtxmlconv command	1355
dtxmlconv example	1355
tbccconv utility command	1355
Syntax summary for tbccconv	1356
Utility command options for tbccconv	1356
-APPEND tbccconv option	1356
-FAIL tbccconv option	1356
-LOG tbccconv option	1357
-NOLOG tbccconv option	1357

-P tbconv option	1357
-R tbconv option	1357
-VERBOSE tbconv option	1357
Command line help for tbconv	1357
Using the tbconv command	1358
tbconv default behavior (without -P option)	1358
tbconv default behavior examples	1358
tbconv example 1 (no constraints)	1359
tbconv example 2 (Min Bytes constraints)	1359
tbconv example 3 (Max Bytes constraints)	1359
tbconv example 4 (Min and Max Bytes constraints)	1360
tbconv default syntax example	1360
tbconv behavior (with -P option)	1360
tbconv example with -P option	1360
tbconv example syntax with -P option	1361
Utility commands for map designer	1361
mcompile utility command	1361
mcompile utility command syntax summary	1361
mcompile utility command on Linux platform	1362
mcompile utility command options	1362
-L mcompile option	1363
-A mcompile option	1363
-K mcompile option	1364
-E mcompile option	1364
-P mcompile option	1364
-O mcompile option	1364
-NO mcompile option	1365
-DP mcompile option	1365
-TX mcompile option	1365
-R mcompile option	1365
-LOG mcompile option	1366
-FAIL mcompile option	1366
-APPEND mcompile option	1366
-VERBOSE mcompile option	1366
-NOLOG mcompile option	1366
mcompile command line help	1367
mcompile command usage	1367
Scenario: run mcompile to display all executable maps	1367
Scenario: run mcompile with -DP and -TX options	1367
Scenario: run mcompile to compile all executable maps	1368
Scenario: run mcompile with -DP and -TX options	1368
Scenario: run mcompile to compile an executable map	1369
mimport utility command	1369
mimport syntax summary	1369
mimport utility command options	1370
-O mimport option	1370
-NO mimport option	1370
-LOG mimport option	1370
-FAIL mimport option	1371
-APPEND mimport option	1371
-VERBOSE mimport option	1371
-NOLOG mimport option	1371
mimport command line help	1371
mimport command usage	1371
mexport utility command	1372
mexport syntax summary	1372
mexport utility command options	1372
-O mexport option	1373
-NO mexport option	1373
-LOG mexport option	1373
-FAIL mexport option	1373
-APPEND mexport option	1373
-VERBOSE mexport option	1374
-NOLOG mexport option	1374
mexport command line help	1374
mexport command usage	1374
mreport utility command	1374
mreport syntax summary	1374
mreport utility command options	1375
-O mreport option	1375
-NO mreport option	1375
-LOG mreport option	1375

-FAIL mreport option	1376
-VERBOSE mreport option	1376
-APPEND mreport option	1376
-NOLOG mreport option	1376
mreport command line help	1376
mreport command usage	1376
Utility commands for Integration Flow Designer	1377
sdeploy utility command	1377
sdeploy syntax summary	1377
sdeploy analyze syntax option	1378
sdeploy buildmaps syntax options	1378
sdeploy generate syntax option	1378
sdeploy deploy syntax options	1378
sdeploy utility command options	1378
-L sdeploy option	1379
-A sdeploy option	1379
-K sdeploy option	1379
-EMODE sdeploy option	1380
-ANALYZE sdeploy option	1380
-BUILDMAPS sdeploy option	1380
-GENERATE sdeploy option	1380
-DEPLOY sdeploy option	1380
-MRC sdeploy option	1381
-SERVER sdeploy option	1381
-USER -PW -IP -sdeploy options	1381
-P sdeploy option	1381
-O sdeploy option	1381
-NO sdeploy option	1382
-R sdeploy option	1382
-LOG sdeploy option	1382
-FAIL sdeploy option	1383
-APPEND sdeploy option	1383
-VERBOSE sdeploy option	1383
-NOLOG sdeploy option	1383
sdeploy command line help	1383
sdeploy command usage	1383
Using sdeploy to display all systems on the console	1384
Using sdeploy to analyze all systems	1384
Using sdeploy to build maps from the command line - example 1	1384
Using sdeploy to generate systems from the command line - example 2	1384
Using sdeploy to deploy scripts from the command line - example 3	1385
Using sdeploy to deploy scripts from the command line - example 4	1385
ssftpdeploy utility command	1385
msdimport utility command	1387
msdimport syntax summary	1387
msdimport utility command options	1387
-APPEND msdimport option	1388
-ASVR msdimport option	1388
-ASYS msdimport option	1388
-FAIL msdimport option	1388
-LOG msdimport option	1388
-NO msdimport option	1389
-NOLOG msdimport option	1389
-O msdimport option	1389
-OSVR msdimport option	1389
-OSYS msdimport option	1389
-REFSVR msdimport option	1389
-REFSYS msdimport option	1389
-VERBOSE msdimport option	1389
msdimport command line help	1390
msdimport command usage	1390
msdexport utility command	1390
msdexport syntax summary	1390
msdexport utility command options	1391
-APPEND msdexport option	1391
-ASVR msdexport option	1391
-ASYS msdexport option	1391
-EPWD msdexport option	1392
-FAIL msdexport option	1392
-LOG msdexport option	1392
-NO msdexport option	1392
-NOLOG msdexport option	1392

-O msdexport option	1392
-REFSVR msdexport option	1393
-REFSYS msdexport option	1393
-VERBOSE msdexport option	1393
msdexport command line help	1393
Using the msdexport command	1393
Utility commands for map tuning	1393
dtxpath utility command	1394
dtxpath syntax summary	1394
dtxpath command line help	1394
dtxpath command usage	1394
dtxpath command usage with the z/OS Batch Command Server	1395
dtxprof utility command	1395
Syntax summary for dtxprof	1395
Utility command options for dtxprof	1395
-f dtxprof option	1396
-fx dtxprof option	1396
-t dtxprof option	1396
-tx dtxprof option	1396
-fs dtxprof option	1396
-ts dtxprof option	1396
-d dtxprof option	1396
-o dtxprof option	1396
-dtx dtxprof option	1396
Command line help for dtxprof	1397
Using the dtxprof command	1397
Type names	1397
Function Times	1397
RUN maps	1397
Output examples	1397
For best results	1399
z/OS Batch Command Server dtxprof usage	1399
Utility command for dtxany2xml	1399
dtxany2xml syntax summary	1400
dtxany2xml command line help	1400
dtxany2xml command usage	1400
Scenario: dtxany2xml usage	1400
Utility command for map trace-file conversion	1401
Utility command for Resource Registry	1401
Resource Registry syntax summary	1401
Resource registry command usage	1402

TX Programming Interface APIs

TX programming interface introduction	1403
TX programming interface overview	1403
TX programming interface methods	1404
GetAllPropertiesXML	1404
GetBinaryProperty	1404
GetCountProperty	1404
GetIntegerProperty	1404
GetTextProperty	1404
GetPropertiesXML	1404
SetBinaryProperty	1404
SetIntegerProperty	1405
SetTextProperty	1405
SetPropertiesXML	1405
TX programming interface object overview	1405
TX programming interface object hierarchy	1405
TX programming interface properties	1405
TX programming interface map object	1406
Map object properties	1406
General map properties	1406
MPIP_MAP_DESTROY_OBJECT_POOL	1406
MPIP_MAP_INSTANCE	1407
MPIP_MAP_MAP_NAME	1407
MPIP_OBJECT_ERROR_CODE	1407
MPIP_OBJECT_ERROR_MSG	1407
Map trace properties	1407
MPIP_MAP_TRACE_DIRECTORY	1407
MPIP_MAP_TRACE_DIRECTORY_CUSTOM_VALUE	1408
MPIP_MAP_TRACE_FILENAME	1408
MPIP_MAP_TRACE_FILENAME_CUSTOM_VALUE	1408
MPIP_MAP_TRACE_FORMAT	1408
MPIP_MAP_TRACE_INPUT_CARD_END	1408

MPIP_MAP_TRACE_INPUT_CARD_NUMBER	1408
MPIP_MAP_TRACE_INPUT_CARD_START	1408
MPIP_MAP_TRACE_INPUT_CONTENT	1409
MPIP_MAP_TRACE_LOCATION	1409
MPIP_MAP_TRACE_RULES_CARD_NUMBER	1409
MPIP_MAP_TRACE_RULES_CONTENT	1409
MPIP_MAP_TRACE_SUMMARY_CONTENT	1409
MPIP_MAP_TRACE_SWITCH	1409
Map workspace properties	1409
MPIP_MAP_WORKSPACE_DIRECTORY	1410
MPIP_MAP_WORKSPACE_DIRECTORY_CUSTOM	1410
MPIP_MAP_WORKSPACE_FILE_ACTION	1410
MPIP_MAP_WORKSPACE_FILE_PREFIX	1410
MPIP_MAP_WORKSPACE_LOCATION	1410
MPIP_MAP_WORKSPACE_PAGING_COUNT	1410
MPIP_MAP_WORKSPACE_PAGING_SIZE	1410
Map sliding century properties	1411
MPIP_MAP_SLIDING_CENTURY	1411
MPIP_MAP_SLIDING_CENTURY_CCLOOKUP	1411
Map custom validation properties	1411
MPIP_MAP_CUSTOM_VALIDATION	1411
MPIP_MAP_CV_COMPONENT_RULE	1411
MPIP_MAP_CV_PRESENTATION_ERROR	1412
MPIP_MAP_CV_RESTRICTION_ERROR	1412
MPIP_MAP_CV_SIZE_ERROR	1412
MPIP_MAP_CV_VALIDATION_ERROR	1412
Map audit properties	1412
MPIP_MAP_AUDIT_BURST_DATA	1413
MPIP_MAP_AUDIT_BURST_DATA_SIZE_VALIDATION	1413
MPIP_MAP_AUDIT_BURST_EXECUTION	1413
MPIP_MAP_AUDIT_DIRECTORY	1413
MPIP_MAP_AUDIT_DIRECTORY_CUSTOM_VALUE	1413
MPIP_MAP_AUDIT_FILENAME	1413
MPIP_MAP_AUDIT_FILENAME_ACTION	1413
MPIP_MAP_AUDIT_FILENAME_CUSTOM_VALUE	1414
MPIP_MAP_AUDIT_LOCATION	1414
MPIP_MAP_AUDIT_MEMORY_SIZED	1414
MPIP_MAP_AUDIT_SETTINGS_DATA	1414
MPIP_MAP_AUDIT_SETTINGS_MAP	1414
MPIP_MAP_AUDIT_SUMMARY_EXECUTION	1414
MPIP_MAP_AUDIT_SWITCH	1414
Map retry properties	1415
MPIP_MAP_MAP_RETRY	1415
MPIP_MAP_MAP_RETRY_ATTEMPTS	1415
MPIP_MAP_MAP_RETRY_INTERVAL	1415
Map object methods	1415
MapGetInputCardNumber	1415
MapGetInputCardObject	1416
MapGetOutputCardNumber	1416
MapGetOutputCardObject	1416
Card object overview	1416
Card object properties	1416
General card properties	1416
MPIP_CARD_DIRECTION	1416
MPIP_CARD_DOCUMENT_VERIFICATION	1417
MPIP_CARD_METADATA_LOCATION	1417
MPIP_CARD_MODE	1417
MPIP_CARD_NAME	1417
MPIP_CARD_NUMBER	1417
MPIP_CARD_OBJECT_COUNT	1417
MPIP_CARD_REUSE_WORK_AREA	1417
MPIP_OBJECT_ERROR_CODE	1407
MPIP_OBJECT_ERROR_MSG	1407
Backup card properties	1418
MPIP_CARD_BACKUP	1418
MPIP_CARD_BACKUP_ACTION	1418
MPIP_CARD_BACKUP_DIRECTORY	1418
MPIP_CARD_BACKUP_DIRECTORY_CUSTOM_VALUE	1419
MPIP_CARD_BACKUP_FILENAME	1419
MPIP_CARD_BACKUP_FILENAME_CUSTOM_VALUE	1419
MPIP_CARD_BACKUP_FILEPATH	1419
MPIP_CARD_BACKUP_WHEN	1419

Card object methods	1419
CardGetAdapterObject	1419
CardGetAdapterType	1420
CardGetMapObject	1420
CardOnNotify	1420
CardOverrideAdapter	1420
Adapter object overview	1420
Adapter object properties	1420
General adapter properties	1421
MPIP_ADAPTER_ABBREV	1421
MPIP_ADAPTER_COMMANDLINE	1421
MPIP_ADAPTER_CONTEXT	1422
MPIP_ADAPTER_MSG_COLLECTION	1422
MPIP_ADAPTER_NUM_MESSAGES	1422
MPIP_ADAPTER_PROVIDER_CODE	1422
MPIP_ADAPTER_PROVIDER_MSG	1422
MPIP_ADAPTER_TYPE	1422
MPIP_ADAPTER_USER_DATA	1422
MPIP_ADAPTERUSESAMECONN	1423
MPIP_ADAPTER_WILDCARD	1423
MPIP_OBJECT_ERROR_CODE	1407
MPIP_OBJECT_ERROR_MSG	1407
Adapter retry properties	1423
MPIP_ADAPTER_RETRY	1423
MPIP_ADAPTER_RETRY_ATTEMPTS	1423
MPIP_ADAPTER_RETRY_INTERVAL	1424
Adapter on failure properties	1424
MPIP_ADAPTER_ON_FAILURE	1424
Adapter on success properties	1424
MPIP_ADAPTER_ON_SUCCESS	1424
Adapter scope properties	1424
MPIP_ADAPTER_SCOPE	1424
Adapter warning properties	1425
MPIP_ADAPTER_WARNINGS	1425
Adapter aggregation properties	1425
MPIP_ADAPTER_FETCH_UNIT	1425
MPIP_ADAPTER_LISTEN_TIME	1425
MPIP_ADAPTER_QUANTITY	1425
Adapter object methods	1425
AdaptGetCardObject	1426
AdaptGetInputStreamObject	1426
AdaptGetOutputStreamObject	1426
TX programming interface function overview	1426
Scenario: running TX programming interface maps	1426
Monitor and manage map overview	1426
Abort	1427
Continue	1427
Pause	1427
RegisterStatusMethod	1427
Load and run map overview	1427
LoadFile	1428
LoadMemory	1428
Refresh	1428
ReloadFile	1428
ReloadMemory	1428
Run	1428
Unload	1428
C API	1429
C API interface methods overview	1429
mpiPropertyGetBinary	1429
mpiPropertyGetCount	1430
mpiPropertyGetInteger	1430
mpiPropertyGetPtr	1430
mpiPropertyGetText	1431
mpiPropertySetBinary	1431
mpiPropertySetInteger	1431
mpiPropertySetPtr	1432
mpiPropertySetText	1432
mpiProperty GetAllPropertiesXML	1433
mpiPropertyGetPropertiesXML	1433
mpiPropertySetPropertiesXML	1434
Object methods	

C API map object methods overview	1434
mpiMapGetInputCardNumber	1434
mpiMapGetOutputCardNumber	1434
mpiMapGetInputCardObject	1435
mpiMapGetOutputCardObject	1435
mpiMapGetUniqueMapInstance	1435
C API card object methods overview	1436
mpiCardGetAdapterObject	1436
mpiCardGetAdapterType	1436
mpiCardGetMapObject	1437
mpiCardOverrideAdapter	1437
Adapter object methods	1437
mpiAdaptGetCardObject	1438
mpiAdaptGetInputStreamObject	1438
mpiAdaptGetOutputStreamObject	1438
Loading, running, and controlling maps	1438
Loading and running maps	1439
mpiMapLoadFile	1439
mpiMapLoadMemory	1439
mpiMapRefresh	1440
mpiMapReloadFile	1440
mpiMapReloadMemory	1440
mpiMapRun	1441
mpiMapUnload	1441
Controlling and monitoring maps	1441
mpiMapAbort	1442
mpiMapContinue	1442
mpiMapPause	1442
mpiMapRegisterStatusMethod	1443
Initialization and termination methods	1443
mpiInitAPI	1443
mpiInitAPIex	1443
mpiTermAPI	1444
Callbacks	1444
MPI_TRACE_CALLBACK_PROC	1444
MPI_LOGFN_CALLBACK	1444
C API examples	1444
Running a single map	1445
Running multiple instances of a map in parallel	1445
Overriding a card in a map	1447
Using a user-provided status method	1447
Using streams to override inputs and outputs	1448
Loading a map from a byte array	1449
Getting and setting properties	1451
Developing a custom adapter	1452
Custom adapters	1452
Developing custom adapters	1452
Adapter registration and the library virtual table	1452
Initializing the adapter	1453
InitializeLibrary	1453
DestroyLibrary	1453
Point to adapter methods	1454
Adapter registration file	1454
Vtable adapter library	1454
Virtual table adapter library	1454
Virtual table library values	1454
Virtual table example structure	1455
Methods provided by adapter	1455
Methods provided by an adapter	1456
Adapter methods used post-instance creation	1456
Adapter instance overview	1457
Adapter instance creation and destruction	1457
CreateAdapterInstance	1458
DestroyAdapterInstance	1458
Connection instance overview	1458
Connection instance creation and destruction	1459
CreateConnectionInstance	1459
DestroyConnectionInstance	1459
Connection interface overview	1460
Connect interface method	1460
Disconnect interface method	1460
CompareConnection interface method	1461

ValidateConnection interface method	1461
Library initialization and termination methods	1462
Listener interface overview	1462
Listen method	1462
Listen method attributes	1463
CombinedListen method	1463
CompareWatches method	1464
Method invocation examples	1464
Get and put method invocation examples	1465
Quantity, FetchUnit, and listen time method invocation examples	1465
Source and target interface overview	1466
Get method overview	1466
Put method overview	1467
OnNotify method overview	1467
BeginTransaction method overview	1468
EndTransaction method overview	1468
Property interface overview	1469
ValidateProperties method overview	1469
C API return codes	1469
Additional adapter methods	1470
Wildcard methods overview	1470
mpiCompareWildcard method overview	1470
mpiExtractWildcard method overview	1471
mpiMapReportStatus method overview	1471
Passing data to and from maps	1471
Receiving data from a map	1471
Sending data to a map	1472
Listener scenario overview	1472
Listener scenario: adapters process similar events	1473
Listener scenario: Resource Manager processes similar events	1473
Listener scenario: Listener and map connection threads are combined	1473
Threads, connections, and transactions overview	1474
COM API	1474
COM API interfaces overview	1474
IMFactory interface overview	1475
IMFactory virtual table order	1475
IMFactory::InitMercAPI method	1475
IMFactory::ExitMercAPI method	1475
IMFactory::MapLoadMemory method	1476
IMFactory::MapReloadMemory method	1476
IMFactory::MapLoadFile method	1476
IMFactory::MapReloadFile method	1477
IMMap interface overview	1477
IMMap virtual table order	1477
IMMap::Run method	1478
IMMap::Refresh method	1478
IMMap::Abort method	1479
IMMap::Pause method	1479
IMMap::Continue method	1479
IMMap::GetInputCard method	1479
IMMap::GetOutputCard method	1480
IMMap::MapSetupNotifications method	1480
IMMap::GetInputCardCount method	1480
IMMap::GetOutputCardCount method	1481
IMMap::GetPropertyCount method	1481
IMMap::MapUnload method	1481
IMMap::GetIntegerProperty method	1481
IMMap::SetIntegerProperty method	1482
IMMap::GetTextProperty method	1482
IMMap::SetTextProperty method	1482
IMMap::GetBinaryProperty method	1483
IMMap::SetBinaryProperty method	1483
IMMap::GetPropertiesXML method	1484
IMMap::SetPropertiesXML method	1484
IMMap::GetAllPropertiesXML method	1484
IMMap properties	1485
IMCard interface overview	1485
IMCard virtual table order	1485
IMCard::GetAdapter	1485
IMCard::GetAdapterType	1486
IMCard::OverrideAdapter	1486
IMCard::GetMap	1486

IMCard::GetIntegerProperty	1487
IMCard::SetIntegerProperty	1487
IMCard::GetTextProperty	1487
IMCard::SetTextProperty	1488
IMCard::GetBinaryProperty	1488
IMCard::SetBinaryProperty	1488
IMCard::GetPropertiesXML	1489
IMCard::SetPropertiesXML	1489
IMCard::GetAllPropertiesXML	1489
IMCard properties	1490
IMAdapter interface overview	1490
IMAdapter virtual table order	1490
IMAdapter::GetCard	1491
IMAdapter::GetStream	1491
IMAdapter::GetInputStream	1491
IMAdapter::GetOutputStream	1491
IMAdapter::GetIntegerProperty	1492
IMAdapter::SetIntegerProperty	1492
IMAdapter::GetTextProperty	1493
IMAdapter::SetTextProperty	1493
IMAdapter::GetBinaryProperty	1493
IMAdapter::SetBinaryProperty	1494
IMAdapter::GetPropertiesXML	1494
IMAdapter::SetPropertiesXML	1494
IMAdapter::GetAllPropertiesXML	1495
IMAdapter properties	1495
IMStream interface overview	1495
IMStream virtual table order	1495
IMStream::Seek	1496
IMStream::Tell	1496
IMStream::IsEnd	1496
IMStream::GetSize	1497
IMStream::SetSize	1497
IMStream::Write	1497
IMStream::Modify	1498
IMStream::Read	1498
IMStream::Flush	1498
IMStream::ReadPage	1499
IMStream::NewStreamPage	1499
IMStream::DeletePage	1499
IMStreamPage interface overview	1500
IMStreamPage virtual table order	1500
IMStreamPage::GetSize	1500
IMStreamPage::GetData	1500
IMStreamPage::DeletePage	1501
COM API object hierarchy	1501
Java API	1501
Java Tools	1501
Python API	1501
REST API	1501
Introduction	1502
Swagger documentation	1502
Running a map directly	1502
Synchronous direct calls	1502
Asynchronous direct calls and map status queries	1503
REST API output	1503
Cataloging a map	1505
Tutorial	1505
Important: Delete the artifacts from asynchronous map runs	1505
Map caching	1506
Map execution logs	1506
RMI API	1506
RMI server overview	1506
RMI server implementation	1506
RMI server error handling	1506
RMI server process modes	1507
RMI server single process mode	1507
RMI server multi-process mode	1507
RMI server in-process mode	1507
RMI server configurations	1507
RMI server properties overview	1507
RMI server properties	1508
RMI server mode property	1508

RMI server port property	1508
RMI server log.mode.full property	1509
RMI server pool.mode.managed property	1509
RMI server pool.max.process.count property	1509
RMI server pool.max.keep.idle.count property	1509
RMI server pool.max.idle.time property	1509
RMI server pool.acquire.process.timeout property	1510
RMI server pool.map.auto.unload.timeout property	1510
Starting the RMI server	1510
RMI API client installation overview	1510
Installing the DK on the same machine	1511
Installing the DK on a separate machine	1511
RMI API configuration overview	1511
RMI API in-process configuration	1511
RMI API single process configuration	1511
RMI API multi-process configuration	1512
Web Services	
Web Services	1512
SOAP (WSDL) Adapter	1057
System requirements	1057
Command alias	1057
SOAP Adapter commands	1057
List of commands	1057
Decode (-DECODE)	1058
Encode (-ENCODE)	1058
Header (-HDR)	1058
Raw (-RAW)	1058
Return (-RETURN)	1059
Return option and XPath	1059
SOAPAction (-SA)	1060
Trace (-T)	1060
Transport (-TRANSPORT)	1061
Syntax summary	1061
Data sources	1061
Example	1061
Data targets	1061
Example	1062
Source options behavior	1062
Using the adapter	1062
Consumer scenario	1062
Provider scenario	1063
Response	1063
Example files	1063
Return codes and error messages	1063
Messages	1063
WSDL Importer	396
z/OS Platform	
Overview	1519
Related publications for z/OS	1520
SMP/E installation	1521
IBM Transformation Extender - Common Components	1521
Configuring the common components	1521
Accessing DB2 databases in CICS and IMS environments	1521
Accessing DB2 databases in Batch and USS environments	1521
Allocating an zFS on USS environments	1521
Defining programs to CICS region for IBM Transformation Extender CICS Adapter server-side components	1522
IBM Transformation Extender with Command Server	1522
Configuring IBM Transformation Extender with Command Server	1522
Binding the DB2 plan	1522
Examples	1522
IBM Transformation Extender with Launcher	1522
Configuring IBM Transformation Extender with Launcher	1523
Configuring UNIX System Services (USS)	1523
IBM Transformation Extender for Application Programming	1523
CICS environment	1523
CICS configuration requirements	1523
Updating CICS CSD definitions and DFHRPL	1524
XPLINK and non-XPLINK	1524
DTXCTBL CSD definitions	1524
DTXCTBL CSD definitions for XPLINK	1524
DTXCTBL CSD definitions for non-XPLINK	1524
Loading Maps	1525
Language Environment (LE) requirement	1525

(Optional) Setting up the DB2 (z/OS) Adapter to use the CICS CAF	1525
Multiple versions of the IBM Transformation Extender for Application Programming for the CICS runtime environment in a single CICS region	1526
Description	1526
Installing multiple versions	1526
Additional updates to CICS CSD definitions	1526
Sizing temporary storage	1527
Sizing temporary storage example	1527
(Optional) Defining and loading the Resource Name file	1528
Examples for the CICS environment	1528
IMS environment	1528
Configuration steps for the IMS environment	1528
(Optional) Binding the DB2 plan	1528
UNIX System Services (USS) Software Development Kit	1528
Information common to all features	1529
Binding the Data Base Request Module (DBRM)	1529
Upgrading from an earlier version	1529
Binding the DBUTILE application plan package	1529
Additional configuration on UNIX System Services (USS)	1529
Modifying the DTXINST JCL	1530
MAKEDIR step	1530
INSTALL step	1531
Setting USS environment variables	1531
Using IBM Transformation Extender for Application Programming on USS	1531
(Optional) Defining DB/2 system load library	1532
(Optional) Defining IBM MQ system load libraries	1532
(Optional) Accessing Oracle 9i database	1532
(Optional) Using Language Environment parameters	1532
(Optional) Defining port numbers when using the E-mail Adapter	1533
Preserving the encoding of XML files	1533
Support for Getting Started Sub-capacity Pricing (GSSP)	1533
Support for z/OS XML System Services processing	1533
Invoking z/OS System Services parsing	1533
z/OS XML System Services parsing limitations	1533

Transformation Extender for Integration Servers

Overview	1534
System requirements	1534
IBM Transformation Extender for Sterling B2B Integrator	1534
Product uses for Sterling B2B Integrator customers	1535
Product uses for IBM Transformation Extender customers	1535
Advantages of IBM Transformation Extender for Sterling B2B Integrator	1536
Examples	1536
Standards, globalization, and data-type support	1537
Transforming data	1537
WTX Map service overview	1537
Services that support IBM Transformation Extender maps	1538
Business process concepts	1538
Sterling B2B Integrator user interface permissions	1538
Developing a business process that uses the WTX Map service	1538
WTX Map service	1539
Specifying the map for the WTX Map service	1540
Using maps from the map repository	1540
Using maps on the server file system	1540
Dynamically specifying maps at runtime	1540
Dynamically loading RUN maps from the Sterling B2B map repository	1541
Map caching	1541
Resource Registry	1542
Modifying resource names	1542
Specifying the Resource Registry location	1542
Input and output cards	1542
Wired and unwired cards	1542
Configuring input and output cards in maps	1542
XML entities	1543
Overriding map and card properties by using map settings	1543
Required property for secure Map Test HTTP Server adapter	1543
Accessing the Sterling B2B Integrator data harness from maps	1544
Data harness functions	1544
Sending output data from a map to a Sterling B2B Integrator document	1546
Sending output data by using a PUT wire	1546
Configuring the WTX Map service PUT rule	1546
PUT rule syntax	1546
WTX Map service BPML	1547
Validating the business process modeling language	1547
Deploying IBM Transformation Extender maps	1547

Deploying to the map repository	1548
Deploying to a file system	1548
Testing IBM Transformation Extender maps	1548
Running a business process	1548
Running multiple concurrent instances of maps	1549
Running maps locally or in a separate JVM	1549
Based on input-data-size threshold	1549
Using service configurations and an adapter container JVM	1549
Assigning the WTX Map service to a container node	1550
Configuring the Translation service to use IBM Transformation Extender maps	1550
Problem determination	1550
WTX Map service correlations	1550
Diagnostic tools	1551
Sterling B2B Integrator service translation report	1551
Configuring IBM Transformation Extender logging properties in Sterling B2B Integrator	1551
IBM Transformation Extender logging properties	1552
Map trace, audit log, and work space directories	1552
Performance tuning	1552
IBM Transformation Extender for IBM Integration Bus	1552
Start here	1553
Getting started	1553
Developing message flow applications	1553
How can I diagnose problems?	1553
Product overview	1554
Introduction	1554
Product uses	1554
For IBM Integration Bus users	1554
For IBM Transformation Extender users	1555
Transforming data	1555
TX Map node overview	1555
Developing a message flow	1556
Using the TX Map node within a message flow	1556
Using the TX Map node with a source map and a single input	1557
Using the TX Map node with a precompiled map and a single input	1557
Using the TX Map node with a source map and multiple inputs	1558
Using the TX Map node with a precompiled map and multiple inputs	1559
Deploying your message flow	1560
Deploying IBM Transformation Extender maps	1560
Using a source map with a TX Map node	1561
Using a precompiled map with a TX Map node	1561
Running your message flow	1561
Problem determination	1562
IBM Transformation Extender map failures	1562
TX Map node exceptions	1562
Using IBM Integration Bus log and trace files	1563
Sending audit data to output and failure terminals through the local environment tree	1563
References	1563
Product requirements	1564
Supported message domains	1564
BLOB domain	1564
DataObject domain	1565
DFDL domain	1565
MRM domain	1566
XMLNSC domain	1566
Other domains	1567
Deploying map resources	1567
Triggering maps to run	1567
Input nodes and events	1568
Triggering maps to run by a single event	1568
Triggering maps to run by multiple events	1568
Triggering maps to run in complex message flows	1569
Input cards in maps	1569
Map triggered by multiple inputs	1569
Collector node	1569
Wired input cards	1570
Unwired input cards	1570
Output cards in maps	1570
Creating the output terminals	1570
Connecting output terminals to nodes	1571
Configuring connected output terminals	1571
Wired output cards	1571
Unwired output cards	1572
Sending output data from a map to an IBM Integration Bus node	1572

Specifying the map for the TX map node	1572
Using source maps	1573
Using precompiled maps	1573
Changing a source map compared to changing a compiled map	1574
Changing a message flow to point to a different map	1574
Deploy-time TX Map node configuration	1575
Overriding map properties at run time	1575
Dynamically overriding a map with a compiled map	1575
Map caching	1576
LocalEnvironment DynamicMap and MapServerLocation caching	1576
Wildcards	1576
Using wildcards	1577
Map Settings interface	1577
Resource Registry	1577
Modifying resource names	1577
Setting resource registry on execution groups	1578
Samples	1578
Containers	
Containers Overview	1579
Runtimes Server Container	1579
Introduction	1579
Installing ITX Runtime Server	1579
Prerequisites	1580
System Requirements	1580
Installing ITX Runtime Server from Command Line	1580
Installing in an Online Cluster through CASE	1580
Installing in an Online Cluster through Native helm CLI Commands	1581
Installing in Air-Gapped (Offline) OpenShift Cluster	1581
Prepare Bastion Host or Portable Device	1581
Prepare CASE	1581
Configure Registry Auth	1582
Mirror Images	1582
Configure Cluster for Airgap	1582
Install ITX definitions to the cluster	1582
Verifying Installation	1583
Configuration	1583
Redis Configuration	1585
SSL/TLS Configuration	1586
Option 1 - Produce Secret with ca.crt, tls.crt and tls.key PEM values	1586
Storage	1586
Access Modes	1587
Cloud Object Storage	1587
Extra Content	1587
Backup	1588
Encryption	1588
SecurityContextConstraints Requirements	1588
ibm-itx-rs Container Image	1589
ibm-itx-rs Image and Container	1589
Configuring a single-container deployment of ibm-itx-rs	1590
Example override file	1591
ibm-itx-rs REST API	1591
ibm-itx-rs Usage Example	1592
Cataloged Map Invocation	1594
ibm-itx-rs V2 Usage Example	1594
Packs	1595
Healthcare	1595
HIPAA Compliance Check or HIPAA Data Compliance	1595
Setup for Design Studio V1-based map executions	1595
Setup for Design Studio V2-based map executions	1596
HL7 Compliance Check	1596
Other Healthcare Components	1597
Supply Chain EDI	1597
EDI Compliance check	1597
Setup for Design Studio V1-based map executions	1597
Setup for Design Server V2-based flow executions	1598
Compliance Check Execution	1598
Supply Chain Examples	1599
SAP Applications	1599
Install and run SAP maps	1600
Run SAP example maps using endpoint for V1 REST API	1600
Run SAP example maps using endpoint for V2 REST API	1601
Pack for Financial Payments (FINPAY)	1601

Prerequisites	1602
Configuring SWIFT MT JVC	1602
Building maps for Linux	1602
Porting artifacts to UNIX host	1603
Example map execution on ITX-RS using endpoint for V1 REST API	1603
Overrides on input, output cards due to backward slashes	1603
Example map execution on ITX-RS using endpoint for V2 REST API	1604
Not available for execution on ITX-RS	1604
Pack for Financial Payments Plus (FINPLUS)	1605
Prerequisites	1605
Configuring CBPR+ Translation	1605
Building maps for Linux	1606
Porting artifacts to UNIX host	1606
Example map execution on ITX-RS using endpoint for V1 REST API	1606
Example map execution on ITX-RS using endpoint for V2 REST API	1608
Example flow execution on ITX-RS using endpoint for V2 REST API	1610
IBM MQ Client	1614
Helm Chart	1615
ibm-itx-rs-prod Package	1615
Installing ibm-itx-rs-prod Package	1615
Kubectl commands	1615
Bulk copy	1616
Deployment Guidelines	1616
GDPR	1616
CIS Benchmarks	1617
Auditing	1617
Scaling	1617
Uninstalling	1618
Upgrades and Rollbacks	1618
Limitations	1618
Migration	1618

Pack for SAP Applications

Introduction	1618
Pack for SAP Applications overview	1619
Integrating information	1619
Overview of SAP R/3 interfaces	1619
Intermediate Documents (IDocs)	1619
Application Link Enabling (ALE)	1619
ALE Message Handler (AMS)	1620
Electronic Data Interchange (EDI)	1620
DXOB/Data Migration Interface (DMI)	1620
Business Application Programming Interface (BAPI)	1620
Setting up your SAP R/3 environment	1620
System requirements and installation	1620
Pack for SAP R/3 Server	1620
SAP Gateway	1620
Configuring the SAP R/3 system	1621
1. Creating a logical system (BD54)	1621
2. Creating RFC destination for outbound data (SM59)	1621
Setting up an RFC destination	1622
Gateway options	1622
tRFC options	1622
Collective error processing	1623
3. Creating a Distribution Model (BD64)	1623
Creating message types for the Distribution Model	1623
4. Generating a Partner Profile (BD82)	1624
5. Manually creating partner profiles (WE20)	1624
Inbound parameters	1625
Creating SAP connections, actions, and schemas	1625
Overview of the Importer Wizard	1625
Running the Importer Wizard	1625
The R/3 adapters	1626
Overview of the R/3 adapters	1626
Unicode support	1626
Using R/3 adapter commands	1627
Execution command overrides	1627
Card settings	1627
R/3 data retrieval behavior for bursts	1627
OnFailure behavior with R/3 adapters	1627
Adapter commands syntax and usage	1627
RUN, GET, and PUT functions	1628
RUN example	1628
GET example	1628
PUT example	1628

From the Map Designer	1628
From the Integration Flow Manager	1629
Connection commands	1629
Required connection commands for JALE sources	1629
Required connection commands for JALE targets and for calling a BAPI	1629
Adapter commands list	1629
Program ID (-A)	1630
Audit (-AR3)	1631
Backup (-B)	1631
Load Balancing (-BAL)	1631
Bytes Per Character (-BPC)	1632
Client Number (-C)	1632
Character Set Encoding (-enc)	1632
Destination (-D)	1633
Gateway Host (-G)	1633
IDoc Field Generation (-GEN)	1634
Group (-GROUP)	1634
Host ID (-H)	1634
Logon Language (-L)	1635
Listen (-LSN)	1635
Listener Threads (-N)	1635
Password (-P)	1635
Packet Size (-PKT)	1636
Reprocess backup files (-R)	1636
System ID (-S)	1636
Timeout (-timeout)	1636
Trace (-T)	1636
Transaction ID (-TID)	1637
IDoc Type (-TY)	1637
User ID (-U)	1637
Gateway Service (-X)	1637
Syntax summaries for R/3 adapters	1638
JALE adapter commands syntax summary	1638
JBAPI adapter commands syntax summaries	1638
R/3 adapter aliases	1638
Using R/3 system commands	1639
Intermediate Documents (IDocs)	1639
Overview of IDocs	1639
IDoc structural format	1639
Generating the IDoc parser report	1640
Using the Importer Wizard for IDocs	1640
Understanding the IDoc schema	1640
Implementing an ALE interface	1640
IDoc selection and metadata download (WE63)	1641
Creating a Unicode metadata file	1641
Mapping	1641
Using the Importer Wizard for ALE	1641
Creating an outbound map	1641
Creating an input card and configuring outbound processing	1641
Creating an output card and configuring inbound processing	1642
Inbound and outbound processes	1642
Inbound ALE to SAP	1642
Outbound ALE from SAP	1643
Control records for IDoc mapping	1643
Control record example	1643
Sending EDI IDocs using ALE	1644
Data Transfer Objects (DXOB)	1644
Overview of Data Transfer Objects	1644
Generating DXOB metadata	1644
Data Transfer Workbench (SXDA)	1645
Mapping	1645
Using the SAP:DXOB Importer	1645
DXOB formats	1645
Understanding the DXOB schema	1645
Creating a map for DXOB formatted data	1646
Transferring the Mapped Data	1646
Transferring the mapped data to the application layer	1646
Creating the Batch Input Session	1646
Processing the Batch Input Session (SM35)	1647
Business Application Programming Interface (BAPI)	1647
Overview of the Interface (BAPI)	1647
Mapping	1648
Understanding the BAPI schema	1648
Calling a BAPI from a map	1648

Unicode	1649
SAP S/4 HANA Adapter	1649
Troubleshooting	1649
Troubleshooting tools	1649
MapAudit log	1649
Data Log	1650
Execution audit	1650
ExecutionLog per burst	1650
ExecutionSummary per map	1650
Map Settings	1650
Data settings	1650
R/3 adapter audit files	1650
R/3 adapter trace files	1651
R/3 adapter trace - Verbose option	1651
R/3 return codes and error messages	1651
Viewing R/3 source and target data	1652
Backup settings	1652
Remaining temporary data in TIDDATA directory	1652

Pack for Financial Payments

Overview	
Financial Services standards	1652
The IBM Transformation Extender product family	1652
International Program License Agreement	1652
Installation and uninstallation	1653
FIX component	
Overview of the FIX component	1653
What the FIX component contains	1653
Major components of the pack	1653
The utilities subdirectory	1653
The trees subdirectory	1654
The schemas subdirectory	1654
FIX component examples	1654
FIX type trees and message structure	1655
FIX message protocol	1655
Message types per FIX version number	1655
Session messages	1656
EventCommunication messages	1656
Indication messages	1657
MarketData messages	1657
QuotationNegotiation messages	1657
SecuritiesReferenceData messages	1657
MarketStructureReferenceData message types	1658
PartiesAction messages	1658
PartiesReferenceData messages	1658
SingleGeneralOrderHandling messages	1659
ProgramTrading messages	1659
OrderMassHandling messages	1659
CrossOrders messages	1660
MultilegOrders messages	1660
Allocation messages	1660
SettlementInstruction messages	1660
SettlementStatusManagement messages	1660
RegistrationInstructions messages	1661
TradeCapture messages	1661
TradeManagement messages	1661
Confirmation message types	1661
PayManagement messages	1661
PositionMaintenance messages	1662
CollateralManagement messages	1662
MarginRequirementManagement messages	1662
AccountReporting messages	1662
BusinessReject messages	1663
Network messages	1663
UserManagement messages	1663
Application messages	1663
Type tree features	1663
FIX message structure	1664
Message syntax	1664
Additional validation	1664
Example message structure	1664
Header example	1664
Header fields	1665
Body example	1666

Body fields	1666
Message trailer	1667
Trailer fields	1667
Additional validation recommendations	1667
FIXML type trees	1668
FIXML message protocol	1668
Import source	1668
Message types per FIXML version number	1668
Session messages	1656
EventCommunication messages	1670
Indication messages	1670
QuotationNegotiation messages	1670
MarketData messages	1670
SecuritiesReferenceData messages	1671
MarketStructureReferenceData messages	1671
PartiesAction messages	1672
PartiesReferenceData messages	1672
SingleGeneralOrderHandling messages	1672
ProgramTrading messages	1673
OrderMassHandling messages	1673
CrossOrders messages	1673
MultilegOrders messages	1674
Allocation messages	1674
SettlementInstruction messages	1674
SettlementStatusManagement messages	1674
RegistrationInstructions messages	1675
TradeCapture messages	1675
TradeManagement messages	1675
Confirmation messages	1675
PayManagement messages	1676
PositionMaintenance messages	1676
CollateralManagement messages	1676
MarginRequirementManagement messages	1676
AccountReporting messages	1677
BusinessReject messages	1677
Network messages	1677
UserManagement messages	1677
Application messages	1677
FIX component examples	1678
Running the examples	1678
Before running the example	1679
FIX mapping from and to FIXML examples	1679
Example scenarios	1679
Example assumptions	1679
Expected results	1680
Running the examples on z/OS	1680
Option 1: Schemas on an MVS data set	1681
Option 2: Schemas on a UNIX hierarchical file system	1681
Reminders	1682
FIX checksum validation	1682
Example scenarios	1682
Example assumptions	1682
Expected results	1683
FIX checksum generation	1683
Example scenarios	1683
Example assumptions	1683
Expected results	1683
Example for Advertisement message	1683
What the example contains	1684
How to run the example in Design Studio	1684
How to run the example in Design Server	1685
Example for Allocation Instruction message	1685
What the example contains	1685
How to run the example in Design Studio	1686
How to run the example in Design Server	1686
Example for Allocation Instruction Ack message	1686
What the example contains	1686
How to run the example in Design Studio	1687
How to run the example in Design Server	1687
Example for Don't Know Trade (DK) message	1688
What the example contains	1688
How to run the example in Design Studio	1688

How to run the example in Design Server	1689
Example for Email message	1689
What the example contains	1689
How to run the example in Design Studio	1690
How to run the example in Design Server	1690
Example for Execution Report message	1691
What the example contains	1691
How to run the example in Design Studio	1691
How to run the example in Design Server	1692
Example for Indication of interest message	1692
What the example contains	1692
How to run the example in Design Studio	1693
How to run the example in Design Server	1693
Example for List Cancel Request message	1693
What the example contains	1694
How to run the example in Design Studio	1694
How to run the example in Design Server	1694
Example for List Execute message	1695
What the example contains	1695
How to run the example in Design Studio	1696
How to run the example in Design Server	1696
Example for List Status message	1696
What the example contains	1696
How to run the example in Design Studio	1697
How to run the example in Design Server	1697
Example for List Status Request message	1698
What the example contains	1698
How to run the example in Design Studio	1698
How to run the example in Design Server	1699
Example for New Order List message	1699
What the example contains	1699
How to run the example in Design Studio	1700
How to run the example in Design Server	1700
Example for News message	1700
What the example contains	1701
How to run the example in Design Studio	1701
How to run the example in Design Server	1701
Example for Order Cancel Reject message	1702
What the example contains	1702
How to run the example in Design Studio	1703
How to run the example in Design Server	1703
Example for New Order message	1703
What the example contains	1703
How to run the example in Design Studio	1704
How to run the example in Design Server	1704
Example for Order Status Request message	1705
What the example contains	1705
How to run the example in Design Studio	1705
How to run the example in Design Server	1706
Example for Quote message	1706
What the example contains	1706
How to run the example in Design Studio	1707
How to run the example in Design Server	1707
Example for Quote Request message	1707
What the example contains	1708
How to run the example in Design Studio	1708
How to run the example in Design Server	1708
NACHA component	
Overview of the NACHA component	1709
ACH network terminology	1709
NACHA credit transfer and direct debit processes	1710
Credit processes	1710
Debit process	1710
The ACH file	1710
Configuration	1710
Directory structure	1710
examples directory	1711
trees directory	1712
Support for streaming format messages	1712
NACHA type trees	1712
Overview of the NACHA type trees	1713
Data elements	1713

File category	1713
Groups and records	1713
Summary of ACH applications	1714
Return entries	1714
Notification of change entries (COR)	1714
Acknowledgement records	1715
Analysis errors	1715
Additional validation recommendations	1715
Creating an application specific type tree	1715
Copying NACHA type trees	1716
Removing unnecessary application types	1716
Creating the target type tree	1716
Merging the application type to the new type tree	1716
NACHA component examples	1717
ACH EDI example	1718
What the example contains	1718
How to run the example in Design Studio	1718
How to run the example in Design Server	1719
Scenarios	1719
Assumptions	1719
ACH SWIFT example	1719
What the example contains	1719
How to run the example in Design Studio	1720
How to run the example in Design Server	1720
Scenarios	1720
Assumptions	1720
ACH to ISO 20022 Credit Transfer example	1721
What the example contains	1721
How to run the example in Design Studio	1722
How to run the example in Design Server	1723
Scenarios	1723
Assumptions	1723
ACH to ISO 20022 Direct Debit example	1723
What the example contains	1724
How to run the example in Design Studio	1724
How to run the example in Design Server	1725
Scenarios	1725
Assumptions	1725
ACH ISO 20022 Rejects example	1726
What the example contains	1726
How to run the example in Design Studio	1727
How to run the example in Design Server	1728
Scenarios	1728
Assumptions	1728
ACH ISO 20022 Returns example	1728
What the example contains	1729
How to run the example in Design Studio	1729
How to run the example in Design Server	1730
Scenarios	1730
Assumptions	1730
XML to ACH PPD example	1731
What the example contains	1731
How to run the example in Design Studio	1731
How to run the example in Design Server	1731
Scenarios	1732
Assumptions	1732
ACH Returns and Notification of Change (COR) example	1733
What the example contains	1733
How to run the example in Design Studio	1733
How to run the example in Design Server	1734
Scenarios	1734
Assumptions	1734
ACH Refused Notification of Change Entry example	1735
What the example contains	1735
How to run the example in Design Studio	1735
How to run the example in Design Server	1735
Scenarios	1736
Assumptions	1736
ACH Dishonored Return example	1736
What the example contains	1736
How to run the example in Design Studio	1737
How to run the example in Design Server	1737

Scenarios	1737
Assumptions	1737
ACH Contested Dishonored Return Entry example	1738
What the example contains	1738
How to run the example in Design Studio	1738
How to run the example in Design Server	1738
Scenarios	1739
Assumptions	1739
ACH Payment Related Information example	1739
What the example contains	1739
How to run the example in Design Studio	1740
How to run the example in Design Server	1740
Scenarios	1741
Assumptions	1741
ACH Return Entries Report example	1742
What the example contains	1742
How to run the example in Design Studio	1743
How to run the example in Design Server	1743
Example scenario	1743
Example assumptions	1743
Application 1	1743
Application 2	1744
Application 3	1744
Application 4	1745
ACH Validation example	1745
What the example contains	1745
How to run the example in Design Studio	1746
How to run the example in Design Server	1746
Scenarios	1746
Assumptions	1747
How to download output files	1747
IBM Sterling B2B Integrator examples	1748
IBM Sterling B2B Integrator	1748
Directory structure of the IBM Sterling B2B Integrator examples	1748
ACH Deenvelope CTX example	1749
ACH Envelope CTX example	1749
ACH Deenvelope DNE example	1749
ACH Envelope DNE example	1749
Deploying and executing the IBM Sterling B2B Integrator examples	1750
Deploying the example maps to IBM Sterling B2B Integrator	1750
Deploying the map using the Design Studio	1750
Deploying the map using IBM Sterling B2B Integrator	1750
Running the ACH service examples	1750
Importing IBM Sterling B2B Integrator artifacts for ACH service examples	1751
Running the ACH service examples from the Transformation Extender	1751
Running the ACH service examples from IBM Sterling B2B Integrator	1752
Expected results	1752
CTX Deenvelope example expected results	1752
DNE Deenvelope example expected results	1752
CTX Envelope example expected results	1753
DNE Envelope example expected results	1753
Design considerations of the examples	1753
IBM Sterling B2B Integrator data harness	1753
Thread safety	1753
Specific map and type tree settings	1754
Burst	1754
Audit settings	1754
ACH Deenvelope service data audit settings	1754
ACH Envelope service data audit settings	1755
Restart	1755
ACH Deenvelope service	1755
ACH Envelope service	1755
Custom type trees	1755
5010_820.mtt type tree	1756
ach_pymtRltdInfo.mtt type tree	1756
ach_dne_flat.mtt type tree	1756
pymt_ordr_rpt.mtt	1756
ach_ctx_csv.mtt	1756
ACH service XML schemas	1756
SEPA component	1757
Overview of the SEPA component	1757
Principal components	1757

The SEPA initiative	1757
SEPA message standards	1758
Message overview	1758
Credit Transfer messages	1758
Credit Transfer - Inter-PSP messages	1758
Credit Transfer - Customer-to-PSP messages	1760
Direct debit messages	1760
Direct Debit - Inter-PSP messages	1760
Direct Debit - Inter-PSP B2B messages	1760
Direct Debit - Inter-PSP Core messages	1761
Direct Debit - Customer-to-PSP messages	1761
Direct Debit - Customer-to-PSP B2B messages	1761
Direct Debit - Customer-to-PSP core messages	1761
Direct Debit - E-Mandate messages	1762
Direct Debit - E-Mandate B2B messages	1762
Direct Debit - E-Mandate Core messages	1762
About the UNIFI message standard	1762
EPC rulebook validation of SEPA messages	1762
Business to Business direct debit implementation	1763
External Purpose Code List	1763
Configuration	1763
Major components of the SEPA component	1763
Examples	1764
mapsandschemas	1764
BICPlusIBAN Utility	1764
res	1765
type_trees	1765
Running the SEPA component on z/OS	1765
Preparing work for execution on z/OS	1765
Working with the default project	1766
Validation overview	1766
sepvalid	1766
SEPA validation customization	1767
Unenforced usage rules	1767
Validation details	1767
Maps inside the validation framework	1767
Stepped map process	1768
Validation framework map naming conventions	1768
How to run SEPA Compliance in Design Studio	1768
How to run the SEPA Compliance in Design Server	1769
How to download output files	1769
Types of customization for SEPA validation	1769
Adding new schemas	1769
Adding new validation steps	1770
About NVP WorkTemp(s)	1770
Add a custom schema	1770
Modifying validation steps	1770
Modify or add validation rules	1771
Guidelines for modifying or adding validation rules	1771
Adding error messages	1771
Handling failures and unexpected results	1771
Checking output card 1	1771
Checking output card 3	1772
Example of output card 3 validation errors	1772
SEPA schema error parser (Deprecated)	1772
Running the error parser	1772
BIC plus IBAN validation and lookup	1772
The BICPlusIBAN directory	1773
The BICPlusIBAN utility	1773
BICPlusIBAN support	1773
Turning BIC and IBAN checking on and off	1774
Disabling BIC validations	1774
Disabling IBAN validations	1774
Domestic format support	1774
Domestic format for the United Kingdom	1775
Domestic format example for the United Kingdom	1775
Domestic format for Germany	1775
Domestic format example for Germany	1775
Domestic format for France	1775
CFONB domestic format converter for France	1776
Payment message record types	1776
MINOS format converter for France	1776

Domestic format example for France	1777
Domestic format for Italy	1777
Domestic format example for Italy	1777
SEPA format converters for SWIFTNet FIN	1777
SWIFTNet FIN format example	1777
SEPA component examples	1777
Validation using the EPC schemas example	1778
Download the EPC schemas	1778
Example maps and schemas	1778
How to run the example in Design Studio	1779
Notes about the maps	1779
How to run the example in Design Server	1779
Building multiple maps using Deploy	1780
How to download output files	1780
Notes about the maps	1781
Example for France	1781
Directory structure	1781
Example for France components	1781
Type trees	1782
Maps	1782
Example files	1782
How to run the example in Design Studio	1782
Customization notes	1783
How to run the example in Design Server	1783
How to download output files	1783
Customization notes	1783
Example for Germany	1784
Directory structure	1784
Example for Germany components	1784
Type trees	1785
Maps	1785
Example files	1785
How to run the example in Design Studio	1785
How to run the example in Design Server	1786
How to download output files	1786
Example for the UK	1786
Directory structure	1787
Data directory	1787
Maps	1787
sepa_uk.mms	1787
uk_sepa.mms	1788
Type trees	1788
How to run the example in Design Studio	1788
How to run the example in Design Server	1788
How to download output files	1789
Customize the maps	1789
Example for Italy	1789
Example for Italy components	1790
Type trees	1790
Maps	1790
Example files	1790
Input files for it_sepa.mms	1791
Input files for sepa_it.mms	1791
Other configuration files	1791
How to run the example in Design Studio	1791
How to run the example in Design Server	1792
Building multiple maps using Deploy	1792
How to download output files	1793
PAIN / PACS example	1793
Conversions demonstrated by the example	1793
Example maps and example files	1794
Conversions demonstrated by the example	1794
How to run the example in Design Studio	1794
How to run the example in Design Server	1794
How to download output files	1795
Customize the maps	1795
Passthrough Example for z/OS	1795
Using the example files	1795
Sample JCL for the validation framework maps example	1796
Example files	1796
How to run the example	1796
SWIFTNet FIN / SEPA example	1797

Example files	1797
Data directory	1797
Maps	1798
sepa_fn.mms	1798
Type trees	1798
How to run the example in Design Studio	1798
Conversion notes	1799
How to run the example in Design Server	1799
Building multiple maps using Deploy	1800
How to download output files	1800
SWIFT component	
Overview of the SWIFT component	1800
How to set up the Transformation Extender project	1801
SWIFT common components	1801
SWIFT MT components	1801
SWIFT MX components	1802
Folder structure	1802
Additional Configuration for MT JVC validation	1802
Common SWIFT components	1802
Common data components	1803
Common map components	1803
Common schema components	1803
Common type tree components	1804
SWIFT MT Components	1804
SWIFT MT core type trees	1804
Syntax validation in SWIFT MT type trees	1804
Supported message types in SWIFT MT type trees	1805
swift_iso7775_ccyy	1805
swift_iso15022_ccyy	1806
Other SWIFT MT trees	1806
Java Validation Component	1806
Overview of the JVC	1806
How to configure the JVC	1807
jvalccyy.prop	1807
How to specify a jvalccyy.prop file location	1808
message.definitions.directory	1809
validate_messages	1810
JVC installation	1811
Installing JVC on non-Windows environments	1811
Logical Message Format	1811
LMF overview	1812
LMF structure	1812
LMF supported message types	1812
LMF message mapping	1814
LMF Header	1814
LMFType attributes	1814
msgtype attributes	1815
Payments	1815
MT 101 request for transfer	1816
Alternative LMF mapping for Tag 35B	1818
ToLMF map	1818
FromLMF map	1819
Universal Confirmation Components	1819
Overview of the Universal Confirmation Components	1820
Configuration Files	1820
Validation Configuration Files	1820
Validation Configuration File scenarios	1821
Translation Configuration File	1821
Translation Configuration File scenarios	1821
Configuration file location	1822
SWIFT MX components	1823
SWIFT MX components overview	1823
SWIFT MX schemas and type tree	1823
MX validation	1823
MX validation overview	1824
Framework map	1824
MX validation maps	1824
MX schema validation maps	1825
MX validation framework map	1825
MX configuration file	1825
Schema name	1826
Map trace	1826

Extended validation settings	1826
Individual MX validation maps	1826
List of supported messages	1827
Account Management list of supported messages	1827
Cash Management list of supported messages	1828
Collateral Management list of supported messages	1829
MX Application Headers list of supported messages	1829
Payments, Clearing and Settlement list of supported messages	1829
Payments Initiation list of supported messages	1830
Reference Data list of supported messages	1830
Securities Clearing list of supported messages	1830
Securities Events list of supported messages	1830
Securities Management list of supported messages	1831
Securities Settlement list of supported messages	1832
Securities Trading list of supported messages	1833
Trade Services Management list of supported messages	1833
Error reporting in the MX validation framework	1834
Adding maps to the MX validation framework	1835
How to use different levels of validation in the MX validation framework	1835
Debugging the MX validation framework	1835
Building and deploying SWIFT map systems	1836
Build Script Files	1836
Script Process Description	1836
Script parameters	1836
Actions performed by the build and deploy scripts	1836
Running the build and deployment scripts	1837
How to enable SFTP to deploy the pack	1837
Implementing SWIFT component in IBM Sterling B2B Integrator (Deprecated)	1837
Deploying the XML schemas to IBM Sterling B2B Integrator (Deprecated)	1838
Using JVC compliance checking on IBM Sterling B2B Integrator (Deprecated)	1838
Enabling JVC compliance checking on IBM Sterling B2B Integrator	1838
Deploying JVC compliance checking on IBM Sterling B2B Integrator	1839
Using MX validation for MX compliance checking on IBM Sterling B2B Integrator (Deprecated)	1839
Enabling MX compliance checking on IBM Sterling B2B Integrator	1839
Deploying MX compliance checking components	1840
SWIFT component examples	1840
Bank to XML example	1840
How to run the example in Design Studio	1840
How to run the example in Design Server	1841
How to download output files	1841
SWIFT to LMF example	1841
How to run the example in Design Studio	1842
How to run the example in Design Server	1842
SWIFT To LMF maps	1843
How to download output files	1841
SWIFT from LMF example	1843
How to run the example in Design Studio	1844
How to run the example in Design Server	1844
SWIFT from LMF maps	1845
How to download output files	1841
SWIFT MT Compliance example	1845
How to run the example in Design Studio	1846
How to run the example in Design Server	1846
How to download output files	1841
MT JVC Report example	1846
Configuration of SWIFT MT JVC Report example in Design Studio	1847
Additional configuration for MT JVC validation	1847
How to run the example in Design Studio	1847
Configuration of SWIFT MT JVC Report example in Design Server	1848
How to run the example in Design Server	1849
How to download output files	1849
MT950 Splitter Map example	1849
MT 950 Splitter Map example overview	1850
How to run the example in Design Studio	1850
How to run the example in Design Server	1850
How to download output files	1841
How to run mt950_split_v2	1851
MT Launcher Validation framework	1851
Deploying the Launcher Validation Framework	1852
Running the Launcher Validation Framework	1852
MT - MX translation maps example (Deprecated)	1852
How to run the example in Design Studio	1853

How to run the example in Design Server	1854
Building multiple maps using Deploy	1854
SWIFT MX Message Bundle and Unbundle map example using Design Server	1855
How to run the example in Design Studio	1855
How to run the example in Design Server	1855
How to download output files	1855
Example executable maps	1856
SWIFT MX validation example	1856
How to run the example in Design Studio	1856
How to run the example in Design Server	1856
How to download output files	1857
IBM Sterling B2B Integrator examples (Deprecated)	1857
IBM Sterling B2B Integrator examples overview	1857
Directory structure of the IBM Sterling B2B Integrator examples	1858
IBM Sterling B2B Integrator Generic Envelope service MT example	1858
IBM Sterling B2B Integrator Generic Deenvelope service MT example	1859
IBM Sterling B2B Integrator Generic Envelope service MX example	1859
IBM Sterling B2B Integrator Generic Deenvelope MX example	1859
Design considerations for the MT and MX Generic Envelope and Deenvelope examples	1859
IBM Sterling B2B Integrator data harness	1860
Thread safety	1860
Specific map and type tree settings	1860
Burst	1860
Audit settings	1860
Restart	1861
Custom MT type tree	1861
Building a custom MT type tree	1862
Deploying and executing the MT and MX Generic Envelope and Deenvelope examples	1862
Deploying the example maps to IBM Sterling B2B Integrator	1862
Deploying the map using Transformation Extender	1862
Deploying the map using IBM Sterling B2B Integrator	1863
Running the MT examples	1863
Deploying the example maps to IBM Sterling B2B Integrator	1863
Deploying maps from within IBM Transformation Extender	1863
Deploying maps using IBM Sterling B2B Integrator	1864
How to run MT examples from IBM Sterling B2B Integrator	1864
Running the MX examples	1864
Importing IBM Sterling B2B Integrator	1864
Deploying the map from within IBM Transformation Extender	1865
Deploying the maps using IBM Sterling B2B Integrator	1865
How to run the MX examples from IBM Sterling B2B Integrator	1865
Expected results	1865
Generic Deenvelope service MT example: expected results	1866
Generic Envelope service MT example: expected results	1866
Generic Deenvelope service MX example, expected results	1866
Generic Envelope service MX example: expected results	1866
Universal Confirmation examples	1866
Configuration of SWIFT Universal Confirmation examples in Design Studio	1866
How to run the Universal Confirmation API example in Design Studio	1867
How to run the Universal Confirmation CSV example in Design Studio	1867
How to run the Universal Confirmation FIN example in Design Studio	1868
Configuration of SWIFT Universal Confirmation examples in Design Server	1868
How to run the Universal Confirmation API example in Design Server	1869
How to download output files of Universal Confirmation API example	1869
How to run the Universal Confirmation CSV example in Design Server	1870
How to download output files of Universal Confirmation CSV example	1870
How to run the Universal Confirmation FIN example in Design Server	1871
How to download output files of Universal Confirmation FIN example	1871
Pack for Financial Payments Plus	
Financial Payments Plus standards	1871
Overview	1872
Glossary	1872
Components	1873
Examples	1876
CBPR+ Examples:	1877
CBPR+ Validation Examples:	1877
CBPR+ enhanced MX validation (using flows) Example	1877
What the example contains	1877
How to run the example	1880
Run the example using three different methods	1883
Run the example from the Design Server user interface	1883
Run the example using the Swagger interface	1883

Run the example using the flow command server process (flowcmdserver)	1884
How to customize the mxconfig.xml file	1884
CBPR+ enhanced MX validation (using maps) Example	1885
What the example contains	1885
Run the example in Design Studio	1887
Run the example in Design Server User Interface	1887
How to customize the mxconfig.xml file	1887
CBPR+ schema validation (using maps) Example	1887
What the example contains	1888
Run the example in Design Studio	1889
Run the example in Design Server User Interface	1889
CBPR+ Translation Examples:	1889
CBPR+ Translation Examples using flows:	1890
CBPR+ MX to MT (camt.029.001.09 to MT196/MT296) Translation Example	1890
What the example contains	1890
How to run the example	1891
Run the example using three different methods	1892
Run the example from the Design Server user interface	1892
Run the example using the Swagger interface	1892
Run the example using the flow command server process (flowcmdserver)	1893
CBPR+ MX to MT (camt.052.001.08 to MT942) Translation Example	1893
What the example contains	1893
How to run the example	1893
Run the example using three different methods	1895
Run the example from the Design Server user interface	1895
Run the example using the Swagger interface	1895
Run the example using the flow command server process (flowcmdserver)	1896
CBPR+ MX to MT (camt.053.001.08 to MT940) Translation Example	1896
What the example contains	1896
How to run the example	1896
Run the example using three different methods	1898
Run the example from the Design Server user interface	1898
Run the example using the Swagger interface	1898
Run the example using the flow command server process (flowcmdserver)	1899
CBPR+ MX to MT (camt.054.001.08 to MT900/MT910) Translation Example	1899
What the example contains	1899
How to run the example	1900
Run the example using three different methods	1901
Run the example from the Design Server user interface	1901
Run the example using the Swagger interface	1901
Run the example using the flow command server process (flowcmdserver)	1902
CBPR+ MX to MT (camt.056.001.08 to MT192/MT292) Translation Example	1902
What the example contains	1902
How to run the example	1902
Run the example using three different methods	1903
Run the example from the Design Server user interface	1904
Run the example using the Swagger interface	1904
Run the example using the flow command server process (flowcmdserver)	1904
CBPR+ MX to MT (camt.057.001.06 to MT210) Translation Example	1905
What the example contains	1905
How to run the example	1905
Run the example using three different methods	1906
Run the example from the Design Server user interface	1906
Run the example using the Swagger interface	1907
Run the example using the flow command server process (flowcmdserver)	1907
CBPR+ MX to MT (camt.058.001.08 to MT292) Translation Example	1907
What the example contains	1907
How to run the example	1908
Run the example using three different methods	1909
Run the example from the Design Server user interface	1909
Run the example using the Swagger interface	1909
Run the example using the flow command server process (flowcmdserver)	1910
CBPR+ MX to MT (camt.107.001.01 to MT110) Translation Example	1910
What the example contains	1910
How to run the example	1910
Run the example using three different methods	1912
Run the example from the Design Server user interface	1912
Run the example using the Swagger interface	1912
Run the example using the flow command server process (flowcmdserver)	1912
CBPR+ MX to MT (camt.108.001.01 to MT111) Translation Example	1913
What the example contains	1913
How to run the example	1913

Run the example using three different methods	1914
Run the example from the Design Server user interface	1914
Run the example using the Swagger interface	1915
Run the example using the flow command server process (flowcmdserver)	1915
CBPR+ MX to MT (camt.109.001.01 to MT112) Translation Example	1915
What the example contains	1915
How to run the example	1916
Run the example using three different methods	1917
Run the example from the Design Server user interface	1917
Run the example using the Swagger interface	1917
Run the example using the flow command server process (flowcmdserver)	1918
CBPR+ MX to MT (pacs.002.001.10 to MT199/MT299) Translation Example	1918
What the example contains	1918
How to run the example	1919
Run the example using three different methods	1920
Run the example from the Design Server user interface	1920
Run the example using the Swagger interface	1920
Run the example using the flow command server process (flowcmdserver)	1920
CBPR+ MX to MT (pacs.004.001.09 to MTnnnRETN) Translation Example	1921
What the example contains	1921
How to run the example	1921
Run the example using three different methods	1923
Run the example from the Design Server user interface	1923
Run the example using the Swagger interface	1923
Run the example using the flow command server process (flowcmdserver)	1923
CBPR+ MX to MT (pacs.008.001.08 to MT103) Translation Example	1924
What the example contains	1924
How to run the example	1924
Run the example using three different methods	1925
Run the example from the Design Server user interface	1926
Run the example using the Swagger interface	1926
Run the example using the flow command server process (flowcmdserver)	1926
CBPR+ MX to MT (pacs.009.001.08 to MT202) Translation Example	1926
What the example contains	1927
How to run the example	1927
Run the example using three different methods	1928
Run the example from the Design Server user interface	1928
Run the example using the Swagger interface	1928
Run the example using the flow command server process (flowcmdserver)	1929
CBPR+ MX to MT (pacs.009.001.08 to MT205) Translation Example	1929
What the example contains	1929
How to run the example	1930
Run the example using three different methods	1931
Run the example from the Design Server user interface	1931
Run the example using the Swagger interface	1931
Run the example using the flow command server process (flowcmdserver)	1932
CBPR+ MT to MX (MT900/MT910 to camt.054.001.08) Translation Example	1932
What the example contains	1932
How to run the example	1932
Run the example using three different methods	1933
Run the example from the Design Server user interface	1934
Run the example using the Swagger interface	1934
Run the example using the flow command server process (flowcmdserver)	1934
CBPR+ MT to MX (MT192/MT292 to camt.056.001.08) Translation Example	1935
What the example contains	1935
How to run the example	1935
Run the example using three different methods	1936
Run the example from the Design Server user interface	1936
Run the example using the Swagger interface	1937
Run the example using the flow command server process (flowcmdserver)	1937
CBPR+ MT to MX (MT196/MT296 to camt.029.001.09) Translation Example	1937
What the example contains	1937
How to run the example	1938
Run the example using three different methods	1939
Run the example from the Design Server user interface	1939
Run the example using the Swagger interface	1939
Run the example using the flow command server process (flowcmdserver)	1940
CBPR+ MT to MX (MT103 to pacs.008.001.08 or MT103 RETN to pacs.004.001.09) Translation Example	1940
What the example contains	1940
How to run the example	1941
Run the example using three different methods	1942
Run the example from the Design Server user interface	1942

Run the example using the Swagger interface	1942
Run the example using the flow command server process (flowcmdserver)	1942
CBPR+ MT to MX (MT103 SWIFTGo to pacs.008.001.08) Translation Example	1943
What the example contains	1943
How to run the example	1943
Run the example using three different methods	1944
Run the example from the Design Server user interface	1945
Run the example using the Swagger interface	1945
Run the example using the flow command server process (flowcmdserver)	1945
CBPR+ MT to MX (MT202 ADV/COR/COV to pacs.009.001.08 OR MT202 RETN to pacs.004.001.09) Translation Example	1946
What the example contains	1946
How to run the example	1946
Run the example using three different methods	1947
Run the example from the Design Server user interface	1948
Run the example using the Swagger interface	1948
Run the example using the flow command server process (flowcmdserver)	1948
CBPR+ MT to MX (MT205 COR/COV to pacs.009.001.08 OR MT205 RETN to pacs.004.001.09) Translation Example	1948
What the example contains	1949
How to run the example	1949
Run the example using three different methods	1950
Run the example from the Design Server user interface	1950
Run the example using the Swagger interface	1951
Run the example using the flow command server process (flowcmdserver)	1951
CBPR+ MT message validation Example	1951
What the example contains	1951
How to run the example	1952
Additional configuration for MT JVC validation	1953
Configuration files for Design Server	1953
Run the example using three different methods	1954
Run the example from the Design Server user interface	1954
Run the example using the Swagger interface	1954
Run the example using the flow command server process (flowcmdserver)	1955
CBPR+ Translation Examples using maps:	1955
CBPR+ MX to MT (camt.029.001.09 to MT196/MT296) Translation Example	1956
What the example contains	1956
How to run the example	1956
CBPR+ MX to MT (camt.052.001.08 to MT942) Translation Example	1957
What the example contains	1957
How to run the example	1957
CBPR+ MX to MT (camt.053.001.08 to MT940) Translation Example	1958
What the example contains	1958
How to run the example	1959
CBPR+ MX to MT (camt.054.001.08 to MT900/MT910) Translation Example	1960
What the example contains	1960
How to run the example	1960
CBPR+ MX to MT (camt.056.001.08 to MT192/MT292) Translation Example	1961
What the example contains	1961
How to run the example	1961
CBPR+ MX to MT (camt.057.001.06 to MT210) Translation Example	1962
What the example contains	1962
How to run the example	1963
CBPR+ MX to MT (camt.058.001.08 to MT292) Translation Example	1963
What the example contains	1963
How to run the example	1964
CBPR+ MX to MT (camt.107.001.01 to MT110) Translation Example	1964
What the example contains	1965
How to run the example	1965
CBPR+ MX to MT (camt.108.001.01 to MT111) Translation Example	1966
What the example contains	1966
How to run the example	1966
CBPR+ MX to MT (camt.109.001.01 to MT112) Translation Example	1967
What the example contains	1967
How to run the example	1967
CBPR+ MX to MT (pacs.002.001.10 to MT199/MT299) Translation Example	1968
What the example contains	1968
How to run the example	1968
CBPR+ MX to MT (pacs.008.001.08 SWIFT Go to MT103) Translation Example	1969
What the example contains	1969
How to run the example	1969
CBPR+ MX to MT (pacs.009.001.08 to MT202 (CORE/ADV/COVE)) Translation Example	1970
What the example contains	1970
How to run the example	1971

CBPR+ MX to MT (pacs.009.001.08 to MT205 (CORE/COVE) Translation Example	1971
What the example contains	1972
How to run the example	1972
CBPR+ MX to MT (pacs.004.001.09 to MT103 RETN/MT202 RETN/MT205 RETN) Translation Example	1973
What the example contains	1973
How to run the example	1973
CBPR+ MX to MT (pacs.008.001.08 to MT103) Translation Example	1974
What the example contains	1974
How to run the example	1974
CBPR+ MT to MX (MT900/MT910 to camt.054.001.08) Translation Example	1975
What the example contains	1975
How to run the example	1976
CBPR+ MT to MX (MT192/MT292 to camt.056.001.08) Translation Example	1976
What the example contains	1976
How to run the example	1977
CBPR+ MT to MX (MT196/MT296 to camt.029.001.09) Translation Example	1977
What the example contains	1978
How to run the example	1978
CBPR+ MT to MX (MT103_COR/MT103_STP to pacs.008.001.08 or MT103_COR RETN to pacs.004.001.09) Translation Example	1979
What the example contains	1979
How to run the example	1979
CBPR+ MT to MX (MT103 SWIFT Go to pacs.008.001.08) Translation Example	1980
What the example contains	1980
How to run the example	1980
CBPR+ MT to MX (MT202 (core/adv/cove) to pacs.009.001.08 (core/adv/cove) or MT202 (RETN) to pacs.004.001.09) Translation Example	1981
What the example contains	1981
How to run the example	1982
CBPR+ MT to MX (MT205 (core/cove) to pacs.009.001.08 (core/cove) or MT205 (RETN) to pacs.004.001.09) Translation Example	1982
What the example contains	1982
How to run the example	1983
CBPR+ MT to MX Translation (using TX maps) Example	1984
What the example contains	1984
Run the example in Design Studio	1985
Run the example in Design Server User Interface	1986
CBPR+ MX to MT Translation (using TX maps) Example	1987
What the example contains	1987
Run the example in Design Studio	1990
Run the example in Design Server User Interface	1991
CHAPS L4L/ENH Examples:	1992
CHAPS L4L Examples: (Deprecated)	1992
CHAPS L4L enhanced MX validation (using flows) Example	1992
What the example contains	1993
How to run the example	1994
Run the example using three different methods	1997
Run the example from the Design Server user interface	1997
Run the example using the Swagger interface	1997
Run the example using the flow command server process (flowcmdserver)	1998
How to customize the mxconfig.xml file	1884
CHAPS L4L enhanced MX validation (using maps) Example	1998
What the example contains	1999
Run the example in Design Studio	2000
Run the example in Design Server User Interface	2000
How to customize the mxconfig.xml file	1887
CHAPS L4L schema validation (using maps) Example	2001
What the example contains	2001
Run the example in Design Studio	2002
Run the example in Design Server User Interface	2002
CHAPS ENH Examples:	2002
CHAPS ENH enhanced MX validation (using flows) Example	2002
What the example contains	2003
How to run the example	2004
Run the example using three different methods	2007
Run the example from the Design Server user interface	2007
Run the example using the Swagger interface	2007
Run the example using the flow command server process (flowcmdserver)	2008
How to customize the mxconfig.xml file	1884
CHAPS ENH enhanced MX validation (using maps) Example	2008
What the example contains	2009
Run the example in Design Studio	2010
Run the example in Design Server User Interface	2010
How to customize the mxconfig.xml file	1887
CHAPS ENH schema validation (using maps) Example	2011

What the example contains _____	2011
Run the example in Design Studio _____	2012
Run the example in Design Server User Interface _____	2012
EBA EURO1/STEP1 Examples:	2012
EBA E1S1 enhanced MX validation (using Flow Server) Example	2012
What the example contains _____	2013
How to run the example _____	2014
Run the example using three different methods _____	2016
Run the example from the Design Server user interface _____	2017
Run the example using the Swagger interface _____	2017
Run the example using the flow command server process (flowcmdserver) _____	2017
How to customize the mxconfig.xml file	1884
EBA E1S1 enhanced MX validation (using maps) Example	2018
What the example contains _____	2018
Run the example in Design Studio _____	2020
Run the example in Design Server User Interface _____	2020
How to customize the mxconfig.xml file	1887
EBA E1S1 schema MX validation (using maps) Example	2020
What the example contains _____	2021
Run the example in Design Studio _____	2022
Run the example in Design Server User Interface _____	2022
NBOR ReGIS Examples:	2022
NBOR ReGIS schema validation (using Flow Server) example	2022
What the example contains _____	2022
How to run the example _____	2024
Run the example using three different methods _____	2025
Run the example from the Design Server user interface _____	2026
Run the example using the Swagger interface _____	2026
Run the example using the flow command server process (flowcmdserver) _____	2026
NBOR ReGIS schema validation (using maps) example	2027
What the example contains _____	2027
Run the example in Design Studio _____	2028
Run the example in Design Server User Interface _____	2028
TARGET2 CLM/RTGS/CoCo/Securities Examples:	2028
TARGET2 CLM Examples:	2029
T2 CLM enhanced MX validation (using Flow Server) example	2029
What the example contains _____	2029
How to run the example _____	2031
Run the example using three different methods _____	2033
Run the example from the Design Server user interface _____	2033
Run the example using the Swagger interface _____	2034
Run the example using the flow command server process (flowcmdserver) _____	2034
How to customize the mxconfig.xml file	1884
T2 CLM enhanced MX validation (using maps) example	2035
What the example contains _____	2035
Run the example in Design Studio _____	2037
Run the example in Design Server User Interface _____	2037
How to customize the mxconfig.xml file	1887
T2 CLM schema validation (using maps) example	2037
What the example contains _____	2038
Run the example in Design Studio _____	2039
Run the example in Design Server User Interface _____	2039
TARGET2 RTGS Examples:	2039
T2 RTGS enhanced MX validation (using Flow Server) example	2039
What the example contains _____	2040
How to run the example _____	2042
Run the example using three different methods _____	2044
Run the example from the Design Server user interface _____	2044
Run the example using the Swagger interface _____	2044
Run the example using the flow command server process (flowcmdserver) _____	2045
How to customize the mxconfig.xml file	1884
T2 RTGS enhanced MX validation (using maps) example	2046
What the example contains _____	2046
Run the example in Design Studio _____	2048
Run the example in Design Server User Interface _____	2048
How to customize the mxconfig.xml file	1887
T2 RTGS schema validation (using maps) example	2048
What the example contains _____	2049
Run the example in Design Studio _____	2050
Run the example in Design Server User Interface _____	2050
TARGET2 CoCo Examples:	2050
T2 CoCo enhanced MX validation (using Flow Server) example	2050

What the example contains	2051
How to run the example	2053
Run the example using three different methods	2055
Run the example from the Design Server user interface	2055
Run the example using the Swagger interface	2056
Run the example using the flow command server process (flowcmdserver)	2056
How to customize the mxconfig.xml file	1884
T2 CoCo enhanced MX validation (using maps) example	2057
What the example contains	2057
Run the example in Design Studio	2059
Run the example in Design Server User Interface	2059
How to customize the mxconfig.xml file	1887
T2 CoCo schema validation (using maps) example	2060
What the example contains	2060
Run the example in Design Studio	2061
Run the example in Design Server User Interface	2061
TARGET2 Securities Examples:	2062
T2 Securities (T2S) schema validation (using Flow Server) example	2062
What the example contains	2062
How to run the example (using t2_sec_validation_flow)	2064
Run the example using three different methods	2067
Run the example from the Design Server user interface	2067
Run the example using the Swagger interface	2067
Run the example using the flow command server process (flowcmdserver)	2067
How to run the example (using t2_sec_head002_validation_flow)	2068
Run the example using three different methods	2070
Run the example from the Design Server user interface	2070
Run the example using the Swagger interface	2070
Run the example using the flow command server process (flowcmdserver)	2071
T2 Securities (T2S) schema validation (using maps) example	2071
What the example contains	2071
Run the example in Design Studio	2074
Test with head.002.001.01 schema	2074
Run the example in Design Server User Interface	2074
Test with head.002.001.01 schema	2074
SWIFT GPI/UC Examples:	2074
SWIFT UC Examples:	2075
SWIFT UC enhanced MX validation (using Flow Server) example	2075
What the example contains	2075
How to run the example	2076
Run the example using three different methods	2077
Run the example from the Design Server user interface	2078
Run the example using the swagger interface	2078
Run the example using the flow command server process (flowcmdserver)	2078
How to customize the mxconfig.xml file	1884
SWIFT UC enhanced MX validation (using maps) example	2079
What the example contains	2079
Run the example in Design Studio	2080
Run the example in Design Server User Interface	2080
How to customize the mxconfig.xml file	1887
SWIFT UC schema MX validation (using maps) example	2081
What the example contains	2081
Run the example in Design Studio	2081
Run the example in Design Server User Interface	2081
SWIFT GPI Examples:	2082
SWIFT GPI enhanced MX validation (using Flow Server) example	2082
What the example contains	2082
How to run the example	2084
Run the example using three different methods	2086
Run the example from the Design Server user interface	2086
Run the example using the Swagger interface	2086
Run the example using the flow command server process (flowcmdserver)	2087
How to customize the mxconfig.xml file	1884
SWIFT GPI enhanced MX validation (using maps) example	2088
What the example contains	2088
Run the example in Design Studio	2089
Run the example in Design Server User Interface	2089
How to customize the mxconfig.xml file	1887
SWIFT GPI schema MX validation (using maps) example	2090
What the example contains	2090
Run the example in Design Studio	2091
Run the example in Design Server User Interface	2091

SIC (SIX Interbank Clearing) Validation Examples:	2091
SIC (SIX Interbank Clearing) Schema validation (using flows) Example	2091
What the example contains	2092
How to run the example	2093
Run the example using three different methods	2094
Run the example from the Design Server user interface	2094
Run the example using the Swagger interface	2095
Run the example using the flow command server process (flowcmdserver)	2095
SIC (SIX Interbank Clearing) Schema validation (using maps) Example	2095
What the example contains	2096
Run the example in Design Studio	2097
Run the example in Design Server User Interface	2097
Pack for Healthcare	
Overview	2097
Installation location	2097
General constraints	2098
Healthcare industry	2098
Healthcare information exchange participants	2098
Healthcare transactions and standards	2098
Healthcare electronic data interchange initiatives	2098
HIPAA legislation	2099
HIPAA transactions	2099
WEDI SNIP transaction compliance types	2099
HIPAA EDI component	2100
Objects included in the HIPAA EDI component	2100
Schemas	2101
Sample data	2101
Maps	2101
Utility modules	2101
HIPAA EDI schemas	2101
Overview	2101
What the schemas support	2102
General schema information	2102
HIPAA X12 schemas	2102
hipaa_x12 schema	2102
hipaa_x12_type_1 schema	2103
hipaa_x12_type_2 schema	2103
hipaa_x12_ruleless schema	2104
Center for Medicare and Medicaid Services (CMS) schemas	2104
cms_276_277_5010_flat schema	2104
cms_277ca_5010_flat schema	2104
cms_835_5010a1_flat schema	2104
cms_837i_5010a2_flat schema	2105
cms_837p_5010a1_flat schema	2105
Utility schema	2105
hipaa_x12_v5010x214_277ca schema	2105
ICD-10 support	2105
Replacing the ICD-10 compliant schemas	2105
Changing maps impacted in the compliance_check source map	2106
Changing maps impacted in the hipaa_x12_pass_through source map	2106
Changing other maps	2106
Maps	2107
Data transformation maps	2107
Institutional claims and coordination of benefits	2107
cms_837i_5010_flat_to_hipaa map	2107
hipaa_837p_5010_to_cms_flatmap map	2107
Professional claims and coordination of benefits maps	2107
cms_837p_5010_flat_to_hipaa map	2108
hipaa_837p_5010_to_cms_flatmap map	2107
Healthcare claims payments maps	2108
cms_835_5010_flat_to_hipaa map	2108
hipaa_835_5010_to_cms_flat map	2108
Healthcare claims status request and response maps	2108
hipaa_276_5010_to_cms_flat map	2109
cms_276_5010_flat_to_hipaa map	2109
hipaa_277_5010_to_cms_flat map	2109
cms_277_5010_flat_to_hipaa map:	2109
hipaa_277ca_5010_to_cms_flat map	2109
cms_277ca_5010_flat_to_hipaa map:	2109
Data validation maps	2109
Pass-through maps	2109
Source map: hipaa_x12_pass_through	2110

Source map: cms_276_5010_flat_pass_through	2110
Source map: cms_277_5010_flat_pass_through	2110
Source map: cms_277ca_5010_flat_pass_through	2110
Source map: cms_835_5010_flat_pass_through	2110
Source map: cms_837i_5010_flat_pass_through	2110
Source map: cms_837p_5010_flat_pass_through	2110
Compliance checking maps	2110
Sample data	2111
Overview of sample data	2111
Transaction data	2111
HIPAA X12	2111
CMS sample data	2111
Compliance check data	2111
HIPAA Compliance Checking	2112
WEDI SNIP transaction compliance types	2099
Type 1 EDI Standard Integrity Testing	2112
Type 2 HIPAA Implementation Guide Requirement Testing	2113
Type 3 HIPAA Balance Testing	2113
Type 4 HIPAA Inter-segment Situation Testing	2113
Type 5 HIPAA External Code Set Testing	2113
Type 6 HIPAA Line of Service Testing	2114
HIPAA operating rules	2114
Type 7 Companion Guide Testing	2114
Parameters for HIPAA processing	2114
HIPAA Data Compliance Parameters	2115
HIPAA Compliance Check Parameters	2120
HIPAA_Relaxed_T2_Structure_Check_Enabled details	2122
Relaxed structure check option	2122
R2 records	2123
Claim Level Rejection	2123
Assumptions	2123
Acknowledgements	2124
Configuration process	2124
Enable Claim Level Rejection	2124
Performance tuning and debugging	2124
Configurable Rules	2124
Enabling Rules with HIPAA DATA compliance	2125
Enabling Rules with HIPAA Compliance Check	2125
Configurable rules objects	2125
Data files	2126
configurable_rules_component_rules_only.dat	2126
configurable_rules_mini_qualifier.dat	2126
configurable_rules_type_5_mini_qualifier.dat	2127
minus.gif	2127
plus.gif	2127
qualifier_rules_report.css	2127
directory_paths.dat	2127
Schemas	2127
t3t4_utility schema	2127
Rule_less schema	2128
Schemas script file	2128
hipaa_x12_master_rule_stub.mts schema script file	2128
Map source file	2128
Description of executable maps	2128
Configurable Rules batch command files	2129
import_trees.bat	2129
configurable_rules.bat	2129
Compliance Check application qualifier file	2129
Executable map files	2129
Configurable rules reporting	2129
Validation results	2130
Modified rules	2130
Unchanged rules	2130
Enabled non-configured rules	2130
Configurable rules assumptions	2130
Basic steps for running the Configurable Rules application	2130
Instructions for running the Configurable rules application	2131
Type 5 Validation	2132
Type 5 Validation Methods	2132
Configuration	2133
Enabling Type 5 Validation and Rules	2133
Code list identifiers table	2133

Type 6 Validation	2136
Implementation	2136
Configuration	2137
Enabling Type 6 Validation and Rules	2137
Type 7 Validation	2137
Implementation	2137
Configuration	2138
Enabling Type 7 Validation and Rules	2138
HIPAA EDI component examples	2138
ICD-10	2139
ICD-10 example	2139
Insulating legacy systems	2139
Locating the ICD-10 example files	2140
Using the ICD-10 example files	2140
278N example	2140
Location of the 278N example files	2140
Using the 278N example	2140
HIX example	2141
Locating the HIX example files	2141
HIX example executable maps	2141
HIPAA to HIX example map	2141
HIX to HIX example map	2142
Using the HIX example files	2142
Preview Draft 80X0	2142
Location of Preview Draft 80X0 example files	2142
Schemas and JSON metadata	2143
Step-up and step-down maps	2143
Preview Draft 8010 example source maps	2143
X12N example	2144
Location of the X12N example files	2144
X12N schemas	2144
Using the X12N example files	2145
External Code List Utilities example	2148
Terminology for NoSQL databases.	2148
Locating	2149
Example components	2149
Configuration files	2149
Maps to load or delete code set information	2149
Maps to convert code set data	2149
Schemas	2149
Data Files	2149
Using the example	2150
Using the load or delete code set maps	2150
Using the maps to convert code set data	2150
Type 7 User Exit example	2150
Location	2150
Example components	2150
Example details	2150
Using the Type 7 user exit example	2151
ClaimToExposure tracking example	2151
Location	2151
ClaimToExposureTracking example maps	2151
Claim to exposure tracking map details	2152
X12N personal health records example	2152
Example location	2152
Example maps	2152
Performance tuning	2152
Duplicate processing	2153
Unnecessary processing	2153
Over reporting	2153
Acknowledgment errors	2153
AK304 segment level	2153
IK304 error code	2154
AK403 element level	2154
IK403 error code	2154
AK502 transaction set level	2155
AK905 functional group level	2155
TA105 interchange level	2155
TED01 - error reporting/acknowledgement process	2156
HL7 component	2156
HL7 schemas	2156
HL7 component schemas	2157

hl7_v2_x schemas	2157
hl7_v2_1 schema	2157
hl7_v2_2 schema	2157
hl7_v2_3 schema	2157
hl7_v2_4 schema	2158
hl7_v2_5 schema	2158
hl7_v2_5_1 schema	2158
hl7_v2_6 schema	2158
hl7_v2_7 schema	2158
Using the Pack for HL7 schemas	2158
Schema data definitions	2158
Message structure	2159
Logical grouping of fields	2159
HL7 validation	2159
Running the HL7 validation utility	2160
HL7 component examples	2160
NCPDP-SCRIPT / HL7 mapping example	2160
Location of HL7/NCPDP-SCRIPT mapping files	2160
HL7 / NCPDP-SCRIPT mapping schemas	2161
NCPDP-SCRIPT / HL7 example maps	2161
FHIR example	2161
Location of the FHIR example files	2161
FHIR example schemas	2161
FHIR example maps	2161
Running the example for FHIR JSON format	2162
Running the example in FHIR XML format	2162
Running the FHIR PAS bundle example map	2162
FHIR Example - PAS Bundle with JSON Schema	2162
Location of the PAS Bundle with JSON example files	2162
FHIR Validation Example	2163
Location of the FHIR Validation example files	2163
HL7 MLLP Example	2163
Location of the MLLP example files	2163
NCPDP component	2163
Objects included in the pack	2164
NCPDP schemas	2164
Sample data	2164
Maps	2164
NCPDP maps	2164
Sample Data	2164
Transaction data	2164
NCPDP sample data	2165
NCPDP component examples	2165
NCPDP SCRIPT example	2165
Location of the Script example files	2166
Release indicator	2166
XML validation	2166
Using SCRIPT examples on UNIX platforms	2167
Using SCRIPT example files in the z/OS environment	2167
NCPDP SCRIPT	2167
ncpdp_script_v10_6_to_xml.mms	2167
ncpdp_xml_to_script_v10_6.mms	2167
ncpdp_script_v10_6_passthru.mms	2167
ncpdp_xml_v10_6_passthru.mms	2168
ncpdp_script_v10_6_xml_v201707.mms	2168
NCPDP PACDR example	2168
NCPDP SCRIPT schemas	2168
SCRIPT schema structure	2168
Data element type names	2168
Structure of XML schemas	2169
SCRIPT validation	2169
NCPDP validation	2169
Running the NCPDP validation utility	2169

Pack for Supply Chain EDI

Pack for Supply Chain EDI overview	2170
About Electronic Data Interchange (EDI)	2170
Example files	2170
Installation and uninstallation	2171
What is an EDI schema?	2171
ANSI X12 data	2171
Envelopes	2171
Transaction sets	2171
Segments	2172

EDIFACT data	2172
TRADACOMS data	2172
EDI version release schemas	2172
Types in EDI schemas	2173
Modifying EDI schemas	2173
Analysis of EDI schemas	2173
Summary of errors	2173
X12 transaction sets that did not pass analysis	2174
Delete X12 transaction sets in error if not used	2174
ANSI2003 the #830 transaction set	2175
ANSI2040 the #830 transaction	2175
ANSI3010 the #830 transaction	2175
ANSI3020 the #110 transaction	2175
ANSI3020 the #838 transaction	2175
ANSI3060, ANSI3070, ANSI4010, and ANSI4020 the #304 transaction	2176
Mapping purchase orders to invoices	2176
By line item	2176
Purchase order	2176
Invoices to be received	2177
ByStore map	2177
ByPO map	2178
ByLine map	2178
Editobol map source file	2178
Mapping inbound EDI #204 transaction sets	2178
EachBOL functional map	2178
MakeNameSet functional map	2179
EachDetail functional map	2179
X12 schemas	2179
About the X12 Standard	2179
Installation	2179
X12 schema details	2179
Overview	2179
ANSI	2180
Interchange	2180
Transmission	2180
ANSI category	2180
Control subtree	2180
Segment group	2181
ISA segment	2181
Interchange acknowledgment	2181
Interchange syntax extension	2181
Control elements	2182
Variant category	2182
Delimiter types	2182
Funct'IGroup subtree	2182
Version-specific subtree	2183
Elements	2183
XComposites and MComposites	2183
Segments	2183
Sets - transactions sets, loops, and blocks	2184
Transaction sets are organized under categories	2184
Loops and blocks	2184
Interchange category	2184
Restart attribute on functional group	2185
Component rule on IEA segment	2185
Transmission category	2185
Restart attribute on an interchange	2185
Mapping ANSI data	2185
Input card - receiving ANSI data from multiple partners	2186
Output card - sending ANSI data to multiple partners	2186
X12 examples	2186
asntoedi example	2187
asntoedi example notes	2187
How to run the asntoedi example	2188
editobol example	2188
editobol example notes	2188
How to run the editobol example	2188
hlLoop example	2189
Understanding Hierarchical Loops	2189
Hierarchical Design Rules	2189
Example of Record Oriented Data	2190
Example Diagram of a Hierarchy	2190
Customizing ITX Standard for Hierarchical Loop level	2190
Mapping HL levels to the output	2191

sqd example	2191
sqd example notes	2191
How to run the sqd example	2191
xmlpo example	2192
xmlpo example notes	2192
How to run the xmlpo example	2192
EDIFACT schemas	2192
About the EDIFACT standard	2192
Installation	2193
EDIFACT schema details	2193
Overview	2193
EDIFACT category	2194
Interchange	2194
Transmission	2194
EDIFACT category details	2194
Control subtree	2194
Envelope types	2195
UNA service string	2195
EDIFACT interchange envelope UNB and UNZ	2195
Functional group envelope UNG and UNE	2195
Message envelopes - UNH and UNT	2195
Element category	2195
Delimiter category	2195
Composite category	2196
Group subtypes	2196
Msg subtype	2196
Version specific subtype	2196
Composite category	2196
Viewing the Element category's full description	2196
Segment category	2197
Set category	2197
Groups of segments	2197
Interchange category	2197
Restart attribute on functional group	2198
Component rule on UNZ segment	2198
Transmission category	2198
Restart attribute on an interchange	2198
Mapping EDIFACT data	2198
Input card - EDIFACT data from multiple trading partners	2198
Output card - sending EDIFACT data to multiple partners	2199
EDIFACT examples	2199
arrival example	2200
arrival example notes	2200
How to run the arrival example	2201
edf2xml example	2201
edf2xml example notes	2201
How to run the edf2xml example	2201
ediblockchain example	2202
Understanding the ediBlockChain example	2202
The structure of the example:	2202
What the ediblockchain folder contains	2203
Quick execution steps:	2203
Packages example	2203
What the map source of packages example contains	2204
Security example	2204
What the map source of security example contains	2204
vermas example	2204
vermas example notes	2204
How to run the vermas example	2205
EDI Compliance Checking	
Compliance check Overview	2205
Validation	2205
X12 Validation considerations	2205
Character Encoding Processing	2206
Validation overview	2206
Validation checklist	2207
Run the EDI compliance check maps	2207
Run the X12/RAIL compliance check maps	2207
To compile all the maps using Design Studio	2208
To compile all the maps using Design Server	2208
To compile select maps using Design Studio	2208
To compile select maps using Design Server	2208
Where is my output?	2208

Rest Container	2209
Run the EDIFACT/IATA compliance check maps	2210
To compile all the maps Design Studio	2210
To compile all the maps Design Server	2210
To compile select maps Design Studio	2211
To compile select maps Design Server	2211
Where is my output?	2211
Rest Container	2212
Configuration	2212
Input card details	2213
Card 1: EDI data	2213
Card 2: unique ID	2213
Card 3: configuration information	2213
Card 4: duplicate interchange flag	2214
Card 5: table data	2214
Card 6: message code list	2214
Card 7: structure detail file	2214
Card 8: system information file	2214
exitlib	2214
Card 9: trace flag	2215
System information file	2215
Configuration information details	2215
RuntimeInfo	2216
CardOutput	2216
FileOutput	2217
FileDirectory	2217
QueueOutput	2217
QueueServerClient	2217
QueueManagerName	2218
QueueNames	2218
WireOutput	2218
CreateRejectLog	2218
JsonReport	2218
MaxNumErrors	2219
WorkSpace	2219
SkipTTValidation	2219
ISXcontrol - X12	2219
IgnoreISX	2220
ReleaseCharacter	2220
CharacterEncoding	2220
CodeListVersionOverride	2220
PartnerInfo	2220
GenerateTA1 - X12, RAIL only	2221
AckInfo	2221
AckType	2221
AckLevel	2221
ValidateContrl	2221
AckVersion - X12, RAIL only	2221
CheckDuplicateGroupControl	2221
CheckDuplicateTrxControl	2222
Authority - X12, RAIL only	2222
Validate	2222
AuthSecInfo	2222
Security - X12, RAIL only	2222
Validate	2223
AuthSecInfo	2223
RecipRef - EDIFACT only	2223
Validate	2223
RecipRefInfo	2223
Deploy compliance check maps	2223
Map deployment procedure	2223
z/OS Implementation	2224
Installing the shared exit library component on z/OS	2225
Installing the shared exit library component on UNIX	2226
File naming conventions	2226
X12 and RAIL file naming conventions	2226
Valid, accepted transactions	2227
Invalid, rejected data	2227
Generated, outbound, functional acknowledgments	2227
Generated, outbound, TA1 acknowledgments	2227
Inbound, received, functional acknowledgments	2227
Inbound, received, TA1 acknowledgments	2227

EDIFACT file naming conventions	2228
Valid, accepted messages	2228
Invalid, rejected data	2228
Generated, outbound, CONTRL acknowledgments	2228
Inbound, received, CONTRL acknowledgments	2228
Performance tuning	2229
Override schema validation maps	2229
Override segment detail validation maps	2229
Debugging	2229
Compliance check logs	2230
Framework log	2230
Troubleshooting tips	2230
Pack for Supply Chain EDI samples	2231
Trading Manager	
Trading Manager Overview	2231
Installation and Configuration	2232
Overview	2232
About the Trading Manager Environment	2232
Partner Manager	2232
Message Manager	2232
Design Studio	2233
Integration Flow Designer	2233
Launcher	2233
Resource Adapters	2233
IBM WebSphere Transformation Extender Packs for EDI	2233
IBM WebSphere Transformation Extender with Launcher	2233
Installation requirements	2233
Minimum system requirements	2233
System requirements for Partner Manager	2234
UNIX platform	2234
Prerequisites	2234
Installation checklists	2234
Launcher	2234
Installing Trading Manager	2234
Determining the Trading Manager configuration	2234
Installing Trading Manager on a single workstation	2235
Installing Trading Manager on separate workstations	2235
Installing only Partner Manager	2235
Installation checklists	2235
Installing components	2235
Installing Trading Manager on Windows	2235
Installing Message Manager components of Trading Manager on UNIX	2236
Removing Trading Manager components	2236
Using Trading Manager with HIPAA	2236
Configuring Trading Manager	2236
Before you begin	2236
SQL scripts installed by Trading Manager	2237
Creating the Trading Manager Database	2237
Using an Access database	2237
Creating a Microsoft SQL Server Database	2237
Creating an Oracle Database	2238
Creating a Sybase Database	2238
Creating an IBM DB2 database	2239
IBM DB2 stored procedures	2239
Installing DB2 stored procedures	2240
Copying drop statements	2240
Determining the ODBC driver	2240
Defining an ODBC data source name	2240
Resolving ODBC difficulties	2241
Configuring Partner Manager	2241
Import structure validation	2241
Installing and using Message Manager	2242
Installing Message Manager	2242
Where to install Message Manager	2242
Client workstation installation	2242
IBM WebSphere Transformation Extender Launcher installation	2242
General configuration and setup	2242
Starting Message Manager	2243
Message Manager deployment tasks	2243
Step 1. Define the server types in the msg_mgr.msd file	2243
Step 2. Modify the msg_mgr.msd default path name settings	2244
Changing design time, run time paths	2244

Changing the .mrn path names	2244
Step 3. Deploying Message Manager on UNIX	2245
Step 4. Create a configuration file	2245
Step 5. Modify the E-mail alert settings	2245
Step 6. Configure the Launcher service startup parameters	2245
Step 7. Modify the Message Manager database/query file	2246
Step 8. Identify the Trading Manager database	2246
Identify the Trading Manager database connection	2247
Step 9. Define Partners, Post Offices and Trade Links	2247
Step 10. Run the EDI Wizard	2247
Step 11. Enable stampandsortFTP, stampandsortVAN, and stampandsortEmail maps	2247
Step 12. Modify Eemail.mtt to support proprietary data	2247
Step 13. Add application maps and link to Message Manager	2247
Adding a Message Manager application map	2248
Configuring Launcher settings	2248
Step 14. Use the deploy facility in the Integration Flow Designer	2248
Using the IFD deploy facility	2248
Step 15. Build and copy application and communication maps	2249
Generate the Launcher control file for an application system	2249
Step 16. Start the Launcher service	2249
Launcher Manager tools	2250
Using Message Manager	2250
Message Manager subsystems	2250
Archive subsystem	2250
EDIFACT subsystem	2251
EDIFACT inbound subsystem	2251
edifact_in_alert map	2251
EDIFACT outbound subsystem	2251
Send problem data alerts map	2251
StampAndSort subsystem	2251
ThreadID definition	2252
Inbound data archiving	2252
stampandsortFTP/VAN/Email maps	2252
Creating multiple stampandsortFTP maps	2253
stampandsortfile map	2253
Get file post office and put file post office	2253
stampandsortack map	2254
Resend map	2254
TRADACOMS subsystem	2254
TRADACOMS inbound subsystem	2254
TRADACOMS_in_alert_data map	2254
TRADACOMS outbound subsystem	2254
alert_data map	2254
Value Added Tax (VAT) reports	2254
Update subsystem	2255
export map	2255
dbalert map	2255
X12 subsystem	2255
X12 inbound subsystem	2256
sortx12 map	2256
X12 FA inbound map	2256
X12_In_Alert_Data map	2256
X12 outbound subsystem	2256
x12_out_alert map	2256
Support for proprietary data types	2256
Inbound data flow	2257
Adding proprietary data types	2257
Using alias files in maps	2257
Defining alias name and user defined fields	2258
Making the alias data available to application maps	2258
Choosing the alias file to use	2258
Alias files type tree (Alias.mtt)	2258
Creating user type trees	2258
Checklists	2259
Server installation checklist for Microsoft SQL Server	2259
Client installation checklist for Microsoft SQL Server	2259
Server installation checklist for Oracle database	2260
Client installation checklist for Oracle Server	2260
Server installation checklist for Sybase	2260
Client installation checklist for Sybase	2260
Server installation checklist for IBM DB2 database	2261
Client installation checklist for IBM DB2 Server	2261

Implementing custom X12 versions	2261
Overview	2261
Type trees	2262
Partner Manager	2262
Message Manager	2262
Creating structure records	2262
TPEC to Trading Manager conversion	2263
TPECAddressBook.xml file	2263
Internal partners	2263
External partners	2263
TPECPPostOffices.xml file	2263
TPECTradelinks.xml file	2264
Inbound functional groups:	2264
Outbound functional groups	2264
TPEC2CMConfig.txt file	2264
Running the TPEC to Trading Manager	2265
Setting up the TPEC2CM conversion map	2266
Running the "ConvertTPEC2CM" executable map	2266
TPEC control number conversion	2266
Running the Partner Manager XML Autoload Utility	2267
Warnings and error messages	2267
TPEC header trailer plug-in for Trading Manager	2268
TPEC plug-in requirements	2268
Planning the conversion	2268
TPEC installation instructions	2268
Description of mailbox_bundle components	2269
mailbox.conf	2269
config.mtt	2269
poheader.mdq	2269
potree.mtt	2269
Partner Manager	2269
Overview	2270
Uses for Partner Manager	2270
Theory of operation	2270
Trade Links and Partner Manager	2270
Partner Manager interface	2271
Navigators	2271
Toolbar	2271
Form area	2271
Audit Info button	2271
Getting started	2271
Starting Partner Manager	2272
Starting Partner Manager from the Start menu	2272
Starting Partner Manager using a database-specific desktop icon	2272
System requirements	2272
Database configuration	2272
System Data Source Name (DSN)	2273
Address books	2273
Trading partner and application partner relationships	2273
One trading partner with one application partner	2273
Many trading partners with one or more application partners	2273
One trading partner with many application partners	2273
Trading partners and application partners	2274
Internal address book	2274
External address book	2274
Internal trading partner and application partner	2274
Configuring internal and external address books	2274
Adding X12 internal trading partners	2275
Trading Partner tab	2275
ISA Options tab	2275
Application Partner tab	2276
Adding X12 external trading partners	2276
Trading Partner tab - External X12	2276
ISA Options tab	2277
TA1 Post Offices tab	2277
Authorization/Security tab	2277
Application Partner tab	2277
Adding Internal EDIFACT Trading Partners	2278
Identification tab	2278
Syntax/Separators tab	2278
Application Partner tab	2279
Adding External EDIFACT Trading Partners	2279

Identification tab	2279
Syntax/Separators tab	2280
Application Partner tab	2280
Adding Internal TRADACOMS Trading Partners	2280
Trading Partner tab	2280
Application Partner tab	2281
Adding External TRADACOMS Trading Partners	2281
Trading Partner tab	2281
Editing a Trading Partner	2281
Copying a Trading Partner	2282
Deleting a Trading Partner	2282
Application partners	2282
Adding an X12 Application Partner	2283
Add X12 Application Partner Window fields	2283
Adding an EDIFACT Application Partner	2283
Add EDIFACT Application Partner window fields	2284
Adding TRADACOMS Application Partners	2284
Add TRADACOMS Application Partner window fields	2284
Editing an Application Partner	2285
Deleting an Application Partner	2285
Managing Administrative Contacts	2285
Adding a Contact	2286
E-Mail Check box and Field	2286
Monitor Directory Check box and Field	2286
Editing a Contact	2286
Deleting a Contact	2287
Post Offices	2287
Overview	2287
File Post Offices	2288
E-Mail post offices	2288
FTP Post Offices	2288
Examples of FTP commands	2288
VAN post offices	2288
HTTP post offices	2288
JMS post offices	2289
MessageQ client post offices	2289
MessageQ server post offices	2289
MQSeries client post offices	2289
MQSeries server post offices	2289
MSMQ post offices	2289
OracleAQ post offices	2289
Viewing Post Offices	2289
Adding Post Offices	2290
Adding a File Post Office	2290
Copying Post Offices	2291
Editing Post Offices	2291
Deleting Post Offices	2291
VAN Post Offices Overview	2291
Supported VAN Types	2292
Adding a VAN Post Office	2292
Configuring a VAN Post Office	2292
VAN source configuration	2293
Specifying options for the Monitor File	2293
Other VAN source configuration options	2293
Trade links	2293
Trade Links Overview	2294
Establishing trade links	2294
Exporting trade links	2294
Inbound trade links	2294
X12 Inbound Trade Links	2295
Adding an X12 Inbound Trade Link	2295
Editing an X12 Inbound Trade Link	2295
Copying an X12 Inbound Trade Link	2296
Deleting an X12 Inbound Trade Link	2296
EDIFACT Inbound Trade Links	2296
Adding an EDIFACT Inbound Trade Link	2297
Editing an EDIFACT Inbound Trade Link	2297
Copying an EDIFACT Inbound Trade Link	2297
Deleting an EDIFACT Inbound Trade Link	2298
TRADACOMS Inbound Trade Links	2298
Adding a TRADACOMS Inbound Trade Link	2298
Editing TRADACOMS Inbound Trade Links	2299

Copying TRADACOMS Inbound Trade Links	2299
Deleting TRADACOMS Inbound Trade Links	2300
Inbound trade link applications	2300
X12 Inbound Trade Link Applications	2300
Adding an X12 Inbound Trade Link Application	2300
Route only option	2301
Editing an X12 Inbound Trade Link Application	2301
Editing a HIPAA X12 document	2302
Deleting an X12 Inbound Trade Link Application	2302
EDIFACT Inbound Trade Link Applications	2303
Adding an EDIFACT Inbound Trade Link Application	2303
Editing an EDIFACT Inbound Trade Link Application	2303
Deleting an EDIFACT Inbound Trade Link Application	2304
TRADACOMS Inbound Trade Link Applications	2304
Adding a TRADACOMS Inbound Trade Link Application	2304
Editing a TRADACOMS Inbound Trade Link Application	2305
Deleting a TRADACOMS Inbound Trade Link Application	2305
Outbound trade links	2305
X12 Outbound Trade Links	2305
Adding an X12 Outbound Trade Link	2306
Editing an X12 Outbound Trade Link	2306
Copying an X12 Outbound Trade Link	2307
Deleting an X12 Outbound Trade Link	2307
EDIFACT Outbound Trade Links	2307
Adding an EDIFACT Outbound Trade Link	2307
Editing an EDIFACT Outbound Trade Link	2308
Copying an EDIFACT Trade Link	2308
Deleting an EDIFACT Outbound Trade Link	2309
Adding a TRADACOMS Outbound Trade Link	2309
Editing TRADACOMS Outbound Trade Links	2310
Copying TRADACOMS Outbound Trade Links	2310
Deleting TRADACOMS Trade Links	2311
Adding an X12 Outbound Trade Links Application	2311
Editing an X12 Outbound Trade Link Application	2311
Deleting an X12 Outbound Trade Link Application	2312
EDIFACT Outbound Trade Links Applications	2312
Adding an EDIFACT Outbound Trade Links Application	2312
Editing an EDIFACT Outbound Trade Link Application	2312
Deleting an EDIFACT Outbound Trade Link Application	2313
TRADACOMS Outbound Trade Link Applications	2313
Adding a TRADACOMS Outbound Trade Link Application	2313
Editing a TRADACOMS Outbound Trade Link Application	2314
Deleting a TRADACOMS Outbound Trade Link Application	2314
ISA and GS Numbering Schemes	2314
Searching for a trade link	2315
Searching for an X12 Trade Link	2315
Searching for an EDIFACT Trade Link	2316
Searching for a TRADACOMS Trade Link	2316
Automatic acknowledgement creation	2317
Outbound CONTRL trade link	2317
Creating a control number map	2317
Exchanging information with Message Manager	2318
Map requirements and features	2318
Example: increments other than 1	2318
Example of transaction set control number by date	2318
Reports	2318
Traffic reports	2319
Drill down for details	2319
View Interchanges in Reports	2320
View Functional Groups in Reports	2320
Display detail in reports	2320
Display Transactions in Reports	2320
Display Segment Errors in Reports	2320
Display Data Types	2321
Report specific search	2321
Specify a date range	2321
Report status icon colors and codes	2321
Partner Manager log report	2323
Source of Partner Manager log message	2323
Alerts report	2323
Resend requests	2323
Transmission report	2323

Interchange report	2324
Drill down tips	2324
Specific trading partner for the interchange report	2324
Specific status codes for the interchange report	2325
Edit and/or resend an interchange	2325
Resend an interchange Immediately	2325
Functional group report	2326
Jumping to X12 997	2326
X12 acknowledgement reporting options	2326
Manual override of acknowledgement settings	2326
Functional group view	2327
Message/transaction report	2327
Viewing 997 content	2328
Jumping to a functional group	2328
X12 TA1 status reports	2328
Additional reports overview	2329
Running additional reports	2329
Printing Partner Manager reports	2329
Exporting reports	2329
Exporting traffic reports	2329
Default traffic report export settings	2330
Value Added Tax (VAT) reports	2330
Partner Manager editor	2331
Resending E-commerce data	2331
Resend ThreadID	2331
Utilities	2331
Configuration	2332
Database settings	2332
Database selection	2332
Creating a new database profile	2333
Importing a database profile	2333
Exporting a database profile	2333
Editing an existing database profile	2334
Deleting an existing database profile	2334
Selecting a different database profile	2334
Display options	2335
Specifying processing options	2335
Specifying transmission data options	2335
Change the look and feel of the display	2336
Message Manager configuration maintenance	2336
Optimization tab	2336
Mapping Options tab	2337
Database tab	2337
Acknowledgements tab	2338
Archive Options tab	2338
Traffic batch archiving tab	2338
Alerts tab	2339
Post Office Options tab	2339
Post Office search and replace	2339
Post Office types	2340
Marking a Post Office type inactive/active	2340
Adding a Post Office type	2340
Deleting a Post Office type	2340
Server directory configuration	2341
Standards	2341
Adding an X12 standard version	2342
Deleting an X12 standard version	2342
Adding X12 interchange standards or ID qualifiers	2342
Adding an X12 hex value	2343
X12 Structure Validation:Import	2343
X12 Structure Validation:Maintenance	2343
Deleting X12 interchange standards or ID qualifiers	2344
Selecting HIPAA institutions	2344
Importing HIPAA type qualifiers	2344
Adding an EDIFACT standard version	2344
Adding an EDIFACT partner ID code qualifier	2345
Adding an EDIFACT functional ID or message type	2345
Adding an EDIFACT hex value	2345
Deleting an EDIFACT standard	2346
Adding and deleting TRADACOMS standards	2346
Adding a TRADACOMS document type	2346
Editing a TRADACOMS document type	2347

Deleting a TRADACOMS document type	2347
Field defaults	2347
Deleting defaults	2347
Database utilities	2348
Merge	2348
Details on merging trade links	2348
Post Office and Trading Partner merge details	2348
Purge	2349
Purging traffic or transmission data	2349
Archiving batch traffic	2349
Folders	2350
Adding a subfolder	2350
Renaming a subfolder	2350
Moving a subfolder	2350
Deleting a Subfolder	2351
Security	2351
Changing passwords	2351
Preventing easily guessed passwords	2351
Changing your password	2351
Password synchronization	2352
Security parameters that impact passwords	2352
Password synchronization run-time errors/messages	2352
Security groups	2353
Adding a new security group	2353
Editing security group settings	2353
Deleting a security group	2353
Security parameters	2354
User maintenance	2354
Adding a user to a security group	2355
Editing a user	2355
Deleting a user	2355
User login with security	2356
Prior version password remover	2356
EDI Wizard	2356
EDI standard type trees	2356
Running the EDI Wizard	2357
Where the Type Trees are located	2357
When the EDI Wizard cannot locate a type tree	2357
When more than one type tree is found for a version	2357
Database copy utility	2358
Copying databases	2358
Database scripts	2358
*drop scripts	2358
*inst scripts	2358
*load scripts	2359
Managing traffic data	2359
Purging current traffic or online archive data	2359
Check the traffic table report count	2360
Archiving traffic data	2360
Archiving data from the current traffic tables to the online archive tables	2360
Moving data from the current traffic tables to a zip file	2361
Moving data from the online archive tables to a zip file	2361
Restoring traffic data from a zip file	2361
Performance Tuning and Debugging	2362
Overview	2362
Performance related considerations	2362
CPU and memory requirements	2362
Analyze transmission sizes	2363
Logical vs. physical implementations	2363
Interfacing with non-IBM WebSphere Transformation Extender systems	2363
Avoiding resource pending configurations	2363
Performance checklist	2363
Message Manager checklist	2363
External X12 partner checklist	2364
Inbound X12 trade link checklist	2364
Outbound X12 trade link checklist	2364
Small transmission size checklist	2364
Large transmission size checklist	2365
Tuning the IBM WebSphere Transformation Extender	2365
Turning on the Launcher trace file	2365
Updating the max threads parameter	2365
Solving IBM Transformation Extender interface problems	2365

Running multiple Launchers	2366
Configuring Partner Manager	2366
Overview	2366
Configuring Message Manager	2366
Opening the Message Manager Configuration Window	2366
Use stored procedures option	2366
TSV sort option	2367
Optional X12/EDIFACT validation	2367
TA1 control option	2367
Store 997 content option	2367
Ignore missing trade links option	2368
Archive options	2368
Options for external X12 partners	2369
Opening the Edit X12 Trading Partner Window	2369
Duplicate control option	2369
Authorization and security validation	2369
Options for inbound X12 trade links	2370
Opening the X12 Inbound Trade Links Window	2370
Routing options (inbound)	2370
Acknowledgments option	2370
HIPAA validation option	2371
Options for outbound X12 trade links	2371
Opening the X12 Outbound Trade Links Window	2371
Routing options (outbound)	2371
Outbound 997's	2371
Analyzing small transmissions	2371
Changing stampandsort options	2372
Changing sendx12 options	2372
Work file option for small transmissions	2372
Analyzing large transmissions	2372
stampandsort and sendx12	2373
Maximum FG validation size	2373
Work file option	2373
Select maximum reject file size	2373
X12 outbound PO packaging option	2374
Debugging Trading Manager	2374
Deployment errors and failures	2374
Startup errors	2374
Running Message Manager	2375
XML Autoload Utility	2375
Overview	2375
Add, change, or delete	2376
Embedded security	2376
Using the XML Autoload Utility	2376
Starting the XML Autoload Utility	2376
Utilities	2377
Edit XML file	2377
Run Options	2377
Run Load	2377
Validate only	2377
Update options	2377
Allow add	2377
Allow change	2377
Allow delete	2378
Change or delete error processing options	2378
Remember database user and password	2378
Data exceeding maximum length	2378
Report options	2378
Detailed	2378
Summary	2378
Print report	2378
Unattended operation	2379
Generating a configuration file	2379
Command line flags	2379
Running unattended	2380
Debugging a background task	2380
Configuration and command line parameters	2380
Examples	2381
Defining tables and fields	2381
Table and Field Types	2381
Post offices	2382
Post Offices (Tag: PostOffices)	2382

Post Offices Delete (Tag: PostOfficesDelete)	2382
Post Offices Change (Tag: PostOfficesChange)	2382
Contacts	2383
Contacts (Tag: Contacts)	2383
X12 Trading Partners	2383
X12 Trading Partners (Tag: X12TradingPartners)	2384
X12 Application Partners (Tag: X12AppPartners)	2385
X12 Application Partners Delete (Tag: X12AppPartnersDelete)	2385
X12 Application Partners Change: (Tag X12AppPartnersAddChange)	2385
X12 Trading Partners Change(Tag: X12TradingPartnersChange)	2386
X12 Trading Partners Delete (Tag: X12TradingPartnersDelete)	2386
Automatic creation of X12 acknowledgement trade links	2387
Inbound/outbound trade link acknowledgments	2387
Activating the feature	2387
X12 inbound trade links	2388
X12 Known Inbound Trade Link: (Tag: X12IBTradeLink)	2388
X12 Unknown Inbound Trade Link (Tag: X12Unknown_Link)	2389
X12 Unknown Inbound Trade Link Change (Tag: X12Unknown_LinkChange)	2389
X12 Inbound Trade Link Applications (Tag: X12InboundGroups)	2389
HIPAA validation type description	2390
X12 Known Inbound Trade Link Change (Tag: X12IBTradeLinkChange)	2390
X12 Inbound Trade Links Applications Add, Change, Delete (Tag: X12InboundGroupsAddChangeDelete)	2391
X12 Known Inbound Trade Link Delete (Tag: X12IBTradeLinkDelete)	2392
X12 Unknown Inbound Trade Link Delete (Tag: X12Unknown_LinkDelete)	2392
X12 outbound trade links	2392
X12 Outbound Trading Partner Trade Links (Tag: X12OBTPLink)	2393
X12 Outbound Application Partner Trade Links (Tag: X12OBAPartnerLnk)	2394
X12 Outbound Applications Trade Links (Tag: X12OBAppLink)	2394
X12 Outbound Trading Partner Trade Links Change (Tag: X12OBTPLinkChange)	2395
X12 Outbound Application Partner Trade Links Change (Tag: X12OBAPartnerLnkChange)	2396
X12 Outbound Applications Trade Links Add/Change/Delete (Tag: X12OBAppLinkAddChangeDelete)	2396
X12 Outbound Trading Partner Trade Link Delete (Tag: X12OBTradeLinkDelete)	2397
EDIFACT partners	2397
EDIFACT Trading Partners (Tag: EDFTradingPartners)	2397
EDIFACT Trading Partners Change (Tag: EDFTradingPartnersChange)	2398
EDIFACT Trading Partners Delete (Tag: EDFTradingPartnersDelete)	2399
EDIFACT Application Partners (Tag: EDFAppPartners)	2399
EDIFACT Application Partners Change (Tag: EDFAppPartnersAddChange)	2400
EDIFACT Application Partners Delete (Tag: EDFAppPartnersDelete)	2400
EDIFACT inbound trade links	2401
Examples	2401
EDIFACT Inbound Trade Links (Tag: EDFIBTradeLink)	2402
EDIFACT Inbound Application Partner Trade Links (Tag: EDFIBAPLink)	2402
EDIFACT Inbound Applications Trade Links (Tag: EDFInboundGroups)	2403
EDIFACT Inbound Trade Link Change (Tag: EDFIBTradeLinkChange)	2403
EDIFACT Inbound Application Partner Trade Links Change (Tag: EDFIBAPLinkChange)	2404
EDIFACT Inbound Applications Trade Links Add, Change, Delete (Tag: EDFInboundGroupsAddChangeDelete)	2404
EDIFACT Inbound Trade Links Delete (Tag: EDFIBTLDelete)	2404
EDIFACT outbound trade links	2405
EDIFACT Outbound Trade Links (Tag: EDOBTPLink)	2405
EDIFACT Outbound Application Partner Trade Links (Tag: EDOBAPLink)	2406
EDIFACT Outbound Applications Trade Links (Tag: EDOBMsgLink)	2406
EDIFACT Outbound Trade Links Change (Tag: EDOBTPLinkChange)	2407
EDIFACT Outbound Application Partner Trade Links Change (Tag: EDOBAPLinkChange)	2407
EDIFACT Outbound Applications Trade Links Add, Change, Delete (Tag: EDOBMsgLinkAddChangeDelete)	2407
EDIFACT Outbound Trade Link Delete (Tag: EDOBTLDelete)	2408
TRADACOMS Trading/Application Partners	2408
TRADACOMS Trading Partners: (Tag TRCTradingPartners)	2409
TRADACOMS Trading Partners Change (Tag: TRCTradingPartnersChange)	2409
TRADACOMS Trading Partners Delete (Tag: TRCTradingPartnersDelete)	2409
TRADACOMS Application Partners (Tag: TRCAppPartners)	2409
TRADACOMS Application Partners Change (Tag: TRCAppPartnersChange)	2410
TRADACOMS Application Partners Delete (Tag: TRCAppPartnersDelete)	2410
TRADACOMS inbound trade links	2410
TRADACOMS Inbound Trade Link (Tag: TRCIBLink)	2411
TRADACOMS Inbound Trade Link Document (Tag: TRCIBTLDocs)	2411
TRADACOMS Inbound Trade Link Change (Tag: TRCIBLinkChange)	2412
TRADACOMS Inbound Trade Link Document Add, Change, Delete (Tag: TRCIBTLDocsAddChangeDelete)	2412
TRADACOMS Inbound Trade Link Delete (Tag: TRCIBLinkDelete)	2412
TRADACOMS Outbound Trade Links	2413
TRADACOMS Outbound Trade Link (Tag: TRCOBLink)	2413
TRADACOMS Outbound Trade Link Document: (Tag: TRCOBTLDocs)	2413

TRADACOMS Outbound Trade Link Change (Tag: TRCOBLinkChange)	2414
TRADACOMS Outbound Trade Link Document Add, Change, Delete (Tag: TRCOBTLDocsAddChangeDelete)	2414
TRADACOMS Outbound Trade Link Delete (Tag: TRCOBLinkDelete)	2415
Setting up the tutorial	2415
Software requirements	2415
Using Partner Manager	2416
Tutorial exercise 1: defining the Partner Manager database	2416
Task: define the ODBC data source	2416
Task: define the database in Partner Manager	2417
Task: define server directories in Partner Manager	2417
Tutorial exercise 2: defining trading partners	2417
Task: analyze the EDI file	2417
Task: define an internal trading partner	2418
Task: define an external trading partner	2418
Task: define an administrative contact for an external trading partner	2419
Tutorial exercise 3: defining post offices	2419
Task: add a file get post office	2419
Task: add a file put post office	2419
Task: add another put post office	2420
Tutorial exercise 4: defining inbound trade links	2420
Task: define the trade link	2420
Tutorial exercise 5: creating trade link type trees	2421
Task: run the EDI wizard	2421
Using message manager	2421
Tutorial exercise 6: deploying message manager	2421
Task: identify the trading manager database	2422
Task: deploy the message manager system	2422
Task: deploy the RunMaps system	2422
Tutorial exercise 7: configuring system resources and the Launcher	2422
Task: create resource configuration files	2423
Task: configure the IBM WebSphere Transformation Extender with Launcher	2423
Task: define the IBM WebSphere Transformation Extender with Launcher in the management Console	2423
Task: run the IBM WebSphere Transformation Extender with Launcher	2424
Task: drop the test file	2424
Task: verify that data was validated and routed in Snapshot Viewer	2424
Task: record any errors and your solutions and check corrections	2424
Integrating external systems	2424
Tutorial exercise 8: deploying an inbound application system	2425
Assumptions	2425
Task: Define a new external trading partner	2425
Task: define a get post office	2426
Task: define a trade link and select acknowledgement level	2426
Task: define an application subsystem	2426
Task: restart the Launcher and run the system	2427
Tutorial exercise 9: deploying an outbound application system	2427
Files for this exercise:	2428
Task: define the application partner	2428
Task: define a post office for outgoing invoice	2428
Task: define a new trade link for the outgoing 810	2428
Task: run EDI type tree wizard	2428
Task: deploy message manager system and RunMaps system	2429
Task: define new application map in the AppMap system	2429
Task: start the Launcher and monitor the system	2429
Partner Manager monitoring	2429
Tutorial exercise 10: running traffic reports	2429
Task: run inbound transmission traffic report	2430
Task: display transmission detail	2430
Task: export transaction set detail report	2430
Tutorial exercise 11: running additional reports	2430
Task: run inbound trade link report	2431
Task: export additional report	2431
Task: Print additional report	2431
Trading Manager maintenance	2431
Tutorial exercise 12: copying the Partner Manager database	2431
Task: create DSN for empty database	2431
Task: copy Partner Manager database	2432
Tutorial exercise 13: security setup	2432
Task: create security group	2432
Task: create user ID and assign to security group	2432
Task: restart Partner Manager and log on	2432

Glossary

Glossary	2433
A	2433
B	2433
C	2433
D	2434
E	2434
F	2435
G	2435
H	2435
I	2435
J	2436
L	2436
M	2436
O	2436
P	2437
R	2437
S	2437
T	2438
U	2439
V	2439
W	2439

Introduction

Congratulations on your selection of IBM® Sterling Transformation Extender as your data transformation engine.

IBM Sterling Transformation Extender is a powerful industry-leading data integration and transformation solution that enables the development of high-volume, complex data integrations without the need for hand-coding.

IBM Sterling Transformation Extender enables your organization to integrate industry-based customer, supplier and business partner transactions across the enterprise and provides industry-leading transformation capabilities supporting many types of data, with native support for JSON, XML or any proprietary data format.

IBM Sterling Transformation Extender provides an easy-to-use browser-based development environment and supports centralized team-based collaboration. The integrated design platform makes it easy for your team to create data flows of any complexity, and to deploy these to a runtime server.

Data integrations can be invoked by use of REST APIs, that run on a scheduled basis, or can be triggered upon a variety of events, such as the arrival of a message on a queue, or the creation of a file. ITX provides a large set of adapters that interface to databases, applications, APIs, messaging middleware and other technologies. A typical integration may read data from one or more sources, validate, transform and merge data, then forward the results to one or more target resources.

IBM Sterling Transformation Extender can engage data residing in the cloud – and in various hybrid cloud contexts – enabling easy access and integration of data and evolving technologies, protecting your development investments.

Additional available advanced transformation support provides metadata for mapping, compliance checking and related processing functions for specific industries – including finance, healthcare and supply chain.

- [The business role of IBM Sterling Transformation Extender](#)
- [Accessibility features for IBM Sterling Transformation Extender](#)

Accessibility features help users who have a physical disability, such as restricted mobility or limited vision, to use information technology products successfully. IBM strives to provide products with usable access for everyone, regardless of age or ability.

- [Globalization](#)
- [Product overview](#)
- [Product architecture](#)
- [Runtime services](#)

With the consistent architecture and integration methodology of IBM Sterling Transformation Extender runtime services, you can create seamless, end-to-end integration solutions that support any combination of applications, interface formats, interprocess communication, and integration flow logic.

- [Design environment](#)
- [Server components](#)
- [Software Development Kit \(SDK\)](#)
- [B2B management \(Trading Manager\)](#)
- [Other IBM Sterling Transformation Extender solutions](#)
- [Integration maps](#)

Maps provide a non-programmatic approach to the development of interfaces, with prebuilt functionality for connectivity, transformation, and routing. Components can be created once and reused many times for different types of integration scenarios.

- [Execution](#)
- [Examples](#)
- [Tivoli License Manager](#)

The business role of IBM Sterling Transformation Extender

IBM® Sterling Transformation Extender provides the following:

- highly automated transformation and routing of complex data across many points of integration in real time to support high-message volumes
- enterprise-wide interoperability supported by a Services-Oriented Architecture (SOA) for seamless connectivity and interoperability across back-office systems
- support for high-performance, event-driven, transactional environments to ensure completion and validation of transactions in real time
- seamless integration across the development, data, and production layers of the enterprise, leveraging existing IT infrastructures
- out-of-the-box solutions for industry standards and regulatory compliance for operational and transactional data integration

Accessibility features for IBM Sterling Transformation Extender

Accessibility features help users who have a physical disability, such as restricted mobility or limited vision, to use information technology products successfully. IBM strives to provide products with usable access for everyone, regardless of age or ability.

Accessibility features

The following list includes the major accessibility features in IBM® Sterling Transformation Extender. These features support:

- Keyboard-only operation.
- Interfaces that are commonly used by screen readers.

Keyboard navigation

The following product components use standard Microsoft Windows navigation keys:

- Design Studio applications

- Command Server
- Launcher Monitor
- Snapshot Viewer

Resource Registry, Management Console, and Launcher Administration use standard Java navigation keys.

Vendor software

IBM Sterling Transformation Extender includes certain vendor software that is not covered under the IBM license agreement. IBM makes no representation about the accessibility features of these products. Contact the vendor for the accessibility information about its products.

Related accessibility information

The following applications have some additional keyboard functionality. The key combinations are documented in the application-specific documentation.

- Database Interface Designer
- Integration Flow Designer
- Launcher Management Tools
- Map Designer
- Type Designer

IBM and accessibility

See the [IBM Accessibility Center](#) for more information about the commitment that IBM has to accessibility.

Related information

- [Unique key combinations for the Database Interface Designer](#)
- [Unique key combinations for the Integration Flow Designer](#)

Globalization

Globalized products can be used without language or culture barriers and can be enabled for a specific locale.

Check the latest information about translated documentation in the [release notes](#).

By default, your computer's locale setting determines the language in which the IBM® Sterling Transformation Extender user interface is displayed. For example, if you purchased your computer in Germany, your computer's locale is most likely set to "German" by default. In this case, the IBM Sterling Transformation Extender interface elements and error messages display in German. Date and time formats, as well as currencies, are presented in accordance with the region that is specified.

- [To change application locale \(Windows\)](#)
IBM Sterling Transformation Extender applications are designed to display in the same language or locale to which your computer is set, provided that the applications support that language.
- [To change the application locale \(UNIX\)](#)
- [To change the Launcher Administration locale](#)
- [To change the Management Console locale](#)
- [To change the Resource Registry locale](#)
- [Locale values](#)
Locale values set the language locale of your computer.

To change application locale (Windows)

IBM® Sterling Transformation Extender applications are designed to display in the same language or locale to which your computer is set, provided that the applications support that language.

You can change the locale of IBM Sterling Transformation Extender to a language or locale other than the one to which the computer is set. For example, the application displays Spanish but you want it to display English instead.

This functionality is not available for MVS, CICS, or Batch.

- [To change the application locale \(Windows\) for the Design Studio application](#)
You can change the locale of the Design Studio application that runs on Windows operating systems.
- [To change the application locale \(Windows\) for tools and runtime applications](#)
You can change the locale of IBM Sterling Transformation Extender tools and runtime applications such as the Command Server and the Launcher that run on Windows operating systems.

To change the application locale (Windows) for the Design Studio application

You can change the locale of the Design Studio application that runs on Windows operating systems.

The Design Studio application must be closed before performing this task.

- From the product installation directory, open the DTXCommon.bat file using a text editor.
In the Locale for Eclipse Applications section, the following verbiage is present:

```
@rem set NLLOCALE=
```

- Remove the remark comment indicator (@rem) in front of `set NLLOCALE=`.
- At the end of the line, enter a valid [locale value](#).
- Save and close the DTXCommon.bat file.

The next time you open the Design Studio application, the interface elements and error messages are displayed in the selected language.

Related concepts

- [Locale values](#)

To revert back to the computer's default locale for the Design Studio application

- Open the DTXCommon.bat file using a text editor.
- Place a remark comment indicator (@rem) in front of `set NLLOCALE=`.
- Save and close the DTXCommon.bat file.

To change the application locale (Windows) for tools and runtime applications

You can change the locale of IBM® Sterling Transformation Extender tools and runtime applications such as the Command Server and the Launcher that run on Windows operating systems.

The IBM Sterling Transformation Extender tools and runtime applications must be closed before performing this task.

- From the product installation directory, open the config.yaml file using a text editor.
In the config.yaml /runtime/locale section, the following verbose is present:

```
# Possible values: de,en,es,fr,it,ja,ko,pt_BR,ru,zh_CN,zh_TW
locale: en
```
- Replace the value `en` with a valid [locale value](#).
- Save and close the config.yaml file.

The next time you open the IBM Sterling Transformation Extender tool or runtime application, the interface elements and error messages are displayed in the selected language.

To revert back to the computer's default locale for tools and runtime applications

- Open the config.yaml file using a text editor.
- Set the value of /runtime/locale: to an empty string in the config.yaml file. For example:

```
/runtime
  /locale:
```

- Save and close the config.yaml file.

To change the application locale (UNIX)

IBM® Sterling Transformation Extender applications must be closed before performing this task.

- Open common.sh in a text editor.
- Find `export USRLOCALE=` and enter a [locale value](#).
Example for Korean: `export USRLOCALE=ko`
- Save and close the file.

To revert back to the computer's default locale, remove the locale value and save changes.

Related concepts

- [Locale values](#)

To change the Launcher Administration locale

The Launcher Administration application must be closed before performing this task.

- Open `install_dir/LauncherAdmin.bat` (launcheradmin.sh) in a text editor.
- At the `REM set USRLANGUAGE=` line, remove the `REM` comment indicator. At the end of the line, enter a [locale value](#).
Example for Korean: `set
USRLANGUAGE=ko`
- At the next line, `REM set USRCOUNTRY=`, remove the `REM` comment indicator. At the end of the line, enter the corresponding country indicator if needed.
Example for Brazilian Portuguese:

```
set USRLANGUAGE=pt
set USRCOUNTRY=BR

4. At the line that begins with REM call %JRE% -DINSTALLDIR="%DTXHOME%" -Duser.language=%USRLANGUAGE% ..., remove the REM comment indicator from the beginning of the line.
5. At the next line (call %JRE% -DINSTALLDIR="%DTXHOME%" -Dsun.java2d.nodraw ...), add an REM comment indicator to the beginning of the line.
For Brazilian Portuguese, the modified text should look similar to the following:
```

```
set USRLANGUAGE=pt
set USRCOUNTRY=BR

call %JRE% -DINSTALLDIR="%DTXHOME%" -Duser.language=%USRLANGUAGE% ...
REM call %JRE% -DINSTALLDIR="%DTXHOME%" -Dsun.java2d.nodraw ...
```

6. Save and close the file.

Related concepts

- [Locale values](#)

To change the Management Console locale

The Management Console application must be closed before performing this task.

1. Open *install_dir/MgmtConsole.bat* (*mgmtconsole.sh*) in a text editor.
2. At the REM set USRLANGUAGE= line, remove the REM comment indicator. At the end of the line, enter a [locale value](#).
Example for Spanish: set
USRLANGUAGE=es
3. At the next line, REM set USRCOUNTRY=, remove the REM comment indicator. At the end of the line, enter the corresponding country indicator if needed.
Example for Simple Chinese:

```
set USRLANGUAGE=zh
set USRCOUNTRY=CN
```

4. At the line that begins with REM call %JRE% -DINSTALLDIR="%DTXHOME%" -Duser.language=%USRLANGUAGE% ..., remove the REM comment indicator from the beginning of the line.
5. At the next line (call %JRE% -DINSTALLDIR="%DTXHOME%" -Dsun.java2d.nodraw ...), add an REM comment indicator to the beginning of the line.

For Simple Chinese, the modified text should look similar to the following:

```
set USRLANGUAGE=zh
set USRCOUNTRY=CN

call %JRE% -DINSTALLDIR="%DTXHOME%" -Duser.language=%USRLANGUAGE% ...
REM call %JRE% -DINSTALLDIR="%DTXHOME%" -Dsun.java2d.nodraw ...
```

6. Save and close the file.

To change the Resource Registry locale

The Resource Registry application must be closed before performing this task.

1. Open *install_dir/ResourceRegistry.bat* (*resourceregistry.sh*) in a text editor.
2. At the REM set USRLANGUAGE= line, remove the REM comment indicator. At the end of the line, enter a [locale value](#).
Example for Japanese: set
USRLANGUAGE=ja
3. At the next line, REM set USRCOUNTRY=, remove the REM comment indicator. At the end of the line, enter the corresponding country indicator if needed.
Example for Traditional Chinese:

```
set USRLANGUAGE=zh
set USRCOUNTRY=TW
```

4. At the line that begins with REM call %JRE% -DINSTALLDIR="%DTXHOME%" -Duser.language=%USRLANGUAGE% ..., remove the REM comment indicator from the beginning of the line.
5. At the next line (call %JRE% -DINSTALLDIR="%DTXHOME%" -Dsun.java2d.nodraw ...), add an REM comment indicator to the beginning of the line.

For Traditional Chinese, the modified text should look similar to the following:

```
set USRLANGUAGE=zh
set USRCOUNTRY=TW

call %JRE% -DINSTALLDIR="%DTXHOME%" -Duser.language=%USRLANGUAGE% ...
REM call %JRE% -DINSTALLDIR="%DTXHOME%" -Dsun.java2d.nodraw ...
```

6. Save and close the file.

Related concepts

- [Locale values](#)

Locale values

Locale values set the language locale of your computer.

Locale value

	Language
de	German
en	English
es	Spanish
fr	French
it	Italian
ja	Japanese
ko	Korean
pt_BR	Brazilian Portuguese
ru	Russian
zh_CN	Simplified Chinese
zh_TW	Traditional Chinese

Related tasks

- [To change the application locale \(Windows\) for the Design Studio application](#)
- [To change the application locale \(UNIX\)](#)
- [To change the Resource Registry locale](#)
- [To change the Launcher Administration locale](#)
- [To change the help locale](#)

Product overview

IBM® Sterling Transformation Extender performs transformation and routing of data from source systems to target systems in batch and real-time environments. The sources may include files, relational databases, MOMs (message-oriented middleware), packaged applications, or other external sources. After retrieving the data from its sources, the IBM Sterling Transformation Extender product transforms it and routes it to any number of targets, providing the appropriate content and format for each target system.

The IBM Sterling Transformation Extender product delivers the following:

- connectivity to a wide range of mainframe, legacy, and enterprise applications, databases, messaging systems, and external information sources
- a comprehensive library of pre-built functions to reduce development time and simplify specification of rules for validation, transformation, and routing
- multiple execution options to support right-time, right-style transformation—whether it is batch, real-time, or embedded
- enterprise-class capabilities for development, deployment, and maintenance plus high-availability platform support.

This reduces on-going administration and implementation risks and delivers results sooner than hand-coding.

IBM Sterling Transformation Extender is a runtime environment that implements both event-driven and command-driven integration for:

- Enterprise Resource Planning (ERP) applications
- Customer Relationship Management (CRM) applications
- Supply Chain Management (SCM) applications
- other off-the-shelf applications
- customer-developed, in-house applications
- direct, business-to-business integration
- business community integration
- integration with trade networks, net markets, and business service providers

In IBM Sterling Transformation Extender terms, an integration solution consists of a system of related interfaces, implemented as rule-based maps, that consume and produce data in support of a business process. Maps define interfaces between applications, data stores, middleware, and other maps.

Product architecture

The IBM Sterling Transformation Extender product is client/server software composed of a client-based design environment, several server-based execution options, and administration and operation capabilities that can be run on the client or server.

Runtime services

With the consistent architecture and integration methodology of IBM® Sterling Transformation Extender runtime services, you can create seamless, end-to-end integration solutions that support any combination of applications, interface formats, interprocess communication, and integration flow logic.

- Launcher
The Launcher monitors and evaluates runtime events and triggers map execution according to associated rules.
- Command Server
The IBM Sterling Transformation Extender Command Server supports command-driven and scripted invocation of integration solutions.
- Engine
The IBM Sterling Transformation Extender engine executes maps in the runtime environments.
- IBM Sterling Transformation Extender resource adapters
The IBM Sterling Transformation Extender resource adapters bind logical map interfaces with applications, databases, middleware services, and other resources, hiding the details of APIs, marshalling, and low-level interaction protocols.

Related concepts

- [Launcher](#)
- [Launcher Agent](#)
- [Resource adapters](#)

Related reference

- [Command-driven execution model](#)

Design environment

With the TX design environment, you can develop event-driven application-to-application (A2A) integration, business-to-business (B2B) integration, and consumer-to-business (C2B) application integration. The design environment models and tests integration solutions.

You use the design environment to create IBM Sterling Transformation Extender data integration solutions. The design environment provides a graphical interface for managing the integration of applications and business processes.

The Design Server is the client interface into an extensive array of functionality that is available to create application integration solutions. Application integration solutions created with the Design Server run on transformation servers specific to the IBM Sterling Transformation Extender product being used. The separation of design and execution provides exceptional flexibility for implementing enterprise-wide integration solutions. A single Design Server can deploy integration solutions to multiple servers. Multiple instances of Design Server can create integration solutions for a single production environment.

With the Design Server, you can:

- Define the structure of source data and target data with the schema designer
- Import database object metadata and identify characteristics of those objects to meet mapping and execution requirements with the Database Interface Designer
- Define transformation rules with the map designer
- Design systems of maps to model data integration processes with the Integration Flow Designer
- Develop and test maps and systems with the map designer and the Integration Flow Designer
- Deploy maps and systems to the execution environment with the Integration Flow Designer
- [**Projects**](#)
A project organizes maps and artifacts in Design Server. You can reuse project-level artifacts in multiple maps.
- [**Schema design**](#)
- [**Map design**](#)
- [**Resource adapters**](#)
- [**Connection and action design**](#)
Connections define the adapter properties that are required to connect to an external resource. An action uses a connection to provide source data to a map or write target data from a map. You can define both connections and actions at the project level or at the map level.
- [**Configuration variables**](#)
Configuration variables are aliases that resolve to actual resources at run time.
- [**Database Interface Designer**](#)
- [**Integration Flow Designer**](#)
- [**Type Tree Maker**](#)

Projects

A project organizes maps and artifacts in Design Server. You can reuse project-level artifacts in multiple maps.

Schema design

Use the schema designer to:

- Create and manage schema that define properties for data structures.
- Define containment of data.
- Create data validation rules.

Using schemas

A **schema** is a graphical data dictionary that contains metadata definitions of the structure of the inputs and outputs that are the source and target data for integration solutions. It provides visual cues for understanding the structure of an object and supports point-and-click techniques enabling you to "drill down" to uncover an object's complete definition.

Each data object in the input or output data is defined as a reusable object in a schema that uses a sophisticated set of properties to describe attributes such as length, justification, possible values, delimiters, and the order of the data objects that comprise a complex data object. You can use the schema designer to define properties for text or binary data, different character sets, data structures, and semantic validation rules. The sophisticated type properties and type model of IBM Sterling Transformation Extender enable the precise definition of highly complex data, such as EDI, SWIFT, and legacy data structures.

- Schemas can be manually created using the schema designer.
- Schemas can be automatically created by a schema importer or the Database Interface Designer. These components generate schemas based on known metadata, such as database catalogs, SAP R/3 IDocs, or COBOL Copybooks.
- Schemas can be predefined and included as part of IBM Sterling Transformation Extender Packs that support particular standards, such as ASC/X12, UN/EDIFACT, SWIFT, or HIPAA.
- Schemas can be created by the Type Tree Maker: a tool that enables creation of custom importers to automate the capture of metadata definitions from other machine-readable sources.
- The IBM Sterling Transformation Extender includes native support for XSD and JSON schemas.

Importing schemas

The schema designer includes importers to automatically generate schemas (data object definitions) from known metadata, such as COBOL Copybooks.

Some importers are available in the Design Server. Additional importers for data formats are provided as part of the IBM Sterling Transformation Extender Packs.

Comparing schemas for differences

You can use the schema differences feature in the schema designer to compare two schemas. When the analysis is complete, both schemas appear. Source compare differences are distinguished by either blue or red text. When a type is present in one schema but not in the other, the text is blue. When a type is present in both schemas, but they each have different properties, components, or restrictions, the text is red. There is a "step-through" facility that cycles through each of the differences in turn. In extremely large, complex schemas, this can save a great deal of time.

Related concepts

- [Other IBM Sterling Transformation Extender solutions](#)

Map design

The IBM® Sterling Transformation Extender map designer is the modeling component used to formulate transformation and business rules. The map designer uses definitions of data objects created in the schema designer as inputs and outputs. The map designer provides functionality for specifying rules for transforming and routing data, as well as the environment for analyzing, compiling, and testing the maps that are developed.

This process is facilitated by convenient "from" and "to" windows, drag-and-drop techniques, and spreadsheet-like rules. Mapping rules are added using logical statements. The map designer provides a rich set of predefined functions for operations such as conditional testing, table lookups, mathematical functions, character string parsing, and data extraction.

Using maps

A **map** is the embodiment of complete definitions of data objects and the rules for their transformation. A map can implement a wide range of integration functions—from simple transformation to sophisticated integration solutions involving multiple, heterogeneous inputs and outputs, rule-based routing, and complex interface structures.

Maps are created using the map designer, a graphical tool that makes complex transformations easier to define and maintain. Maps are analyzed, compiled, and tested directly in the Design Server environment.

Any number of inputs or outputs can be used within a single map in an any-to-any fashion. A map contains a *card* for each input and output of the transformation.

The capability for mapping from multiple inputs to multiple outputs gives IBM Sterling Transformation Extender products unique transformation power. Each input to and output from a map is associated with a specific data object definition and a resource adapter. Data object definitions are reusable—the same definition can apply to one or more inputs; to both input and output for the same map; or to different maps. When a data object is used as input, data is validated to ensure the data conforms to the data object definition. When used as output, the data object is constructed automatically, in its entirety.

Input objects are not consumed when used in a transformation rule. This allows for a one-to-many relationship between an input and its outputs. An output can be constructed from any number of zero or more inputs. This creates a one-to-many relationship between an output and its inputs. Taken together, IBM Sterling Transformation Extender products provide many-to-many transformation support between inputs and outputs.

Data object definitions, data maps, and resource adapters are all separately managed objects. Each object is reusable in any number of integration scenarios. The result is flexibility, ease of use, and reduced maintenance as integration requirements change.

Defining server platforms for maps

The map definition is completely portable between all supported server platforms. Simply select the target platform from a list when building the map to generate a compiled map file that is optimized for the specified platform.

Exporting maps

The Design Server can export the rules and execution settings of both maps and schemas to JSON documents.

Comparing maps for differences

The map designer allows you to compare two map source files.

- [Input and output cards](#)
- [Transformation, routing, and business logic rules](#)
- [Adapter configuration for input and output](#)

Related concepts

- [Integration maps](#)

Input and output cards

Each input and output is represented in a separate map card. The data structure for each of the inputs and outputs is represented by a schema created using the schema designer.

Transformation, routing, and business logic rules

Schemas are completely interchangeable between inputs and outputs. The same schema can be reused multiple times by multiple maps. After an output card for a map is associated with a type in a schema in the map designer, a rule in the output card defines how to build the output data.

Adapter configuration for input and output

IBM® Sterling Transformation Extender connections, actions, and resource adapters retrieve data from sources and send data to targets. They do not perform transformation. (Map rules describe the nature of the transformation, and the transformation server transforms the data.)

IBM Sterling Transformation Extender includes adapters for common, enterprise-level resources. In its simplest form, specifying an adapter for an input or output card means selecting the required adapter from a list.

Adapter settings defined in a connection can be overridden with different adapter settings at map run time.

The resource adapters are loosely coupled with the transformation process. An adapter retrieves data to be processed by the map as a byte stream. This means that, to the extent that the type definitions and byte streams are compatible, schemas and adapters can be used interchangeably.

For example, you might use a file as a data source when testing, while in the production environment, the data source is a database. As long as the format of the data retrieved from the database is consistent with the format of the data in the file, you only need to change the input from using the **File** adapter to using the **Database** adapter to run the map in production.

In fact, the map source file does not need to be changed at all because the map execution settings, including the adapter configuration settings, can be overridden at run time.

Where possible, resource adapters use the native client libraries of the concerned resources to connect to those resources. It is through this mechanism that IBM Sterling Transformation Extender products can achieve "direct" connectivity to those resources. For example, the IBM Sterling Transformation Extender Oracle adapter uses the Oracle Net8 client to achieve connectivity to Oracle databases.

IBM Sterling Transformation Extender provides an open API to allow for the development of a user-defined adapter, should the need arise. This allows connectivity to unforeseen, possibly proprietary systems (such as a "home-grown" HR system), in addition to the standards-based adapters that IBM Sterling Transformation Extender provides.

Related concepts

- [Map settings](#)

Resource adapters

IBM Sterling Transformation Extender includes an extensive collection of resource adapters to connect maps to middleware, DBMSs, and utility services.

Resource adapters can indirectly integrate with enterprise applications that consume inputs or produce outputs using message queues, standard transports, file systems, and databases. In addition, resource adapters enable message-based integration with external and internal applications, including applications and services from SAP, Oracle, and others.

- Database adapters support databases such as DB2, Informix, ODBC, OLE DB, Oracle, and Sybase.
- Internet adapters support email, FTP and HTTP.
- Message-Oriented Middleware (MOMs) adapters support Microsoft MSMQ, Oracle AQ, and TIBCO Rendezvous.
- Synchronous adapters support COM+ and Sockets.
- Utility adapters support archive, batch, and directory (LDAP) services.
- File adapters support platform-specific file systems.

In addition to pre-built resource adapters and packs, the Adapter Toolkit provides a standard interface and set of services for implementing your own adapters to connect directly with other applications and services.

Related concepts

- [Runtime services](#)
-

Connection and action design

Connections define the adapter properties that are required to connect to an external resource. An action uses a connection to provide source data to a map or write target data from a map. You can define both connections and actions at the project level or at the map level.

Configuration variables

Configuration variables are aliases that resolve to actual resources at run time.

You can use variables for connection and action properties, paths in map and card settings, and map rules. You can define a variable to resolve to a different resource based on the server it's deployed to. Configuration variables make it possible to use the same map in different deployment environments.

Database Interface Designer

The Database Interface Designer is the modeling component used to import metadata about queries, tables, and stored procedures for data stored in relational databases. The Database Interface Designer identifies characteristics such as update keys and database triggers, of those objects to meet mapping and execution requirements.

Use the Database Interface Designer to:

- Specify the databases to use for a source or target, along with the user IDs and passwords required for the connection.
- Define query statements.
- Automatically generate schemas for queries, stored procedures, input and output parameters, or tables.
- Define tables for which any inserts, updates, or deletions would trigger maps using the Launcher.

If connectivity to the database server is not available from the development platform, the mtmaker utility is provided to execute on the server platform. This utility creates a script file that can be transferred to the development environment to generate the schemas.

Integration Flow Designer

The IBM® Sterling Transformation Extender Integration Flow Designer is the modeling component used to define and manage data integration processes. The Integration Flow Designer is used to define interactions among maps and systems of maps, to validate the logical consistency of workflows, and to prepare systems of maps to run.

Using systems

A system is a set of maps, or subsystems, that are combined to form a data integration process flow. Systems are defined in the Integration Flow Designer.

Systems are defined as having one of two execution modes-Launcher or Command Server, depending upon how they will be deployed. If a system is developed for the Launcher, the system will include definitions of events based upon the map execution to be triggered. As mentioned previously, these events may be time events (for example, execute every 10 minutes or every Monday at 2:43 AM), source events (for example, a message appearing on a queue, a row being updated in a database), or compound events comprised of some combination of time and/or source events.

The set of events that initiates a map within a system is called a *watch*. Each event that contributes to the initiation of a map is called a *trigger*. When the resource (file, database, or message queue) that is an input to a map changes state (that is, it is created or modified), this input event can cause the Launcher to initiate a map. Adapters that can be used to trigger maps have a listener function to detect the appropriate external resource change.

The Launcher starts maps based upon these triggers. There may be multiple triggers that must collectively exist before a map is initiated. The Launcher manages the coordination of these events to ensure that the correct set of circumstances has occurred before a map is initiated.

For more information about the Integration Flow Designer, see the *Integration Flow Designer* documentation. For more information about the Launcher, see the *Launcher* documentation.

Using links

A link is a connection that the Integration Flow Designer creates between two components in a system. Links represent the direction of data flow between the system components at execution time, thus showing the source and target dependencies between system components.

Managing systems

The Integration Flow Designer provides clarity and control to manage the more complex systems built using IBM Sterling Transformation Extender products. It also reduces the administrative burden and simplifies the non-development activities associated with interface development. Some of the key features of the Integration Flow Designer are:

- Graphical data flow design
Visual data flows are created to aid in the design of integration solutions. You can use a graphical palette to produce system definition diagrams and to easily navigate among the IBM Sterling Transformation Extender Design Studio tools. Point-and-click techniques can be used to select the map and subsystem components to place in a specific integration flow diagram.

Using the Integration Flow Designer, you can also define specific map settings that take effect at run time. Select the maps and the data flow between them is automatically determined from the inputs and outputs. The **Override** tool in the Integration Flow Designer toolbox can be used to modify the inputs and outputs to achieve the desired data flows. These modifications do not affect the source map. This allows the same map to be reused in multiple data flows.

- Analysis of integration provided for logical consistency
The Integration Flow Designer includes a System Analyzer that checks your system definitions for logical consistency to ensure they can be executed. The System Analyzer detects any inconsistencies you may have introduced in the definition while experimenting with system models.
- Capability to build and port entire systems with simple mouse-clicks
After a system has been designed, the Integration Flow Designer offers another benefit that saves time when preparing a system for operation. One mouse-click builds and deploys all of the integration objects and supporting data (such as lookup files and scripts).

Parameters for different servers representing different operating environments can be defined within the Integration Flow Designer.

After the appropriate server definitions have been created, it is then possible to produce, using the wizard, automated deployment definitions that are specific to each server.

It is possible to specify, within a script, that either all maps are built and transferred or that only a specific subset of the maps in a system are build and transferred. You can also include additional files required to support the system's operation in the transfer.

For more information about the Integration Flow Designer, see the *Integration Flow Designer* documentation.

- [Developing application interfaces](#)

Developing application interfaces

The IBM Sterling Transformation Extender development environment includes a robust set of integrated tools designed to solve varied application integration problems. Except for scenarios in which the IBM Sterling Transformation Extender Software Development Kit (SDK) is used, you can use these tools to solve integration problems without modifying your applications.

IBM Sterling Transformation Extender products are designed to fit your interface architecture, whether it is hub-based or distributed. Some companies prefer hub-models with a centrally managed integration server. Other organizations choose to distribute integration servers to reside on the same platform as the applications they service. IBM Sterling Transformation Extender products can be used in either configuration.

To design an application interface, you need to understand the content of each data source and target, as well as the middleware, transports, and data services needed to communicate with corresponding sources and targets. You also need to define rules for transforming data content from inputs to outputs.

As you develop more and more interfaces, you will want to reuse metadata and integration rules. Additionally, you will want to formulate business process models: interface systems consisting of integration components that are configured to work together. These systems also need to be reusable.

Use the IBM Sterling Transformation Extender Design Server to develop new integration components and to reuse existing components so that you can leverage your development efforts across integration projects.

- [Modeling the data of business objects](#)
- [Defining transformation rules](#)
- [Defining data sources and targets](#)
- [Defining process models](#)

Modeling the data of business objects

Although you can use the IBM® Sterling Transformation Extender Design Studio to represent the properties of individual fields, elements, and records, its strength lies in defining complete *business objects*. In the IBM Sterling Transformation Extender Design Studio, business objects are the substance of business transactions such as the following: health claims, invoices, bank transfers, airline reservations, ship notices, or telephone bills.

Business objects that include interface syntax, semantics, and structure are implemented in the IBM Sterling Transformation Extender products as *schemas*, which may be pre-defined (in the case of standards supported out-of-the-box by IBM Sterling Transformation Extender), imported from metadata, or defined using the IBM Sterling Transformation Extender schema designer. Once defined, they can be re-used to describe interface content for multiple sources and targets.

- Use the Type Designer to define properties for text or binary data, different character sets, data structures, and semantic validation rules. The resulting type definition is enforced automatically and transparently, when the IBM Sterling Transformation Extender system executes.

- When external metadata exists, you can reduce the effort of defining business objects by using application importers. For example, you can use the COBOL Copybook Importer to generate schemas from COBOL record structures, or the Database Interface Designer to generate type trees from database catalog entries. You can use the DTD Importer to generate type trees from XML Document Type Definitions (DTDs). SAP R/3 users can generate type trees from application-maintained metadata, using IBM Sterling Transformation Extender-supplied importers.
- IBM Sterling Transformation Extender products provide a repository of predefined type trees for popular e-commerce, healthcare, insurance, and financial standards, including ASC X12, EDIFACT, HIPAA, HL7, SWIFT, ISO 15022, and others.
- Using the Type Tree Maker, which is also part of the IBM Sterling Transformation Extender Design Studio, you can build your own importer to automate the capture of metadata definitions from machine-readable sources.

Regardless of method used, the result is a set of reusable business object definitions encapsulated in a schema.

Defining transformation rules

Complete definitions of data objects and the rules for their transformation are embodied in an IBM® Sterling Transformation Extender *map*. Map objects are created using the Map Designer, a graphical tool that makes complex transformations easier to define and maintain. Maps are analyzed, compiled, and tested directly in the IBM Sterling Transformation Extender Design Studio environment.

A map object can implement a wide range of interface functions, from simple transformation to sophisticated integration solutions involving multiple, heterogeneous inputs and outputs, rule-based routing, and complex interface structures.

The IBM Sterling Transformation Extender Design Studio transformation server processes interface data without pre-formatting or preprocessing. You can compare, move, and transform data using any combination of sources and targets, without regard for underlying interface formats. The graphical environment of the Design Studio also delivers complete integration solutions that require no programming to complete the job. Complex interface problems that involve multiple input sources and multiple output targets can be handled in a single map. There is no need to break up a unit of work into multiple steps unless the business problem requires it.

The Design Studio's comprehensive set of integration rules enables you to go beyond transformation to construct and route entirely new business objects. You can apply rules to route business objects to appropriate targets based on data content, runtime conditions, or any other criteria you choose. You can apply rules to filter, scrub, calculate, validate, parse, substitute, expand, change character set or otherwise convert data from multiple inputs to multiple outputs. All of these capabilities supplement simple, drag-and-drop mapping where needed.

With the Design Studio, data sent to an output does not have to come from an input. You can compute data, look up something in a database, sort, extract, merge, or use any of dozens of IBM Sterling Transformation Extender functions to create output, where appropriate.

You can also map outputs to each other. For example, you can transform data that was generated at the top of an output to the bottom of the same output so that you can summarize, tally, or validate what has been created. You can also map an output that was generated in one output to another output. For example, you can generate a message as one output and archive all or part of it as another output.

Multiple input and output types can be used within a single map for any number of sources (adapters) in an any-to-any fashion. IBM Sterling Transformation Extender is also designed for one-to-many, many-to-one and many-to-many processing. This flexibility provides unconstrained support for common integration needs, such as creating a sales distribution document and an invoice from the same purchase order and updating the inventory database at the same time.

Defining data sources and targets

A data source is defined in an IBM® Sterling Transformation Extender system by specifying a resource adapter for an input. A data target is defined by specifying a resource adapter for an output. IBM Sterling Transformation Extender includes resource adapters for files, memory, databases, messaging middleware products, HTTPS, FTP, e-mail, and more. Application adapters are also available for interfacing with popular commercial applications. Furthermore, you can develop your own adapters to integrate directly to new types of sources and targets by using the IBM Sterling Transformation Extender open adapter interface.

The way in which business objects are defined and the way in which map rules are specified is generally independent of the actual data source or data target due to the abstraction afforded by the schema implementation. Business objects deal with the *format* and *content* of the data, serving to answer questions such as "What should the data look like?" and "What should the value of the data be?" Maps deal with the transformation logic required to create new business objects from existing ones. By contrast, resource adapters answer such questions as "Where should the data come from or go to?" and "How does the application or resource expect the map to send or receive the data?"

During interface design, you specify a resource adapter for each input and for each output. This determines the default adapters to be used during execution. You can always change the source or target by using the process modeling component, the Integration Flow Designer, or at execution time. IBM Sterling Transformation Extender transformation solutions are resource-independent.

You can also use resource adapters in map rules. For example, you can use database adapters in a map rule to look up cross-reference data stored in a database. When the resource adapter you want to use varies, depending upon the data content, you can dynamically allocate the resource from within a map rule. The transformation solutions from IBM Sterling Transformation Extender software can be reconfigured.

Defining process models

The Integration Flow Designer is the graphical process modeling component of the IBM® Sterling Transformation Extender Design Studio. Use the Integration Flow Designer during development to model a set of logically related maps called a *system*. Systems can be used as a focal point for interfacing to other development components and for preparing maps for execution.

- [Formulating business process models](#)
- [Defining resource connectivity](#)
- [Defining event rules](#)
- [Configuring map settings](#)
- [Managing systems](#)

Formulating business process models

Using the Integration Flow Designer, you formulate a business process model as a system. System components can be maps or other systems. A transformation component and its inputs and outputs are represented in graphical form. Data flow links are automatically diagrammed from information about the resource adapters assigned to inputs and outputs. During development, you can navigate to an underlying map where integration rules are defined and to the underlying definitions of input and output business objects.

Defining resource connectivity

When you use the Integration Flow Designer to specify resource adapters for map components, links are automatically established when the target of one component is used as the source of another. You can see the flow of information among system components as you design.

Defining event rules

The Integration Flow Designer can be used to configure the event rules that determine the conditions under which a map will run.

A map component can be initiated based on one or more time, file, database application, or message events. For example, you can specify the insertion of data into a database table to be a map trigger. Alternatively, you can also specify the synchronous coordination of multiple events, such as "Trigger this map every day at 4:00 P.M. if there is a message on my queue." Event management can also include wildcard matching for events across inputs and outputs. For transaction-based transformations, there are also rollback, delete, and retry options.

Configuring map settings

The Integration Flow Designer includes many options for configuring each map such as: priority, event expiration settings, audit logging, paging, multi-threading options, and more. This allows you to configure your production systems for optimal performance.

Managing systems

To prepare a system for deployment, you will want to build, analyze, and possibly port, all the map components in that system. Use the Integration Flow Designer to do this.

Consistent control information can be generated from the graphical diagram of system components. For example, you can generate map command files for a Command Server or control files for an Launcher. This information can be used to distribute transformation components and subsystems across different servers to execute systems as an integrated whole.

Type Tree Maker

The Type Tree Maker is a scripting tool that automates the capture of metadata from machine-processable sources to create graphical metadata in the form of type trees.

For more information about the Type Tree Maker, see the *Type Tree Maker* documentation.

Server components

Several mechanisms, including a Java API that enables Java-based interfaces to maps, are available for integration components. All of the methods share a common core engine and adapters. The main methods of execution are:

- Command-driven
The Command Server provides command-driven execution and is used primarily for batch processing.
- Event-driven
The Launcher provides event-driven execution, in which systems of maps can be coordinated to provide a scalable, distributed, multithreaded, execution environment.
- Application-embedded model
The Software Development Kit maps to be embedded in third-party applications.
- [Management and monitoring tools](#)

Related concepts

- [Execution](#)

Management and monitoring tools

The management and monitoring tools are a set of graphical applications that provide a way to manage and view processes running in the Launcher. These graphical tools are available on Windows- and UNIX-based platforms and can be used with IBM Sterling Transformation Extender Launchers that are either local or remote.

- [Launcher Administration](#)
- [Simple Network Management Protocol \(SNMP\) support](#)
- [Management Console](#)
- [Launcher Monitor](#)
- [Snapshot Viewer](#)

Launcher Administration

The Launcher Administration is the administrative interface to the Launcher, from which the user specifies deployment directories, configures users and user access rights, specifies listening ports, and defines properties for Java Remote Method Invocation (RMI).

Configurations can be defined for:

- automatic system startup when the Launcher service is started
- connection ports for the Management Console
- resource resolution directives as defined in the Resource Registry

Simple Network Management Protocol (SNMP) support

Using the SNMP Agent, you can monitor and manage the status of the Launcher and associated map events. The SNMP Agent can be used to configure thresholds, monitor Launcher performance, and assist in identifying failure situations.

SNMP support is provided with the Launcher on all Windows and UNIX platforms. It supports the following traps:

Trap

Warnings and Errors Generated

Map failure

It signals when a map fails.

State and listener change

It signals a change in the state of the Launcher or listeners.

Launcher/resource connection failure

It notifies the SNMP management software that the connection to the Launcher and the resource could not be established.

Thresholds

It signals when a map or connection pending state threshold has been met.

Additionally, an SNMP adapter allows for custom-defined traps to be issued from within the business logic of a map.

For more information about the SNMP Agent, see the *SNMP Agent* documentation.

Management Console

The IBM Sterling Transformation Extender Management Console is the management and monitoring interface for the Launcher, from which the user can start, stop, pause, and resume the system, as well as view information about the status of the Launcher and maps that are currently running. The Management Console also permits centralized, remote management of all Launcher installations.

Separate connections are defined for each Launcher installation. Each Launcher then appears in the main window of the Management Console, and each can be managed from the single, remote interface.

From the Management Console, it is possible to remotely stop, start, and pause systems. Launcher debugging can also be enabled to assist in operational troubleshooting of Launcher operations.

The Management Console provides a variety of statistics about the Launcher, including the length of time it has been running and the number of maps that have been processed.

- Summary: provides a general view of performance and status information.
- Status: contains active and pending system status information.
- History: contains a rollup of the total number of maps that have been run, the number of successes and failures, and so on.
- Configuration: contains the configuration parameters for the Launcher.

For more information about the Management Console, see the *Launcher* documentation.

Launcher Monitor

The IBM Sterling Transformation Extender Launcher Monitor is a GUI that provides a dynamic, detailed view of watches (single map instances) as they run. It also provides the capability to create snapshots of detailed watch activity.

For more information about the Launcher Monitor, see the *Launcher* documentation.

Snapshot Viewer

The Snapshot Viewer is a GUI that displays snapshots taken in the Launcher Monitor. These snapshots show details about the activity of the Launcher at a specific moment in time.

For more information about the Snapshot Viewer, see the *Launcher* documentation.

Software Development Kit (SDK)

The IBM Sterling Transformation Extender Software Development Kit (SDK), which is included in the Design Studio product installation, and related options, provide the means to include transformational capabilities within a custom application or environment and to extend IBM Sterling Transformation Extender product capabilities using an adapter toolkit. When used with an application server or web server, the components in the SDK can be used to facilitate web-based integration scenarios.

The following components comprise the Software Development Kit:

- Software Development Kit Application Programming Interfaces (APIs)
- Software Development Kit Options
- Adapter Toolkit
- [Software Development Kit Application Programming Interfaces \(APIs\)](#)
- [Software Development Kit options](#)
- [Adapter toolkit](#)

Software Development Kit Application Programming Interfaces (APIs)

At the center of the IBM® Sterling Transformation Extender Software Development Kit (SDK) are object-oriented application programming interfaces (APIs) that are available for environments such as C, Java, RMI, and COM. These APIs provide common functionality that is represented in language-specific methods and objects. All APIs within the SDK contain the same underlying objects and methods. The APIs support multithreading to ensure high performance by allowing multiple instances of maps that can be run in parallel.

Map and card settings and parameters that can be changed in the Design Server or Integration Flow Designer can also be passed to the map at runtime through these APIs, thus allowing runtime control of specific instances of the map. For example, an application variable can be used to control audit settings for a particular map or to pass in a filename to be accessed.

Related concepts

- [Application-embedded execution model](#)

Software Development Kit options

Software Development Kit options integrate IBM® Sterling Transformation Extender maps with other application integration and business process management products.

Adapter toolkit

The Software Development Kit includes a toolkit for developing robust custom adapters that can:

- tightly integrate with IBM® Sterling Transformation Extender products using the XML registration file
- participate in connection pooling
- provide an event listener

B2B management (Trading Manager)

The IBM Sterling Transformation Extender Trading Manager is an optional B2B management component. It is a client/server product for managing and processing electronic commerce data that provides the capabilities for managing and controlling the Business-to-Business (B2B) integration of partner relationships and message flow. Trading Manager users can audit, control, monitor, and view the entire B2B integration environment across the extended enterprise with secure data exchange fully integrated with back-end systems. The Trading Manager consists of the following components:

- Partner Manager
Partner Manager is a graphical administration environment used to trade partner profiles and relationships. Partner Manager also provides runtime monitoring, auditing, and reporting functions.
- Message Manager

Message Manager is a runtime subsystem that uses trading profiles maintained by the Partner Manager to automate document exchange among e-commerce partners.

For more information about Trading Manager and its components, see the Trading Manager documentation.

- [Partner Manager](#)
 - [Message Manager](#)
-

Partner Manager

The IBM Sterling Transformation Extender Partner Manager is a GUI application used to create a comprehensive database of an organization's electronic commerce information and to manage its e-commerce activity. Partner Manager is used to:

- define the trading relationships with Electronic Data Integration (EDI) partners, including details such as the version of each document, control numbering schemes, tracking, error handling, status, and post office used.
- maintain the information for each trading partner, including details regarding addresses, contacts, and the individual departments or other groups within trading partner organizations.
- provide audit and control information collected in the Commerce Manager database that is used to monitor ongoing inbound and outbound e-commerce transaction activity.
- provide alerts when error conditions occur in e-commerce inbound or outbound processing.

For more information about the Partner Manager, see the Partner Manager documentation.

Message Manager

The IBM Sterling Transformation Extender Message Manager is a predefined system of maps specifically designed to process electronic commerce data. The maps are used to validate and route data. The Message Manager is the runtime component of the Trading Manager.

Electronic commerce data flows from external sources to internal destinations, from internal sources to external destinations, and possibly to other internal organizations. The Trading Manager manages this flow and the tracking of e-commerce data.

The Message Manager supports two distinct data flows: inbound and outbound. The inbound data flow typically represents the receipt of data in EDI or proprietary format from an external trading partner. The outbound data flow typically represents the retrieval of data from an internal application system in its native format for subsequent conversion and routing to an external trading partner.

The information from the Partner Manager is shared with the Message Manager for the validation, tracking, and routing of e-commerce data. Information about this e-commerce activity is then incorporated into the Partner Manager database for monitoring and reporting.

Trading Manager can be used for internal/external, external/internal, and internal/internal trading scenarios.

For more information about the Message Manager, see the Trading Manager documentation.

Other IBM Sterling Transformation Extender solutions

A range of products extend the out-of-the-box integration services provided by IBM® Sterling Transformation Extender and the Software Development Kit, which is included in the Design Studio product installation. There are Industry Solutions and Enterprise Application packages contain predefined interface definitions (schemas), adapters, importers, and other components designed to facilitate integration with various enterprise applications and business-to-business interfaces.

Related reference

- [Schema design](#)
-

Integration maps

Maps provide a non-programmatic approach to the development of interfaces, with prebuilt functionality for connectivity, transformation, and routing. Components can be created once and reused many times for different types of integration scenarios.

Maps support integration scenarios such as:

- Asynchronous, event-driven architecture
- Component or Enterprise JavaBean (EJB) architecture
- Synchronous, web-based architecture
- Application-embedded architecture
- Batch interface architecture

A map is made up of:

- Input and output map cards that represent the sources and targets.
- Map rules that define how the output data is built based upon business rules.
- Specifications of the resource adapter used to retrieve and send data.
- [Map behavior during execution](#)

- [Input and output card settings](#)
- [Map settings](#)
- [Configuration variables](#)

Configuration variables are aliases that resolve to specific resources within the enterprise. The variables resolve automatically to actual resources such as directory paths, database names, message queue names, and server names. Configuration variables make it possible to use the same map and system definitions in different deployment environments.
- [Reserved words and symbols](#)
- [Quotation marks](#)
- [Hex, decimal, and symbol values](#)
- [File name extensions](#)

Related concepts

- [Map design](#)

Map behavior during execution

The execution characteristics of a map are determined by the execution parameters that are configured at design time. However, these map execution characteristics or map settings can also be overridden at run time.

The behavior of a single map invocation is as follows:

1. The adapters that are configured for their respective input cards retrieve some of the input data.

The input data is validated against the schema definition of the respective input card(s). This ensures that only valid data is passed to the output building process. It is for this reason that *any* piece of *any* of the input data can be used to build *any* of the output data. All of the input data is validated and stored in the map's workspace before any output data is built. Because the entire workspace is available to the map at the time it has to build the output, any portion of the input data can be used, counted, or manipulated to generate any part of the output data.
2. After the input data has been validated, output objects can be used as inputs for a subsequent output.

You can create temporary outputs to signify map logic in certain situations. The Sink adapter can be employed to ensure that the data is not retained.

By default, the rollback behavior applies to the entire map thread level. Therefore, nothing will be committed to output nor deleted from input until all inputs have been successfully validated *and* all outputs have been successfully written. Many commit and rollback options are available within the configuration settings of each input and output card. By manipulating these settings, you can significantly change the default commit and rollback behavior of a map.

Input and output card settings

Card settings define the data object that the card represents and how the data is retrieved or routed and compiled into the map. The behavior of input and output cards configured in the compiled map can be overridden.

- Schemas represent the data source for an input card and the output data structure for an output card.
- SourceRule settings in the input card specify the data to be retrieved and how to do it, where to get it, how much of it to get, and what to do when an error occurs.
- TargetRule settings in the output card specify the data that is output, how the output data is routed, and what to do when an error occurs.

Map settings

Map settings dictate the behavior of the entire map—across *all* input and output cards.

The following map settings provide the ability to control aspects of the map's execution such as:

- [MapAudit](#)

This setting specifies options to create an audit of the map's execution, data, and execution settings, in addition to the level of detail of the audit.
- [MapTrace](#)

For troubleshooting purposes during development, this setting specifies options to create a trace and the level of detail of the trace.
- [WorkSpace](#)

This setting specifies the location (in a file or in memory) of the workspace and the amount of memory to allocate for the execution of the map.
- [Retry](#)

If compiled map files, work files, audit log files, or trace files that are needed during map execution cannot be opened, this setting specifies whether to attempt to run the map again and the interval of time between each attempt.

Almost all of the source, target, and map setting parameters can be overridden with different values at run time. This capability is available for all modes of map execution.

This allows you to change map execution parameters for different deployment environments, sources, and destinations (such as development, user acceptance testing (UAT), and production environments) without having to modify and recompile the maps.

Related concepts

- [Adapter configuration for input and output](#)

Configuration variables

Configuration variables are aliases that resolve to specific resources within the enterprise. The variables resolve automatically to actual resources such as directory paths, database names, message queue names, and server names. Configuration variables make it possible to use the same map and system definitions in different deployment environments.

Reserved words and symbols

Reserved words and symbols have a special meaning and can be used only in the manner described in this documentation. The following is the list of reserved words and symbols, and their intended use.

Reserved Word or Symbol

Intended Use

\$ Shorthand representation of the object name in a component rule or a map rule.

: Used to separate components.

:: Shorthand representation of a unique component or partition path.
For example, **A::B** refers to the unique path from the type **A** to the type **B**.

<> Separates a partition from its partitioned type

, Used to separate arguments of functions and maps

"" Used to enclose literal text strings.

() Used for a component range, to enclose function or map arguments, and to block sub-expressions in a rule

@ Separates a comment from a component.

[] Denotes an indexed member of a series.

{} Used to enclose a list of literals.

Names of Functions

Names of functions are reserved words. Names of functions can be used as type names, but cannot be used as map names.
For a complete list of functions, see the Functions and Expressions documentation.

Operators

Operators are reserved.

For a complete list of operators, see the Functions and Expressions documentation.

ANY

Represents one or more types in a component name.
ANY cannot be used in component rules or map rules.

COMPONENT

Refers to any component preceding the current one in the same component list. Used in the Type Designer to refer to preceding components of the same component list.

This reserved word can only follow the reserved word IN. For example:

COUNT (Record IN COMPONENT)

FALSE

Represents the logical false.

IN

Separates a component from an object that contains it. For example, **LineItem IN File** refers to all occurrences of the type **LineItem** in the type **File**.

LAST

Special index value that refers to the previous data object of a particular series.

NONE

Stands for the null value or non-existence of data content. Can be used in a component rule to test the presence of data. Can be used in a map rule to generate no occurrences of an output.

<space>

Separates type names. (This is an actual space.)

The names of individual types in component names are separated by spaces. Component names themselves are separated by colons. For example,

C:A B is the **C** component of the **A** subtype of **B**;

C:A B:X is the **C** component of the **A B** component of type **X**; and so forth.

<symbol>

Symbol abbreviations for entering non-printable characters. For example, <CR>.

TRUE

Represents the logical true.

WHERE

Can be used in the LOOKUP or EXTRACT function in place of the comma (,) between arguments.

Quotation marks

Although there are platform-specific limitations regarding the usage of double and single quotation marks, when adding a text string into any map rule, component rule, or command line, use the following characters for quotation marks.

Symbol	ASCII Character	Text Reference in Documentation
'	039	Single quotation mark
"	034	Double quotation mark
"	039 and 039	Two single quotation marks (for some non-Windows environments)

Using any other characters can produce unexpected results.

Hex, decimal, and symbol values

In this list, some characters in the character name are capitalized to indicate the common abbreviation for the character. For example, the <STX> abbreviation represents the Start of TeXt character.

The hexadecimval values are provided as a reference. For any literal value specified in the Type Designer Properties window, hexadecimval values must be enclosed in double angle brackets. For example, <<0A>> represents the hexadecimval value for a line feed character.

The decimal values are provided as a reference. Decimal values can be used with the SYMBOL function. For any literal value specified in the Type Designer Properties window, decimal values must be enclosed in single angle brackets. For example, <10> represents the decimal value for a line feed character.

Symbols can be inserted into map rules and component rules. You can insert them from the Symbols dialog box or by typing the hexadecimval or decimal value.

Character	Hex Value	Decimal Value	Symbol
NewLine	-	-	<NL>
WhiteSpace	-	-	<WSP>
KanjiSPace (WideSpace)	-	-	<KSP>
NULL	00	0	<NULL>
StartOfHeading	01	1	<SOH>
StartofTeXt	02	2	<STX>
EndofTeXt	03	3	<ETX>
EndOfTrans.	04	4	<EOT>
ENQuiry	05	5	<ENQ>
ACKnowlege	06	6	<ACK>
BELL	07	7	<BELL>
BackSpace	08	8	<BS>
HorizTab	09	9	<HT>
LineFeed	0A	10	<LF>
VerticalTab	0B	11	<VT>
FormFeed	0C	12	<FF>
CarriageReturn	0D	13	<CR>
ShiftOut	0E	14	<SO>
ShiftIn	0F	15	<SI>
DataLinkEscape	10	16	<DLE>
DeviceControl1	11	17	<DC1>
DeviceControl2	12	18	<DC2>
DeviceControl3	13	19	<DC3>
DeviceControl4	14	20	<DC4>
NegativeAcK	15	21	<NAK>
SYNchron.Idle	16	22	<SYNI>
EndTransBlock	17	23	<ETB>
CANcel	18	24	<CAN>
EndofMedium	19	25	
SUBstitute	1A	26	<SUB>
ESCAPE	1B	27	<ESC>
FileSeparator	1C	28	<FS>
GroupSeparator	1D	29	<GS>
RecordSep.	1E	30	<RS>
UnitSeparator	1F	31	<US>
SPace	20	32	<SP>
!	21	33	-
"	22	34	-
#	23	35	-
\$	24	36	-
%	25	37	-
&	26	38	-
'	27	39	-

Character	Hex Value	Decimal Value	Symbol
(28	40	-
)	29	41	-
*	2A	42	-
+	2B	43	-
,	2C	44	-
-	2D	45	-
.	2E	46	-
/	2F	47	-
0	30	48	-
1	31	49	-
2	32	50	-
3	33	51	-
4	34	52	-
5	35	53	-
6	36	54	-
7	37	55	-
8	38	56	-
9	39	57	-
:	3A	58	-
;	3B	59	-
<	3C	60	-
=	3D	61	-
>	3E	62	-
?	3F	63	-
@	40	64	-
A	41	65	-
B	42	66	-
C	43	67	-
D	44	68	-
E	45	69	-
F	46	70	-
G	47	71	-
H	48	72	-
I	49	73	-
J	4A	74	-
K	4B	75	-
L	4C	76	-
M	4D	77	-
N	4E	78	-
O	4F	79	-
P	50	80	-
Q	51	81	-
R	52	82	-
S	53	83	-
T	54	84	-
U	55	85	-
V	56	86	-
W	57	87	-
X	58	88	-
Y	59	89	-
Z	5A	90	-
[5B	91	-
\	5C	92	-
]	5D	93	-
^	5E	94	-
_	5F	95	-
.	60	96	-
a	61	97	-
b	62	98	-
c	63	99	-
d	64	100	-
e	65	101	-
f	66	102	-
g	67	103	-
h	68	104	-
i	69	105	-
J	6A	106	-

Character	Hex Value	Decimal Value	Symbol
k	6B	107	-
l	6C	108	-
m	6D	109	-
n	6E	110	-
o	6F	111	-
p	70	112	-
q	71	113	-
r	72	114	-
s	73	115	-
t	74	116	-
u	75	117	-
v	76	118	-
w	77	119	-
x	78	120	-
y	79	121	-
z	7A	122	-
{	7B	123	-
	7C	124	-
}	7D	125	-
~	7E	126	-
DElete	7F	127	-

File name extensions

Extension

File type description

.bak	Map source backup
.dpa	Compiled DataPower map
.dpl	DataPower session log
.fnl	Master work file
.Ixx	Input work
.log	Audit log
.mmc	Compiled map
.mme	Map build analysis results
.mms	Map source
.mopt	Map source options file
.mtr	Trace
.mtx	Temporary native schema
.omm	Backup of a 1.4.x map source
.Oxx	Output work
.tmp	Temporary map work files

Note: Zero-byte temporary map work files might not be deleted if the Map Designer application is not gracefully disconnected.

Execution

The main methods for map execution are as follows:

Execution Method

Description

command-driven

This is primarily geared towards batch processing using the IBM® Sterling Transformation Extender Command Server.

event-driven

This is a method in which systems of maps can be coordinated together by the IBM Sterling Transformation Extender Launcher to provide a scalable, distributed, multi-threaded, execution environment.

application-embedded

This is a method in which maps can be embedded in third-party C, Java, and COM applications. When used with an application server or web server, the components in the Software Development Kit can facilitate web-based integration scenarios.

- [Executing in test and production environments](#)
- [Executing on multiple platforms](#)
- [Command-driven execution model](#)
- [Event-driven execution model](#)
- [Application-embedded execution model](#)

Related concepts

- [Server components](#)

Related information

- [Deploying what you develop](#)

Executing in test and production environments

During the development process, test execution takes place in the same IBM® Sterling Transformation Extender Design Server environment in which you develop your integration solution. Testing can be performed offline, using the Design Server, even when the planned execution platform, deployment mode, or adapter assignments differ from those used in the test environment.

You can configure your environment with multiple servers at multiple sites, and on multiple platforms. Alternatively, you can select a single server as your deployment environment. For example, you can develop a system that includes all the information about how to process payments. You can then provide this map to multiple locations in your company, for execution on different server platforms. The distributed sites do not need to know anything about your development environment.

Executing on multiple platforms

A single IBM® Sterling Transformation Extender map can execute on any of the platforms that the IBM Sterling Transformation Extender products support.

Command-driven execution model

The IBM® Sterling Transformation Extender Command Server provides a command-driven capability for the execution of integration maps. This model is normally associated with batch-driven, on-demand integration processes. The methods for executing maps in command mode include the following:

- Executing a single map from a command prompt
- Executing multiple maps in sequence using a command parameter file
- Executing from the Windows-based Command Server, which provides a graphical user interface (GUI) for loading and running maps, as well as for overriding map settings
- Executing as part of a batch file or shell script
- Executing as part of a mainframe job as specified in the Job Control Language (JCL)
- Executing using commands specified in a scheduling program

By default, the execution settings of a map (such as `MapAudit` and `WorkSpace`) are compiled into the runtime object. In all modes of execution, it is possible to override almost all settings at run time. For Command Server invocation, the overrides are supplied as parameters on the command line.

- [Execution command overriding a database target example](#)

Related concepts

- [Runtime services](#)

Execution command overriding a database target example

Suppose you want to run the map `multitran.hp`, overriding some of the settings for the fourth output card, which is a database target.

Use the following command:

```
/install_dir/bin/dtx /install_dir/maps/multitran.hp -G  
-OD4R5:3B '-TRACE ERROR'
```

Option	Description
<code>multitran.hp</code>	This specifies the name of the map to be executed.
<code>-G</code>	All item properties (including size, presentation, and restrictions) are fully validated.

Option	Description							
<p>-OD4R5:3B `...'</p> <p>Value</p> <table> <thead> <tr> <th style="text-align: center;">Description</th> </tr> </thead> <tbody> <tr> <td>R5:3</td> </tr> <tr> <td>If the database resource is unavailable, attempt to make the connection five more times at three-second intervals.</td> </tr> <tr> <td>B</td> </tr> <tr> <td>If the map does not complete successfully, roll back any changes made to the database table.</td> </tr> <tr> <td>'-TRACE ERROR'</td> </tr> <tr> <td>Generates an adapter trace file containing error information only.</td> </tr> </tbody> </table>	Description	R5:3	If the database resource is unavailable, attempt to make the connection five more times at three-second intervals.	B	If the map does not complete successfully, roll back any changes made to the database table.	'-TRACE ERROR'	Generates an adapter trace file containing error information only.	This overrides the default settings compiled into the map using the database target for the data of output card #4 (-OD4) as follows:
Description								
R5:3								
If the database resource is unavailable, attempt to make the connection five more times at three-second intervals.								
B								
If the map does not complete successfully, roll back any changes made to the database table.								
'-TRACE ERROR'								
Generates an adapter trace file containing error information only.								

For more information about execution commands, see the *Execution Commands* documentation.

Event-driven execution model

The event-driven execution model relies upon the Launcher to coordinate the triggering of map threads based upon either input event or time event triggers. With its ability to define complex event-triggering criteria, combined with a robust set of adapters that provide listener interfaces to detect these events, the Launcher is the cornerstone for providing real-time, event-driven integration.

- [Launcher](#)
- [Launcher Architecture](#)
- [Audit and transaction logging](#)

Launcher

The IBM Sterling Transformation Extender Launcher provides a real-time, event-driven model for executing maps and systems of maps defined in the Integration Flow Designer. The Launcher runs on a server platform and manages one or more systems concurrently. Systems are asynchronous, multi-threaded processes, whose atomic units are transactional maps. Systems can execute in a distributed environment in which a Launcher resides on each server. When events occur, the Launcher is notified and starts maps based upon that notification. Distributed data is received or routed during map execution.

- [Triggers and watches: asynchronous communication](#)
- [Synchronous coordination](#)

Related concepts

- [Runtime services](#)

Triggers and watches: asynchronous communication

The set of events that initiate a map within a system is called a *watch*. Each event that contributed to the initiation of a map is called a *trigger*. Triggers are asynchronous events resulting from time or source state changes (for example, the creation of a file in a directory or a message appearing on a queue). The Launcher starts maps based upon triggers defined in the Integration Flow Designer. In some cases, there may be multiple triggers that must collectively exist before a map is initiated. The Launcher manages the coordination of these events to ensure that the correct set of circumstances has occurred before a map is initiated. Although each instance of a map is considered a separate watch, the Launcher must also manage the resource interfaces used by other maps and other instances of the same map to properly synchronize these asynchronous events.

Data sources using database adapters can serve as input event triggers that are defined in the Database Interface Designer, enabled in the Integration Flow Designer, and executed by a Launcher.

The listener interface, supported by various adapters, handles triggering through database updates and messages arriving on queues.

Synchronous coordination

Synchronous interaction between steps within the data integration process may sometimes be required. For example, it might be important to complete a transaction only when an acknowledgment of an action is received. Synchronous interaction can be accomplished in different ways:

- Communicate with other maps from a map rule during the data transformation process. Using a map rule in this way, the Launcher manages coordination among maps to avoid deadlocked situations and to ensure that shared resources are reliably accessed or deleted.
- Synchronize coordination to interact with other applications using a resource adapter for a data source or target. For example, when a file is sent to a server using the FTP target adapter, the FTP server response is received before the mapping process completes.

Launcher Architecture

The Launcher architecture is an event-driven software model executing maps. Systems contain maps supporting a system data flow that is specified by using the Integration Flow Designer.

An Launcher runs on a server platform and can concurrently manage one or more systems. Launcher functionality is based upon the following premises:

- The event manager sub-component of the Launcher controls the initiation of maps based upon the sources to which a map subscribes.
 - The resource manager sub-component of the Launcher synchronizes shared data and timing interfaces among heterogeneous sources and targets.
 - Maps publish data that result from the content transformation of source data.
 - Maps are transactional processes, with each map having its own error detection and recovery procedures.
 - The Launcher runs as a single process with multiple threads. An associated process monitors input source resource changes.
 - [Launcher methodology](#)
 - [Business Advantages of the Launcher](#)
-

Launcher methodology

Whenever the Launcher detects a new event trigger, the watches associated with that trigger are either:

- initiated, thereby causing the map to execute
- placed in an initiation pending state, awaiting the completion of all specified triggers
- maintained in an initiation-pending state until operating system resources (such as threads) are available

The Launcher initiates an individual map thread for each successful combination of specified triggers. Multiple instances of the same map component can be invoked and executed simultaneously if the Launcher is configured to take advantage of concurrent multi-threading of map execution for performance. Alternatively, if strictly sequential execution architecture is required (FIFO); configuring the Launcher accordingly can accommodate this. This concurrent-versus-sequential capability can be configured at the individual map component level. As a result, maps contained in a system could be concurrent, while others could be set to execute sequentially.

Business Advantages of the Launcher

Some of the main business advantages to using the Launcher are:

- automatic system execution
 - multi-threaded environment for multiple transactions, which allows linear scaling across multiple CPUs
 - process prioritization
-

Audit and transaction logging

The totally flexible architecture of IBM® Sterling Transformation Extender products allows the design and development of sophisticated transaction and audit logging using standard functionality. This is supported by the complete range of resource adapters and the many tools available in the complete line of IBM Sterling Transformation Extender products.

Maps are transactional processes and each map can be configured to have its own error detection and recovery procedures, as well as auditing capability. The resulting audit logs themselves can be mapped, as input data, to dedicated execution logging, error logging, and exception handling routines.

Typically, audit and transaction logging processing is achieved by designing reusable map systems to process audit logs resulting from map execution.

Application-embedded execution model

The mechanism that is used for this model is basically the same as the one providing runtime instructions for a map's execution from the command prompt. Overrides are submitted to control the map's execution. The difference is that these overrides are programmatically supplied, in memory, rather than on the command line. However, even under IBM® Sterling Transformation Extender Command Server execution, the overrides specified on the command line are passed by the Command Server, in memory, to the core execution code, which is embodied in the various APIs. Note that, regardless of the execution model employed, the same underlying IBM Sterling Transformation Extender technology prevails.

The benefit of embedding a map within an application is that all of the power of the standardized transformation and connectivity capabilities of the IBM Sterling Transformation Extender products can be fully leveraged. Rather than needing to write Java, C, or C# code to perform complex transformation, connectivity, and routing, maps can be fully leveraged to perform these tasks. Again, as with other execution models, the same inherent IBM Sterling Transformation Extender technology components are used: maps. This means that a more unified skill set is applicable across all integration models, thereby reducing the overhead of writing the code to perform transformation, routing, and connectivity to a minimum. This allows more time to concentrate efforts upon developing the business logic in the application layer code, resulting in a much quicker implementation. In addition, because the source components, such as schemas and map source files, are not procedural code listings, but rather visual, object-based components, the ongoing maintenance of these components is more straightforward, using the extensive capabilities provided by the Design Server.

APIs are provided for the following applications:

- [Applications](#)
- [Java applications](#)
- [COM applications](#)
- [C# applications](#)
- [J2EE applications](#)

C applications

IBM Sterling Transformation Extender includes two APIs available to C programmers: the TX Programming Interface and the z/OS Platform API. Customers creating new applications should use the TX Programming Interface, C API. The z/OS Platform API continues to be supported for legacy applications. However, any new features or functionality will be available only in the TX Programming Interface, C API.

Java applications

There are several ways in which a Java application can run a map:

- By using the TX Programming Interface Java API
- By using the TX Programming Interface RMI API

With the RMI API, a map (and the underlying native core code) can be executed out of process. Also, with RMI, IBM Sterling Transformation Extender can run on any platform that supports the Java RMI API. (And this is not necessarily the same platform as the ones upon which Java Application Servers run.) This enables maps to be executed by an Enterprise JavaBean containing business logic and subsequently to be deployed to an application server. Examples of this are provided.

Java-based applications that invoke a map through the API can also take advantage of using the Java Class Adapter to access additional Java objects. For EJB functionality, access to serializable objects is supported. From the user's Java Application or EJB, objects are serialized using Java Serialization and passed (using a STREAM adapter override) to input cards of a map. Within the map, the Java Class adapter can be used to deserialize an object (and store it in the Object Pool). Access to fields and methods on Objects stored in the Object Pool is permitted using the Java Class adapter. (Objects are also accessible using the JNDI and JMS adapters.) To pass objects back to the user's application from a map, objects are serialized using the Java Class adapter. The resulting byte[] is passed back to the calling EJB using a STREAM adapter override. The Java application or EJB re-constructs the Object using Java Serialization.

COM applications

For customers using Microsoft's Component Object Model (COM) technology, the IBM Sterling Transformation Extender Programming Interface (TXPI) for COM provides the functionality to run an IBM Sterling Transformation Extender map from scripting languages such as VBScript and Jscript. This also allows IBM Sterling Transformation Extender maps to be executed from Active Server Pages.

C# applications

Using the TX Programming Interface, COM API, and the interoperability library provided in the Software Development Kit (SDK) examples, a C# application running on .NET can also run an IBM Sterling Transformation Extender map for transformation.

J2EE applications

Java 2 Platform, Enterprise Edition (J2EE) defines the standard for developing component-based, multi-tier, enterprise applications. Applications built using J2EE can use the various Java-based APIs described.

Related concepts

- [Software Development Kit Application Programming Interfaces \(APIs\)](#)

Examples

Another source for learning about IBM® Sterling Transformation Extender products is by running the examples that are provided with the product. The examples illustrate how to use IBM Sterling Transformation Extender product components to accomplish specific tasks and can prompt ideas for achieving your goals. The examples are located at *install_dir/examples*.

Tivoli License Manager

IBM Tivoli License Manager (ITLM) enables you to monitor the usage of IBM (and other) software products. ITLM provides you with the following software auditing functionality:

- Monitor the licenses used across different machines.
- Help keep unnecessary licenses to a minimal.
- Guard against software license compliance problems.

IBM® Sterling Transformation Extender supports ITLM. To find out more about using ITLM to monitor usage of your IBM software, see the [IBM License Metric Tool](#) knowledge center.

What's new in V11.0.1.0

This is what's new in this version of IBM Sterling Transformation Extender:

Installation

IBM Sterling Transformation Extender can be installed on Linux platforms in three modes: Native, Docker, and Podman. Podman is the new installation mode supported on Linux platform in this version of product.

On the Red Hat Linux distribution, the IBM Sterling Transformation Extender installation supports Podman installation types, while on other Linux distributions (like Ubuntu, SuSE), IBM Sterling Transformation Extender supports both Docker and Podman container engines.

Nodes

New nodes

Format Converter node

The Format Converter node can be used to quickly convert data from one format to another.

The Format Converter node supports these formats:

- CSV
- Delimited
- JSON
- YAML
- XML

Adapters

New adapter

Redis adapter: The Redis adapter provides access to Redis databases for storing and retrieving the keys and their values as part of data transformation. The Redis adapter is used in maps and flows to set and get key-value pairs in the Redis databases. Redis adapter is supported with all runtime map and flow execution environments.

Updates to existing adapter

- Oracle adapter: Oracle adapter is updated with multi-row table insert support. A new command line option **-INSERTALL/-IA** enables multiple rows of data to get inserted into an Oracle table at once instead of one row at a time. Throughput can be increased several-fold with multi-row insert support under heavy load transaction scenarios.
- Apache ActiveMQ adapter: Apache ActiveMQ adapter has added support for standard JMS headers and user-defined custom properties in the messages produced to the ActiveMQ broker, and while messages consumed from the ActiveMQ broker.
- RabbitMQ adapter: RabbitMQ adapter is now supported in Design Studio and Integration Flow Designer. RabbitMQ adapter can be invoked from maps running under Launcher systems during runtime.

Maps

Map Audit enhancement

Input validation errors are included in the Map Audit log when SummaryAudit setting is turned ON in the map settings. In environments where map trace is not allowed to capture the validation errors, a map audit log having detailed error information would be the only way to identify and resolve data errors.

Map Audit enhancement is not available in z/OS Batch and CICS environments, but z/OS USS environment generates data validation errors in the map audit log when SummaryAudit is turned ON.

Launcher

Cooperative Listener directory permissions

Cooperative listener directories' permissions have been adjusted to allow multiple launchers running under cooperative mode to be able to use the same directories. Instead of bailing out with errors due to insufficient permissions while writing onto them by launchers running under different accounts within the same group on Unix platforms.

ini2yaml migration utility

A new migration tool, ini2yaml, has been added for this release which facilitates the configuration information in dtx.ini file to config.yaml file. Customer environments that use several configuration files, ini2yaml tool eases the migration effort and reduces errors during migration.

Design Studio

Design Studio has been enhanced to run on the newer Eclipse 4.31 platform. This newer Eclipse platform brings in a newer look and feel, advanced features of the platform for users. Eclipse based applications integrated with Design Studio can take advantage of the newer Eclipse 4.31 platform and Java 17 support provided in this version of the product.

Design Server

Transport Layer Security (TLS) support

Transport Layer Security (TLS) client settings can be configured for outbound connections from the Design Server to Runtime Servers. See the mapping server.outbound.ssl in config.yaml. Use of these settings is not required if the previous behavior for outbound connections is sufficient. These settings are intended to allow secure connections to runtime installations including those that use a customized TLS configuration. Mutual TLS (mTLS) is supported when values are provided for config.yaml keys server.outbound.ssl.certFile and server.outbound.ssl.keyFile. These values are optional when mTLS is not used.

Java

Java 17 support

IBM Sterling Transformation Extender distributes IBM Semeru Java 17 Certified Edition Runtime Environment with all supported platforms of the product. All Java based components and tools in the product support Java 17. Removed packages from Java 8 are substituted with Jakarta API packages in the components for this product release. Applications that use IBM Sterling Transformation Extender Java API or Java RMI API are recommended to recompile their applications with Java 17.

- [**What's new in Packs V10.2.3.0**](#)

This is what's new in this version of packs:

- [**What's new in Packs V10.2.0.1**](#)

This is what's new in this version of packs:

What's new in Packs V10.2.3.0

This is what's new in this version of packs:

What's new in Pack for Financial Payment v10.2.3.0

SWIFT component:

- Support is provided for 2024 SRG Standards
- Corresponding updates are as follows:
 - SWIFT type trees
 - LMF maps and trees
 - Java Validation Component
 - MX schemas, type trees, and validation framework
- For TX version 11.0.1, and later
 - IFD deployment script will need saved .msl to TX_INSTALL_DIR/system folder.
 - Prop file will no longer have sub-directories references under TX install directory.
 - As a pre-requisite, users will need to edit the property file to replace generic PACK_INSTALL_DIR with the actual location where the ZIP distro was unzipped.
 - Utility jvcsetup.sh in swiftJVCconfigIBM.targz is for Docker, and Podman based installations.
- For Design Server installation:
 - Copy the jvalccyy.jar and jvcwrap.jar into the directory set in the config.yaml server.persistence.libs, by default, this is set to /opt/tlxlibs. Then restart the tx-server, for example ./ITX, stop and then start ./ITX
- Configuring JVC on Design Server:
 - After importing the swiftMTCompliance.zip project, the jvalccyy.prop file must be copied into the workdir directory under directory set in the config.yaml server.persistence.files, default settings is /opt/tlxfiles/workdir.
 - On the project, update the map rule on the validate_message map, on output card #1 item JvalPropPath. Change the map rule from =NONE to the location of the jvalccyy.prop.
 - For example:
 - On UNIX: ="DS_DATA_DIR/workdir"
 - On WINDOWS: ="DS_DATA_DIR\workdir"Where DS_DATA_DIR refers to the Design Server data directory, set in the config.yaml server.persistence.files previously TX_FILE_DIR.
- Post EAP updates:
 - Enforce field level validation related to error code T78 on tag 81J, 91J, and 96J on SWIFT MT common messages MTn92, MTn95, and MTn96
 - Support the new SWIFT MX Funds messages
 - reda.004.001.07 - will replace reda.004.001.06
 - setr.006.001.05 - will replace setr.006.001.04
 - setr.012.001.05 - will replace setr.012.001.04

NACHA component:

Support 2024 NACHA Operating Rules - Supplement #1-2024.

FIX component:

Support FIX 5.0 SP2 Extension Packs (FIX Latest) up to EP291.

Miscellaneous:

- For ITX 11.0.1, build JVC jars using IBM Semeru JAVA 17.
- ZIP distros are digitally signed.

Deprecated components:

- SWIFT MT-MX translation example maps.
- SWIFT Sterling B2B Integration examples.

What's new in Pack for Financial Payment Plus v10.2.3.0

CBPR+ Translation:

- For Design Server installation:
 - Copy the following jars into the directory pointed to the config.yaml server.persistence.libs by default, this is set to /opt/tlxlibs, then restart the running application ./ITX stop and then ./ITX start.
 - jnodes0.jar
 - jackson-core-n.n.n.jar
 - jackson-annotations-n.n.n.jar
 - jackson-databind-n.n.n.jar
- Updated translation frameworks to support CBPR+ Translation guide version 1.5.2 bug fixes.

- Additional CBPR+ Translation scenarios:
 - MT103 RETN <=> pacs.004.001.09
 - MT202 RETN <=> pacs.004.001.09
 - MT205 RETN <=> pacs.004.001.09

CBPR+ Validation:

- Support version SR2023, SR2024 schemas based on MyStandards Readiness CBPR+ Readiness Portal, date of publication 24 April 2023 and 12 December 2023.
- New 2024 CBPR+ messages:
 - camt.105.001.02
 - camt.105.001.02 mlp (multiple charges)
 - camt.106.001.02
 - camt.106.001.02 mlp (multiple charges)
 - camt.109.001.01 (already included in version 10.2.2.0)

EBA E1S1 Validation:

- Support version EBACL_E1S1_UG2024 (MR2019) based on SWIFT MyStandards Readiness EBA Portal, date of publication 01 August 2024.

NBOR ReGIS Validation:

- New version v1.1 based on SWIFT MyStandards Readiness NBOR ReGIS Portal, date of publication 20 August 2024.

T2 CLM Validation:

- New version UDFS T2 R2024.JUN schemas based on SWIFT MyStandards Readiness TARGET2 CLM Portal, date of publication 11 June 2024.

T2 RTGS Validation:

- New version UDFS T2 R2024.JUN schemas based on SWIFT MyStandards Readiness TARGET2 RTGS Portal, date of publication 11 June 2024.

T2 CoCo Validation:

- New version UDFS CRDM T2 R2024.JUN schemas based on SWIFT MyStandards Readiness TARGET2 CoCo Portal, date of publication 11 June 2024.

T2S Validation:

- New validation frameworks (schema only) based on SWIFT MyStandards TARGET2 T2S Portal version UDFS T2S R2024.JUN date of publication 9 June 2024.

SWIFT GPI/SWIFT Go Validation:

- New version schemas based on SWIFT MyStandards Readiness Swift GPI Portal, date of publication as below:
 - trck.001.001.03, trck.002.001.02
 - gCCT/gCOV/gFit - date of publication 15 March 2024
 - swiftGo - date of publication 15 January 2024
 - trck.003.001.02 - date of publication 05 March 2024
 - trck.004.001.02 - date of publication 14 December 2023
 - trck.005.001.02 - date of publication 12 January 2024

SIC RTGS Validation:

- New validation frameworks (schema only) based on SIX Interbank Clearing (SIC) Ltd Platform Releases 4.11 (RTGS services SIC and euroSIC) and 5.1 (SIC IP service).

Miscellaneous:

- Validation frameworks enhanced reporting to handle messages with unsupported versions.
- Support for XML messages with diacritic characters.
- New utility map for parsing the validation reports (under tools folder).
- Design Server flow redesign to standardize and streamline examples.
- For ITX 11.0.1, built CBPR+ Translation jar using IBM Semeru JAVA 17.
- ZIP distros are digitally signed.

Deprecated components:

- CHAPS L4L validation

What's new in Pack for Healthcare v10.2.3.0

HIPAA

- Support for 277DRA compliance checking (5010X364).
- Support for 6020X313 and 6020X315 transactions to support CMS-0053-P for claims attachments.
- New parameter added for 277CA/277DR acknowledgements to control whether the STC segment at all 277CA and 277DR loop levels reflects an error at that level or lower and includes a valid entity code instead of the generic Ack/Receipt code.
- Support in Type 7 checking to allow return of invalid element and sub element values.
- Ability to include ISA Sender and Receiver IDs, ISA Control number, GS Sender and Receiver Codes and GS Control number in the transaction data passed to the Type 7 user exit.
- Code List updates.

NCPDP

- The Annual Code lists updates.
- Additional validation in NCPDP PACDR trees versions 51-53.
- Support for Java 17 in validation exit jar file.

HL7

- FHIR Mapping examples updated to use JSON schema, and to also create a JSON response bundle based on X217 278 Response.

What's new in Pack for Supply Chain EDI v10.2.3.0

New configuration settings:

- **Edicc.ini file:**

This file contains configuration for the edisvu exit location and z/os processing flag. A new configuration for z/os processing has been added to suppress the run map log. This setting could show a performance enhancement for large file processing with reduced I/O.

- **XML and JSON configuration files:**

- SKIPTVALIDATION configuration:

There are three main processes performed during the compliance checking as follows:

- Standard Type Tree validation
 - Structure checking
 - Segment checking

The Standard Type Tree validation uses the full standard to determine if the input file contains errors. If there are no errors, no further validation is needed. If there are errors, the second and third processes are executed. This first process can take a considerable amount of time depending on the standard and the message type. The SKIPTVALIDATION setting will skip the first process and always execute the structure checking and segment checking and eliminate the amount of data parsing and restarts. It also eliminates the ttval run map execution.

Tooling:

With the extensive list of EDI standard versions supported and the compliance checking process to validate the structure of the messages/transactions, the structure validation file has become large. New utilities in the EDIFACT and X12 components can be used to reduce the size of the structure validation file by selecting standard versions and message/transaction type being used to eliminate unused records.

In addition, a new referenced structure file can be found in the mapsandschemas folder to be used with input card #7. This file contains a reference record for each message/transaction that identifies each standard version a particular structure record can be used with. This avoids duplication or records when the message/transaction structure does not change from release to release.

X12 and X12U component:

- Contains the X12 Standard currency updates with version 8050.

EDIFACT component:

- Contains the EDIFACT Standard currency updates with D23A version.
- Configuration setting to allow validation for inbound EDIFACT CONTRL messages.

Pack installation:

ITX has configured CISO Code signing infrastructure to verify the ITX software (including the Industry Packs) and images in the HCL lab based on the guidelines provided by IBM. IBM Code Signing service holds the certificate used to sign the ITX software and images.

What's new in Packs V10.2.0.1

This is what's new in this version of packs:

What's new in the Pack for SAP V10.2.0.1

Support for the latest versions of IBM Sterling Transformation Extender:

The Pack for SAP Applications v10.2.0.1 includes installation packages for all currently supported releases of TX at the time of the release, which includes ITX versions 10.1.0, 10.1.1, 10.1.2, 11.0.0, and 11.0.1.

SAP S/4 HANA Adapter option to use API key authentication:

In this version, the SAP S/4 HANA Adapter includes an option to use API key authentication. Before, the API key needed to be added to the Base Headers. Now, there is a separate option and field for API key authentication, simplifying the process of defining a connection.

Configuration

This section describes the configuration before or during the installation.

- [Data Directory](#)

The Data Directory is where you maintain a list of directories that can be used by the data loading and optionally, receive and store files transmitted Download.

- [The IBM Sterling Transformation Extender configuration file config.yaml](#)

- [Customizing the Trace](#)

This section describes how effectively you can configure custom trace settings for IBM Sterling Transformation Extender Rest execution. This will allow you to make the level of logging and information displayed according to your needs, enhancing the troubleshooting and analysis process.

- [How to verify the Config.yaml file](#)

This section describes how to verify the validity of your config.yaml file after making changes in it.

- [Enhancing Security, preventing Server Side Forgery Requests attacks](#)

Data Directory

The Data Directory is where you maintain a list of directories that can be used by the data loading and optionally, receive and store files transmitted Download.

Data Directory is the location for IBM Sterling Transformation Extender writable files after the installation.

During the Windows installation, you can use the current behavior by making the Data Directory same as an installation directory.

On non-Windows platforms, the Data Directory will be the same as the installation directory.

Directories and files under the Data Directory

- **Config**

Contains config.yaml, .bin and .property files for legacy java tools, MRC and MRN files default location for resource registry, MKF (Master Key File) for encryption.

- **Examples**

User who chose a read-only installation folder can run the examples that are shipped.

- **Logs**

Contains log files that include redis and tomcat logs.

- **Snapshots**

It is a new directory for the Legacy Launcher Monitor and to create the snapshot (MSS) files. Prior to this release, files were written to the root of the installation directory.

- **Systems**

Default location for Launcher System (MSL) files.

The IBM Sterling Transformation Extender configuration file config.yaml

For the IBM Sterling Transformation Extender installation, configuration information is provided in config.yaml. This file contains several YAML mappings. The value of a key of such a mapping may itself be a YAML mapping. In this documentation, JSON Pointer strings are used to refer to the value of a key value of a YAML mapping. For example, /mongo/auth/password refers to the value of the MongoDB password which should be used to access the MongoDB database which IBM Sterling Transformation Extender uses for design data. Alternatively, dot notation may be used for the same purpose, for example mongo.auth.password.

- [Configuration file syntax validation](#)

- [Configuration file expansion expressions](#)

Two kinds of expansion expressions allow values to be provided in the config.yaml configuration file from another source. An expansion expression is allowed as the value of a key-value pair of a YAML mapping.

Configuration file syntax validation

For each execution, the flow command server attempts to read the installation configuration file. This behavior allows the flow command server version reporting option -v to be used to validate a configuration file for syntax errors. If present, a YAML syntax error will be reported when the following command is executed. See the "Flow Command Server" section in the "Runtimes" section for more information about using the flow command server. The following command demonstrates use of the -v option.

```
flowcmdserver -v
```

Configuration file expansion expressions

Two kinds of expansion expressions allow values to be provided in the config.yaml configuration file from another source. An expansion expression is allowed as the value of a key-value pair of a YAML mapping.

- [Environment variable expansion](#)

- [Master Key File \(MKF\) expansion](#)

Environment variable expansion

Environment variable substitution is performed at run-time only. An expansion expression of the following form is replaced with the value of the named environment variable: \${env:<environment variable name>}

For example, the value \${env:DTXHOME} would be replaced with the value of the environment variable DTXHOME.

If the environment variable is not defined, then the expansion expression is replaced with the empty string.

Master Key File (MKF) expansion

To allow the inclusion of sensitive data, a Master Key File (MKF) encryption expansion expression is replaced at run-time and install-time with its decrypted value. The MKF encryption expansion expression should have been produced using a tool which is provided for this purpose. See the following description of the available tools given below.

MKF encryption expansion expressions have the following form:

```
 ${enc_mkf: mk_id=<master key ID>, check_id=<encryption check data>, iv=<encryption initialization vector>, data=<encrypted data>}
```

The master key which is used for encryption determines the encryption algorithm which is used to encrypt the data.

A default MKF file is provided with an IBM Sterling Transformation Extender installation. A new MKF file can be created and used instead of the default file. See the sections on MKF files in the “Variables” documentation for information about creating a master key file.

A configuration property value can be encrypted in both Windows and Linux installations of IBM Sterling Transformation Extender. Encryption can be performed in place. This means that the value is read from the config.yaml file and replaced with an MKF encryption expansion expression. Alternatively, a value to be encrypted can be provided in a command line. The encryption tools use a JSON Pointer string to refer to the configuration property value to be encrypted.

For a Windows installation:

1. Open an administrator shell, and change the working directory to <installation root>/DesignServer
2. Execute the following script using either the -inplace option or not.
 - a. **encryptproperty.bat -inplace <JSON Pointer config.yaml path>**
 - b. **encryptproperty.bat <JSON Pointer config.yaml path> <string to encrypt>**

For a Linux installation:

1. Under an appropriate user, open a shell.
2. Execute command IBM Sterling Transformation Extender encrypt with the appropriate arguments. For example, from the installation root directory, execute one of:
 - a. **./IBM Sterling Transformation Extender encryptproperty -inplace <JSON Pointer config.yaml path>**
 - b. **./IBM Sterling Transformation Extender encryptproperty <JSON Pointer config.yaml path> <string to encrypt>**

Customizing the Trace

This section describes how effectively you can configure custom trace settings for IBM Sterling Transformation Extender Rest execution. This will allow you to make the level of logging and information displayed according to your needs, enhancing the troubleshooting and analysis process.

- [Customizing the Trace for IBM Sterling Transformation Extender Rest Execution](#)

Customizing the Trace for IBM Sterling Transformation Extender Rest Execution

About this task

To customizing the trace for IBM Sterling Transformation Extender Rest execution, complete the following steps:

Procedure

1. Locate the config/config.yaml file in the data directory of the product. The data directory is configurable on Windows during the installation. On Unix, it is defined with **DTX_DATA_DIR** environment variable.
2. Open the config.yaml file using a yaml editor.
3. Find the /rest/logging path in the config.yaml file. This is where you can configure the trace settings for the REST API execution.
4. To enable verbose logging, set the trace level to "ALL" or "TRACE" by updating the value associated with the trace level setting. For example:

```
/rest/logging:  
  trace_level: ALL
```

For informational messages, set the trace level to "INFO":

```
/rest/logging:  
  trace_level: INFO
```

To trace errors only, set the trace level to "ERROR":

```
/rest/logging:  
  trace_level: ERROR
```

Note: Trace level specifies the verbosity level of execution logs. The options range from least verbose to most verbose: ALL, TRACE, INFO, ERROR, and NONE. Also, to include a header that explains each row in the trace file, set the Header value as true.

5. Customize the information displayed in the log file by selecting desired columns. Use the /rest/logging/columns path to include or exclude specific trace file columns. For example, if you want to include specific columns:

```
/rest/logging/columns: (This section specifies which columns of information to show in the log entries. Each column has  
its own sub-settings, if any values set to false, so that column would not be shown.)  
  - time (Displays the time of the log entry in UTC with milliseconds)  
  - uuid (Displays the UUID of each instance.)  
  - rc (Displays the return code associated with the log entry.)  
  - rcText (Displays a text description of the error code.)  
  - msgID (Displays the ID of a trace message.)  
  - flowName (Displays the flow, node, or adapter name that generated the message.)  
  - msg (Displays the actual trace message.)
```

```
-funcName (Displays the function name)
- sourceInfo (Displays source file and line information).
```

6. Save the changes to the config.yaml file.
7. The trace file will be saved in the product logs directory (<DATA_DIR>/logs) with a naming format such as flowexecredis_<PID>_<DATE>.log.
Note: Make sure that any modifications you make to the config.yaml file is valid. To verify the config.yaml file, see [How to verify the config.yaml file](#) for detailed instructions.

How to verify the Config.yaml file

This section describes how to verify the validity of your config.yaml file after making changes in it.

When editing the config.yaml file, it is crucial to ensure that the modifications you make are valid. To verify the validity of your config.yaml file after making changes, follow these steps:

1. Open a command prompt or terminal window.
2. Run the following command:

```
flowcmdserver -v
```

This command will check the configuration file for any errors and provide feedback on its validity.

3. If the config.yaml file is valid, you should see an output similar to the following:

```
flowcmdserver.exe -v
IBM Sterling Transformation Extender
11.0.0.0(16)
(c) Copyright IBM Corporation 2006, 2023 All rights reserved.
Configuration loaded from <Program Data Folder>\config\config.yaml
```

This indicates that your configuration file is valid and successfully loaded.

4. If there are any issues with the config.yaml file, you will receive an error message similar to the following:

```
flowcmdserver.exe -v
IBM Sterling Transformation Extender
11.0.0.0(16)
(c) Copyright IBM Corporation 2006, 2023 All rights reserved.
Failed to load yaml file. illegal map value line=53 column=6
Configuration file is C:\IBM\TransformationExtender_11.0.0\config\config.yaml
```

This error message will provide details about the specific problem in the configuration file.

Note: Make sure to carefully review any error messages and correct the issues in your config.yaml file accordingly. This step is essential to ensure the proper functioning of your system after making changes.

Enhancing Security, preventing Server Side Forgery Requests attacks

Server Side Forgery Requests (SSFR) are attacks that involve port and network scanning, as well as probing at the backend of a web application, using the application's built-in capabilities. ITX Design Server has two capabilities that can be misused to perform SSFR attacks:

1. When an end user defines the ITX Design Server REST API connection parameters while designing REST API service endpoints (services are designed using the ITX Design Server's services functionality), SSFR can be performed by manipulating the external endpoint connection parameters. This can be done by pointing the REST URL connections to intranet addresses or by pointing to external URL internet addresses.
2. When an end user defines runtime server connections (runtime server connections are designed using the ITX Design Server's servers functionality, and the created servers are used to deploy flows, maps, and endpoints), SSFR can be performed by manipulating the server URL. This can be done by pointing the URL to intranet addresses or by pointing to external URL internet addresses.

To mitigate SSFR attacks, ITX Design server offers protection through fine tuning of config.yaml settings:

```
"""
...
server:

# Defines minimum timeout for the runtime server connection test
# This feature prevents port scans using SSF
testConnectionMinimumTimeout: 6

runtime:
  #Deployment is allowed only to this list of runtime servers
  servers:
    - localhost:8443
    - localhost:9443
    - localhost:8080
    - tx-rest:8443
    - tx-rest:8080

  services:
    endpoints:
      # List any addresses that should be blocked in service endpoint URLs
      # Block local and private network requests to prevent SSF attacks
      blacklist:
        - "0.0.0.0-255.255.255"
        - "127.0.0.0-127.255.255.255"
        - "169.254.0.0-169.254.255.255"
        - "100.64.0.0-100.127.255.255"
```

```

    - "10.0.0.0-10.255.255.255"
    - "192.168.0.0-192.168.255.255"
    - "172.16.0.0-172.31.255.255"
    - "localhost"
#If set to true, allow only connections to endpoints that are whitelisted
enable_whitelist: false
whitelist:
    - "127.0.0.1"
    - "localhost"
    - "myserver3:port"
...
...

```

Installation

Everything you need to know to install IBM Sterling Transformation Extender.

- [Windows installer](#)

This documentation describes how to install the IBM Sterling Transformation Extender using the Windows Installer.

- [Linux installation](#)

You have the choice of three installation types: Native, Docker, and Podman.

- [Using the OneDB wire listener as a MongoDB server](#)

The OneDB wire listener may be used in place of a MongoDB server when it is suitably configured.

- [IBM Sterling Transformation Extender design data](#)

The documentation provided here describes data backup, data restoration, and data migration of IBM Sterling Transformation Extender design data.

Windows installer

This documentation describes how to install the IBM Sterling Transformation Extender using the Windows Installer.

- [IBM Sterling Transformation Extender Server Components](#)

The IBM Sterling Transformation Extender windows installation has the following components:

- [Installation prerequisites](#)

If you have a previous version of IBM Sterling Transformation Extender installed, you must uninstall it completely. IBM Sterling Transformation Extender windows installation does not support multiple IBM Sterling Transformation Extender installations on a single computer.

- [Installation procedures](#)

- [Starting and Stopping IBM Sterling Transformation Extender application](#)

After Installing, to start and stop IBM Sterling Transformation Extender , follow the steps:

- [Accessing IBM Sterling Transformation Extender](#)

Instructions for accessing the IBM Sterling Transformation Extender.

- [Customizing the installation](#)

IBM Sterling Transformation Extender customization instructions.

- [Customizing the Redis and IBM Sterling Transformation Extender Databases](#)

IBM Sterling Transformation Extender installation includes Redis server. If you want to use a preinstalled Redis server, you must customize your IBM Sterling Transformation Extender installation.

- [Customizing IBM Sterling Transformation Extender URL and Selecting HTTPS protocol](#)

- [Uninstalling IBM Sterling Transformation Extender](#)

Instructions for uninstalling IBM Sterling Transformation Extender.

- [Troubleshooting the installation](#)

In the case of IBM Sterling Transformation Extender Install/Start/Stop errors, review the log files from this folder:

- [Troubleshooting IBM Sterling Transformation Extender installation process](#)

If IBM Sterling Transformation Extender installation fails, verify that ports listed in the table are available on your machine.

- [Manually updating Apache Tomcat 9 for the IBM Sterling Transformation Extender application](#)

This procedure defines the steps to manually update the installed Apache Tomcat 9 software for the IBM Sterling Transformation Extender application. If the currently installed version of Apache Tomcat 9 has a security issue that is resolved in a later version, you can follow these steps to update it.

IBM Sterling Transformation Extender Server Components

The IBM Sterling Transformation Extender windows installation has the following components:

- An IBM Sterling Transformation Extender design time, used to design Schemas, Maps and Flows. The IBM Sterling Transformation Extender design time is a Web based application, runs on a Node.js Application Server and on a Tomcat Application Server. IBM Sterling Transformation Extender design time is also referred by the Windows IBM Sterling Transformation Extender installer as IBM Sterling Transformation Extender Server.
- An IBM Sterling Transformation Extender runtime (REST API), used to run Maps and Flows, runs on a Tomcat Application Server.
- An IBM Sterling Transformation Extender design time MongoDB Database that stores User accounts and IBM Sterling Transformation Extender Map and Flow designs. The database requires the MongoDB server to be available before installation is started. Optionally OneDB can be used instead of MongoDB.
- An IBM Sterling Transformation Extender design time execution engine, the runtime that runs designed Maps and Flows directly from the IBM Sterling Transformation Extender UI.
- A Redis server, is used by the IBM Sterling Transformation Extender runtime and design-time for communication purposes. The IBM Sterling Transformation Extender windows installer installs Redis server as default option. IBM Sterling Transformation Extender can be configured to use preinstalled Redis server.

A Windows IBM Sterling Transformation Extender installation typically installs runtime and design time, you can customize the installation and select to install IBM Sterling Transformation Extender runtime or design time.

Installation prerequisites

If you have a previous version of IBM Sterling Transformation Extender installed, you must uninstall it completely. IBM Sterling Transformation Extender windows installation does not support multiple IBM Sterling Transformation Extender installations on a single computer.

Before you install IBM Sterling Transformation Extender design time on a Windows computer, verify that MongoDB Server is available on the network (Instead of MongoDB, OneDB with MongoDB json listener can be used to store the IBM Sterling Transformation Extender database). The IBM Sterling Transformation Extender windows Installer will present you the MongoDB connection configuration dialog, and test the MongoDB connection with the specified parameters, the installation will not continue if MongoDB connection test fails.

Note: Refer to the Recommendations for configuring the OneDB wire listener documentation, if a OneDB wire listener will be used instead of a MongoDB server.

- [**Verify that the following ports are available on your machine**](#)

Required ports and related IBM Sterling Transformation Extender configuration properties:

- [**Data folders used by IBM Sterling Transformation Extender**](#)

Required data folders and related IBM Sterling Transformation Extender configuration properties:

- [**Additional data folders**](#)

This version of IBM Sterling Transformation Extender Windows installation introduces additional data folders that are used to store the IBM Sterling Transformation Extender configuration files, logs, examples and temporary files, Table below lists new data folders and its default location and description.

Verify that the following ports are available on your machine

Required ports and related IBM Sterling Transformation Extender configuration properties:

Default Port	Used by IBM Sterling Transformation Extender features	Configuration file location (relative to the IBM Sterling Transformation Extender installation folder)	Configuration property name
4443	IBM Sterling Transformation Extender Server	config\config.yaml	/client/inbound/port
80	IBM Sterling Transformation Extender Server	config\config.yaml	/client/inbound/protocol (To activate https protocol client.inbound.protocol must be set to https)
8080	Runtime and IBM Sterling Transformation Extender Server	config\config.yaml	TomcatPingURL TomcatPortHttp AppDocsURL_tx-rest Note: These properties are controlled by config.yaml properties tomcat.inbound.* Note: Do not change these properties manually as these properties are updated automatically by the ITX installer and ITX start stop commands.
8443	Runtime and IBM Sterling Transformation Extender Server	config\config.yaml	authentication.server (used only If the authentication of the runtime is enabled, authentication.basic.auth.enabled property must be set to true) TX_SERVER_HOST Note: These properties are controlled by config.yaml properties tomcat.inbound.* Note: Do not change these properties manually as these properties are updated automatically by the ITX installer and ITX start stop commands.
8005	Runtime and IBM Sterling Transformation Extender Server	config\config.yaml	client.outbound settings

The IBM Sterling Transformation Extender windows installation will present you with a dialog that allows you to configure the IBM Sterling Transformation Extender design time (IBM Sterling Transformation Extender UI) ports, typically 4443 and 80. The installation verifies that the port is available on the machine. The installation will not continue if the port is not available.

Also, the IBM Sterling Transformation Extender Windows installer opens a dialog that allows you to configure IBM Sterling Transformation Extender application server ports (typically ports 8080, 8443 and 8005). The Installation verifies that the selected ports are available on the machine. The installation will not continue if those ports are not available. The Event Reporter, port 15901 from the table, must be manually verified to ensure that the port is available

Data folders used by IBM Sterling Transformation Extender

Required data folders and related IBM Sterling Transformation Extender configuration properties:

This table lists data folders used by the IBM Sterling Transformation Extender application. Verify that the free disk space is large enough to contain the IBM Sterling Transformation Extender data, 50GB of free disk space should be enough in most cases.

Default folder location	Used by IBM Sterling Transformation Extender feature	Configuration property name
C:\IBM\txfiles	IBM Sterling Transformation Extender Server	/server/persistence/files
C:\IBM\txmodules	IBM Sterling Transformation Extender Server	/server/persistence/modules
C:\IBM\txlibs	IBM Sterling Transformation Extender Server	/server/persistence/libs
C:\data	IBM Sterling Transformation Extender Server	/rest/persistence/files
C:\data\maps	Runtime	/rest/persistence/maps
C:\data\config	Runtime	/rest/persistence/config
C:\data\tmp	Runtime	/rest/persistence/work
C:\data\log	Runtime	/rest/persistence/log

Additional data folders

This version of IBM Sterling Transformation Extender Windows installation introduces additional data folders that are used to store the IBM Sterling Transformation Extender configuration files, logs, examples and temporary files, Table below lists new data folders and its default location and description.

Default location	Used by IBM Sterling Transformation Extender feature	Description
C:\ProgramData\IBM\TransformationExtender_<version>	All components	Base data folder location, this location is selected during the ITX Windows InstallShield procedure at the beginning of the installation process.
C:\ProgramData\IBM\TransformationExtender_<version>\logs	All components	Stores Tomcat service, Flow execution runtime service, and NodeJS design time log files.
C:\ProgramData\IBM\TransformationExtender_<version>\examples	All components	Stores the example files.
C:\ProgramData\IBM\TransformationExtender_<version>\config	All components	Stores the config.yaml configuration file. Encryption key file used to encrypt/decrypt config.yaml property values and resource registry values
C:\ProgramData\IBM\TransformationExtender_<version>\tmp	All components	Used to store temporary files by runtime and design time services.

Installation procedures

- [**Installing IBM Sterling Transformation Extender Server with Install Shield**](#)

To install IBM Sterling Transformation Extender, complete the following steps:

- [**Manually Installing IBM Sterling Transformation Extender Server and Runtime**](#)

The IBM Sterling Transformation Extender Install Shield typically installs all the features. If you do not select IBM Sterling Transformation Extender Server and Runtime at IBM Sterling Transformation Extender Install Shield Installation time, you can install them later.

- [**Post installation action**](#)

Modify Windows user access to IBM Sterling Transformation Extender application.

- [**Modify an existing installation**](#)

If you modify any of the settings in <installation root>\config\config.yaml, then you must restart the IBM Sterling Transformation Extender application.

Installing IBM Sterling Transformation Extender Server with Install Shield

To install IBM Sterling Transformation Extender, complete the following steps:

1. Copy the .exe file that you downloaded at the time of purchase into a directory called IBM on your computer.
2. Double click on the .exe file.
3. Install Shield opens to guide you through the installation process.

Manually Installing IBM Sterling Transformation Extender Server and Runtime

The IBM Sterling Transformation Extender Install Shield typically installs all the features. If you do not select IBM Sterling Transformation Extender Server and Runtime at IBM Sterling Transformation Extender Install Shield Installation time, you can install them later.

Use these steps to manually install IBM Sterling Transformation Extender Server and runtime:

1. Configure your IBM Sterling Transformation Extender Server and runtime installation by making any required changes to C:\ProgramData\IBM\TransformationExtender_<version>\config\config.yaml.
2. Open a Windows terminal window, and select Run as administrator.
3. Go to the installation directory of the IBM Sterling Transformation Extender base product.
4. Run the command: clean.bat
5. Run the command: install.bat

Post installation action

Modify Windows user access to IBM Sterling Transformation Extender application.

By default, all Windows users that are logged in can modify files and folders used by IBM Sterling Transformation Extender application. To limit access to IBM Sterling Transformation Extender to a specific Windows user, assign the IBM Sterling Transformation Extender services “Log On” credentials and the IBM Sterling Transformation Extender files access privileges to a specific Windows user account.

1. Open the Windows services management console.
2. Decide Windows user to run the IBM Sterling Transformation Extender services.
3. Modify the following IBM Sterling Transformation Extender Windows services “Log On” credentials by selecting this account option and providing Windows credentials for the selected user from step 2:
 - a. “Apache Tomcat 9.0 IBM Sterling Transformation Extender”
 - b. “redis server for IBM Sterling Transformation Extender”
 - c. “IBM Sterling Transformation Extender WEB UI”
4. Give write access to all IBM Sterling Transformation Extender folders to the user which was selected in step 2. For the list of folders used by IBM Sterling Transformation Extender, refer to the “Data folders by IBM Sterling Transformation Extender” table in the “Installation prerequisites” section.

Modify an existing installation

If you modify any of the settings in <installation root>\config\config.yaml, then you must restart the IBM Sterling Transformation Extender application.

In addition, if you change any variable in the config.yaml file, you must restart the application following the steps:

1. Open a Windows terminal window and select Run as administrator.
2. Run the command: <install_dir>\stop.bat
3. Run the command: <install_dir>\install.bat

Starting and Stopping IBM Sterling Transformation Extender application

After Installing, to start and stop IBM Sterling Transformation Extender , follow the steps:

1. You can open a shell with the appropriate privileges, and change the current folder to c:\IBM\IBM Sterling Transformation Extender_<version>\DesignServer
2. To start the server run, the start.bat command. The server start time might take several minutes depending on the performance of your computer.
3. To stop the server, run the stop.bat command. The server should stop in several seconds.

Accessing IBM Sterling Transformation Extender

Instructions for accessing the IBM Sterling Transformation Extender.

You can access the IBM Sterling Transformation Extender web UI by pointing the web browser to the local URL: <https://localhost:4443/login>. To login for the first time, use username “admin” and the initial administrator password which is stored in c:\IBM\txfiles\adminpassword.txt. After you log in, change the initial password of the administrator account.

After changing the administrator account password, the password file can be deleted.

Customizing the installation

IBM Sterling Transformation Extender customization instructions.

IBM Sterling Transformation Extender customization can be performed by editing settings in the file: c:\IBM\IBM Sterling Transformation Extender_<version>\config\config.yaml

Review the file content for detailed description of all customization options.

Customizing the Redis and IBM Sterling Transformation Extender Databases

IBM Sterling Transformation Extender installation includes Redis server. If you want to use a preinstalled Redis server, you must customize your IBM Sterling Transformation Extender installation.

Redis customization instructions are written in the config.yaml file. Locate the “redis:” section, ND modify Redis host and port settings to match your environment and then restart the IBM Sterling Transformation Extender application.

Customizing IBM Sterling Transformation Extender URL and Selecting HTTPS protocol

The IBM Sterling Transformation Extender default URL
<https://localhost:4443/login>
uses HTTPS protocol on port 4443.

If you want to use a non-default port and HTTPS, then you must update the client inbound settings in the configuration file <install_dir>/config/config.yaml section:

- server.inbound.port
- server.inbound.protocol
- server.inbound.host

For detailed instruction see the config.yaml file.

Uninstalling IBM Sterling Transformation Extender

Instructions for uninstalling IBM Sterling Transformation Extender.

Follow this procedure to uninstall IBM Sterling Transformation Extender:

1. Start Windows system settings console by typing "Add or remove programs" in the windows search dialog.
2. Select "IBM Sterling Transformation Extender <version number> runtime" from the list of programs.
3. Click on the "Uninstall" button.

Troubleshooting the installation

In the case of IBM Sterling Transformation Extender Install/Start/Stop errors, review the log files from this folder:

c:\IBM\IBM Sterling Transformation Extender_<version number>\DesignServer

The install.log/start.log/stop.log logs contain information about the issues and steps you will need to resolve these issues.

- [Manually Uninstalling the IBM Sterling Transformation Extender](#)

Follow these Instructions on how to uninstall the IBM Sterling Transformation Extender manually.

Manually Uninstalling the IBM Sterling Transformation Extender

Follow these Instructions on how to uninstall the IBM Sterling Transformation Extender manually.

If the IBM Sterling Transformation Extender installation gets corrupted and the uninstall procedure does not work, follow these steps to uninstall the application manually.

1. Open the Windows terminal with administrative privileges.
2. Uninstall IBM Sterling Transformation Extender services:

a. To uninstall the IBM Sterling Transformation Extender WEB UI, run the following commands:

```
sc stop ibmsterlingtransformationextenderwebui.exe  
sc delete ibmsterlingtransformationextenderwebui.exe
```

b. To uninstall the IBM Sterling Transformation Extender Application Server Service, run the following commands:

```
sc stop "IBM-Sterling-Transformation-Extender"  
sc delete "IBM-Sterling-Transformation-Extender"
```

c. To uninstall the IBM Sterling Transformation Extender Redis Service, run the following commands:

```
sc stop "redis server for ibm transformation extender"  
sc delete "redis server for ibm transformation extender"
```

3. Cleanup registry (ensure you create a backup of the registry keys before modifying the keys).

- a. Run command regedit.
- b. Navigate to the registry key Computer\HKEY_LOCAL_MACHINE\SOFTWARE\IBM\Transformation Extender
- c. Delete the registry key.
- d. Navigate to the registry key Computer\HKEY_LOCAL_MACHINE\SOFTWARE\WOW6432Node\Apache Software Foundation\Procrun 2.0\IBM.
- e. Delete the registry key.
- f. Search for the string 'IBM Sterling Transformation Extender' Under Computer\HKEY_LOCAL_MACHINE\SOFTWARE\WOW6432Node\Microsoft\Windows\CurrentVersion\Uninstall.
- g. Delete the registry Key containing the String 'Transformation Extender'.

4. Restart the computer.

5. Delete the IBM Sterling Transformation Extender installation folder, for example, C:\IBM\IBM Sterling Transformation Extender_<version>.

Troubleshooting IBM Sterling Transformation Extender installation process

If IBM Sterling Transformation Extender installation fails, verify that ports listed in the table are available on your machine.

To verify the required port availability, follow this procedure:

1. Open the windows terminal with administrative privileges.
2. Run the command for each of the listed ports

```
powershell -command  
" (Get-NetTCPConnection -LocalPort <port  
number>) .OwningProcess"
```

Note: If any of the ports are occupied, command will return the owning process id.

If any of the ports are already used by some other applications, either uninstall the port owning application, or change the default port values. You can change the related port configuration parameters:

1. During IBM Sterling Transformation Extender InstallShield installation.
2. Post installation, you will need to reinstall IBM Sterling Transformation Extender as described in the table below.

Refer Installation prerequisites section for "Ports used by IBM Sterling Transformation Extender application" table.

Manually updating Apache Tomcat 9 for the IBM Sterling Transformation Extender application

This procedure defines the steps to manually update the installed Apache Tomcat 9 software for the IBM Sterling Transformation Extender application. If the currently installed version of Apache Tomcat 9 has a security issue that is resolved in a later version, you can follow these steps to update it.

Before you begin

Important: Ensure that you back up the <IBM Sterling Transformation Extender installation folder>/restapi/ folder.

About this task

To manually update Apache Tomcat 9 for IBM Sterling Transformation Extender application on windows platform, complete the following steps:

Procedure

1. Download the latest 64-bit Windows binaries of Apache Tomcat 9 in ZIP format from the official [Apache Tomcat website](#).
2. Copy the downloaded ZIP archive into the <IBM Sterling Transformation Extender installation folder>/restapi/tomcat folder.
3. Complete the following steps:
 - a. Type **WINDOWS+R**.
 - b. Right-click the Command Prompt window and select Run as Administrator.
 - c. In the Command Prompt window, change the current directory to <IBM Sterling Transformation Extender installation folder>\DesignServer.
4. Run the following command to clean any existing Apache Tomcat installation:

```
clean.bat
```
5. Use a Text Editor to modify the dtxtomcat.ini configuration file located at the <IBM Sterling Transformation Extender installation folder>\restapi\tomcat\ location. Update the string TomcatVersion=9.0.XX to match the version of the downloaded Apache Tomcat 9 from Step 1. Save the changes.
6. From the same command prompt window folder, as mentioned in Step 3, run the following command to install Apache Tomcat:

```
install.bat
```

Linux installation

You have the choice of three installation types: Native, Docker, and Podman.

There is a consistent command line interface for the installation and management of either type. The IBM Sterling Transformation Extender script is the entry point for the CLI.

- **[IBM Sterling Transformation Extender Utility](#)**

After you extract the IBM Sterling Transformation Extender package, you can interact with the IBM Sterling Transformation Extender script to begin installation.

- **[Native installation](#)**

Procedures for installing IBM Sterling Transformation Extender using the Native installation.

- **[Docker installation](#)**

Procedures for installing IBM Sterling Transformation Extender using the Docker installation.

- **[Podman installation](#)**

Procedures for installing IBM Sterling Transformation Extender using the Podman installation.

- **[Troubleshooting](#)**

Documentation to describe how to troubleshoot an installation.

IBM Sterling Transformation Extender Utility

After you extract the IBM Sterling Transformation Extender package, you can interact with the IBM Sterling Transformation Extender script to begin installation.

See the steps provided in the Native, Docker, and Podman Installation sections of this documentation.

The IBM Sterling Transformation Extender script supports these commands:

- ITX configure: Configure the installation. Required prior to running other commands.
- ITX install: Install IBM Sterling Transformation Extender according to your configuration.
- ITX uninstall: Uninstall IBM Sterling Transformation Extender according to your configuration.
- ITX start: Start IBM Sterling Transformation Extender according to your configuration.
- ITX stop: Stop IBM Sterling Transformation Extender according to your configuration.
- ITX status: Retrieve the status of all IBM Sterling Transformation Extender components.
- ITX createuser: Create an IBM Sterling Transformation Extender user with administrative privileges.
- ITX encryptproperty: Encrypts the specified property in the configuration /home/docker_install_itx/config.yaml file.
- ITX logs: Prints the primary log of chosen component.
- ITX help: To read about a specific command.

You can view details about these commands by using the 'ITX <command> --help' option when interacting with the utility.

- **CONFIGURE command**

The CONFIGURE command is required before using any other command. This command is used to specify the installation type, the license file location, and the user/group that the installation is associated with upon completion.

- **INSTALL, UNINSTALL, START, STOP commands**

The behavior of the INSTALL, UNINSTALL, START and STOP commands is similar so we can group them for simplicity.

- **STATUS command**

The STATUS command retrieves the status of each IBM Sterling Transformation Extender component. Though the usage is similar between the native, docker, and podman installation types, the possible states are different.

- **LOGS command**

The LOGS command prints the primary log of the chosen component.

CONFIGURE command

The CONFIGURE command is required before using any other command. This command is used to specify the installation type, the license file location, and the user/group that the installation is associated with upon completion.

You can rely on the default values, but it is recommended that you specify **--type**

CONFIGURE command options

--type <native/docker/podm an>	Specify the installation type. Default: native
--integration	This option is intended for customer use only. Do not use if you purchased IBM Sterling Transformation Extender as a standalone product.
--user	The user to install IBM Sterling Transformation Extender as. This determines which user has ownership of the processes and directories. This must be an existing user. Default: current user.
--group	The group to install IBM Sterling Transformation Extender as. This provides you with the flexibility to share IBM Sterling Transformation Extender directories with a group of administrators. Default: current group

Note: You cannot change the integration type after you run install. If you notice an error after installation, you must extract the IBM Sterling Transformation Extender package into a new directory and begin again.

INSTALL, UNINSTALL, START, STOP commands

The behavior of the INSTALL, UNINSTALL, START and STOP commands is similar so we can group them for simplicity.

All commands will act upon the default components if none are specified as options. You can finetune the scope of each action by listing individual components.

INSTALL, UNINSTALL, START, STOP options

-r, --runtime	Installs the Runtime REST API used for running maps and flows.
-d, --server	Installs the Server and Client used to design, manage, and deploy maps and flows.

STATUS command

The STATUS command retrieves the status of each IBM Sterling Transformation Extender component. Though the usage is similar between the native, docker, and podman installation types, the possible states are different.

STATUS command options

-r, --runtime	Installs the Runtime REST API used for running maps and flows.
-d, --server	Installs the Server and Client used to design, manage, and deploy maps and flows.

Possible states - Native

- Not installed
- Stopped
- Running
- Unknown

Possible states - Docker

- Not installed
- Created
- Dead
- Exited
- Paused
- Running
- Restarted

Possible states - Podman

- Not installed
- Created
- Dead
- Exited
- Paused
- Running
- Restarted

LOGS command

The LOGS command prints the primary log of the chosen component.

Log files can grow to a large size. For that reason, you should pipe the output of this command into a tool like less, to facilitate better navigation.

LOG command options

-r, --runtime-app	Prints the log of the Runtime REST API Web App.
-d, --server-app	Prints the log of the Server Web App.
-c, --client	Prints the log of the Client.

Native installation

Procedures for installing IBM Sterling Transformation Extender using the Native installation.

- [**Prerequisites**](#)

Prerequisites for the Native installation.

- [**System Configuration**](#)

The documentation provided here describes how to configure the IBM Sterling Transformation Extender Native installation.

- [**Installation procedures**](#)

These instructions describe how to perform a Native installation.

- [**Manually updating Apache Tomcat 9 for the IBM Sterling Transformation Extender application**](#)

This procedure defines the steps to manually update the installed Apache Tomcat 9 software for the IBM Sterling Transformation Extender application. If the currently installed version of Apache Tomcat 9 has a security issue that is resolved in a later version, you can follow these steps to update it.

Prerequisites

Prerequisites for the Native installation.

System dependencies:

- bash (Bourne Again Shell)
- ksh (KornShell)
- libnsl
- GNU sed

Runtime dependencies:

- MongoDB Server or the OneDB wire listener.

Note: Refer to section “Using the OneDB wire listener as a MongoDB server” if a OneDB wire listener will be used instead of a MongoDB server.

System Configuration

The documentation provided here describes how to configure the IBM Sterling Transformation Extender Native installation.

To configure most properties of IBM Sterling Transformation Extender, use this file:

config.yaml

Review the **config.yaml** file before you perform the installation to confirm that the default options are as expected.

To reconfigure after the installation, update those files, and restart IBM Sterling Transformation Extender.

You will notice several directories specified in the configuration files. If you, or your group, do not have Read / Write access to these directories, contact a system administrator to create the directories and transfer ownership to you.

If you are performing the installation as a non-root user, use ports above 1024 to avoid permission issues. The default ports are:

Tomcat (REST API + Server): HTTPS 8443, HTTP 8080

UI Client: HTTPS 4443

- [Temporary directory mounting options](#)

Temporary directory mounting options are provided here.

Temporary directory mounting options

Temporary directory mounting options are provided here.

The system must not mount /tmp using the noexec option. If the system does mount /tmp using the noexec option, follow one of the two options listed here.

- [Changing the mount option for /tmp](#)

Check to see if the /tmp directory is mounted with noexec by running the following command from a command window.

```
mount | grep /tmp
```

```
tmpfs on /tmp type tmpfs (rw,nosuid,nodev,noexec,relatime)
```

Changing the mount option for /tmp

Check to see if the /tmp directory is mounted with noexec by running the following command from a command window.

```
mount | grep /tmp
```

If /tmp is mounted with noexec, follow this procedure to change the mount option for /tmp:

1. Edit **/etc/fstab** as root and remove **noexec** from the mount options of **/tmp**.
2. Reboot the system, or run the following command from a command window:

```
mount -o remount /tmp
```

Changing the IBM Sterling Transformation Extender temporary directory

To change the IBM Sterling Transformation Extender temporary directory, complete the following steps:

1. Create a new directory. Ensure that the location is not mounted with the **noexec** option. Check the output of the mount. For example, you can run the following from a command window:

```
mkdir /opt/hiptmp  
chmod 777 /opt/hiptmp
```

2. Add a new line to **tomcat-context/install/restapi/tomcat/setenv.sh** to specify the new directory. For example:

```
export JAVA_OPTS="$JAVA_OPTS -Djava.io.tmpdir=/opt/hiptmp"
```

3. Modify the existing lines in **config.yaml** for the long and short workers as shown here:

```
longTaskProcessJvmOptions="-Xmx2g -Djava.io.tmpdir=/opt/hiptmp"  
shortTaskProcessJvmOptions="-Djava.io.tmpdir=/opt/hiptmp"
```

4. Restart the IBM Sterling Transformation Extender.

Installation procedures

These instructions describe how to perform a Native installation.

Read the IBM Sterling Transformation Extender Utility documentation before following these installation instructions. You may need to adjust commands according to your use-case.

The commands shown here demonstrate a quick-start scenario:

```
./ITX configure --type native
```

```
./ITX install
```

Either start all components or continue to the troubleshooting section for any errors.

```
./ITX start
```

Manually updating Apache Tomcat 9 for the IBM Sterling Transformation Extender application

This procedure defines the steps to manually update the installed Apache Tomcat 9 software for the IBM Sterling Transformation Extender application. If the currently installed version of Apache Tomcat 9 has a security issue that is resolved in a later version, you can follow these steps to update it.

Before you begin

Important: Ensure that you back up the <IBM Sterling Transformation Extender installation folder>/tomcat-context/install/restapi/ folder.

About this task

To manually update Apache Tomcat 9 for IBM Sterling Transformation Extender application on Linux native platform, complete the following steps:

Procedure

1. Download the latest Linux binaries of Apache Tomcat 9 in **.tar.gz** format from the official [Apache Tomcat website](#).
2. Open a Linux terminal window and change the current directory to the IBM Sterling Transformation Extender installation directory <IBM Sterling Transformation Extender installation folder>.

Important: Ensure that you are logged in as the same user who initially installed the IBM Sterling Transformation Extender application.

3. Execute the following commands to stop and uninstall the current IBM Sterling Transformation Extender installation:

```
./ITX stop  
./ITX uninstall
```

4. Change the current directory to the installation directory <IBM Sterling Transformation Extender installation folder>/tomcat-context, run the following command:

```
cd <IBM Sterling Transformation Extender installation folder>/tomcat-context
```

5. Create a temporary folder and extract the contents of IBM Sterling Transformation Extender hip-core.tar.gz to it, run the following command:

```
mkdir tmp  
tar -xf hip-core.tar.gz -C tmp
```

6. Copy the downloaded **.tar.gz** binaries into <IBM Sterling Transformation Extender installation folder>/tomcat-context/tmp/restapi/tomcat folder.

7. Open the dtxtomcat.ini configuration file located at <IBM Sterling Transformation Extender installation folder>/tomcat-context/tmp/restapi/tomcat/ location using a Text Editor. Update the string TomcatVersion=9.0.XX to match the version of the downloaded Apache Tomcat 9 from Step 1. Save the changes.

8. Update the IBM Sterling Transformation Extender hip-core.tar.gz file by running the following command from the <IBM Sterling Transformation Extender installation folder>/tomcat-context/tmp folder:

```
tar -czf .. /hip-core.tar.gz .
```

9. Change the current directory to the IBM Sterling Transformation Extender installation directory <IBM Sterling Transformation Extender installation folder>, by running the following command:

```
cd <IBM Sterling Transformation Extender installation folder>
```

10. Execute the following commands to install and start IBM Sterling Transformation Extender:

```
./ITX install  
./ITX start
```

Docker installation

Procedures for installing IBM Sterling Transformation Extender using the Docker installation.

Note: Before you continue with any of the procedures in this section, make sure certain Docker is installed.

Note: IBM Sterling Transformation Extender does not offer Docker installation compatibility on Red Hat platforms version 8 and beyond.

- [Docker configuration](#)
Description of how to configure your Docker installation.
- [Installation](#)
Procedures in this documentation describe how to perform a Docker installation.

Docker configuration

Description of how to configure your Docker installation.

To configure most properties of IBM Sterling Transformation Extender, use this file:

`config.yaml`

Review the **config.yaml** file before you perform the installation to confirm that the default options are what you expect.

To reconfigure after installation, update those files, and restart IBM Sterling Transformation Extender.

You will notice several directories specified in the configuration files. If you, or your group, do not have Read / Write access to these directories, you will need a system administrator to create the directories and transfer ownership to you.

If you are installing as a non-root user, use ports above 1024 to avoid permission issues. The default ports are:

REST API: HTTPS 9443, HTTP 8080

Server: HTTPS 8443

UI Client: HTTPS 4443

Installation

Procedures in this documentation describe how to perform a Docker installation.

Read the IBM Sterling Transformation Extender Utility documentation before following the installation steps. You may need to adjust commands according to your use-case.

The commands shown here demonstrate a quick-start scenario.

`./ITX configure --type docker`

`./ITX install`

Either start all components or continue to the troubleshooting section for any errors.

`./ITX start`

Podman installation

Procedures for installing IBM Sterling Transformation Extender using the Podman installation.

On the RedHat Linux distribution, IBM ITX application installation supports Podman installation types. On other Linux distributions (For example: Ubuntu), IBM ITX supports both Docker and Podman container engines. Select Podman method for ITX installation.

Prerequisites

Before installing ITX in Podman mode, ensure that the ITX installation folder and data folders are not located on a network file system. The ITX application and its data must be on the local computer's file system. Additionally, do not inherit data folders from previous ITX installations of different types.

The **config.yaml** file specifies the ITX data folders. Make sure these folders and their subfolders are located on the local computer's file system:

- `/opt/data`
- `/opt/itx`
- `/opt/txfiles`
- `/opt/txmodules`
- `/opt/txlibs`
- `/opt/tx-rest`
- `/tmp`

Podman Configuration

To configure the ITX installation to Podman mode, run the following command:

`./ITX configure -type podman`

Installation Procedures

After configuring the ITX installation to install in Podman mode, run the following command:

`./ITX install`

Troubleshooting

Documentation to describe how to troubleshoot an installation.

Most installation issues include an error message printed either to your terminal, or written to `<command>.log`

For any errors relating to start up of a component, check the log for the component. Use the `logs` command to view the log.

Common errors

Error:

IBM Sterling Transformation Extender client logs show *ENOSPC: System limit for number of file watchers reached*

Solution:

Increase the inotify max_user_watches property.

```
echo fs.inotify.max_user_watches=524288 | sudo tee -a /etc/sysctl.conf
```

```
sudo sysctl -p
```

Using the OneDB wire listener as a MongoDB server

The OneDB wire listener may be used in place of a MongoDB server when it is suitably configured.

OneDB wire listener configuration recommendations:

- Choose a MongoDB API version for the value of the mongo.api.version configuration property that is compatible with the installed version of IBM Sterling Transformation Extender.
 - Error messages may occur in the IBM Sterling Transformation Extender User Interface, if the wire listener uses a short idle connection timeout value for the value of the listener.idle.timeout configuration property. A suitable large value, for example a timeout of 12 hours (listener.idle.timeout=43200000), should be used. If appropriate, idle connection closure may be disabled (listener.idle.timeout=0).
-

IBM Sterling Transformation Extender design data

The documentation provided here describes data backup, data restoration, and data migration of IBM Sterling Transformation Extender design data.

- [Design data storage](#)
This documentation describes data stores used by IBM Sterling Transformation Extender for design activities, and data storage configuration.
 - [Design data backup and migration](#)
Overview of backing up and restoring, or migrating data used by IBM Sterling Transformation Extender for design activities.
-

Design data storage

This documentation describes data stores used by IBM Sterling Transformation Extender for design activities, and data storage configuration.

For design activities, an IBM Sterling Transformation Extender installation uses these data stores:

- Databases which are managed by a MongoDB server
- Directories in a file system.

When you install IBM Sterling Transformation Extender, configuration information for each of these data stores is provided. This information resides in a configuration file. Depending upon the operating system, the path to the configuration file is as follows:

- Windows: <Data directory>\config\config.yaml, for more details see [Additional data folders](#).
 - Linux (native and docker installations): <installation directory>/config.yaml
-

Design data backup and migration

Overview of backing up and restoring, or migrating data used by IBM Sterling Transformation Extender for design activities.

Note: Due to the variability in MongoDB installations and file system configuration, the exact steps for backing up and restoring design data will vary between individual IBM Sterling Transformation Extender installations.

- [MongoDB database operations](#)
 - [Filesystem data backup and restore](#)
-

MongoDB database operations

This documentation includes a description of how to back up the design databases and how to restore the databases.

The documentation also includes an example of commands that can be executed in order to migrate from a local MongoDB server instance to a remote MongoDB server instance.

- [Design databases](#)
IBM Sterling Transformation Extender uses a master database and a resource database to store some of its design data.

- [**Additional accounts**](#)

Additional accounts may be associated with a MongoDB server instance. Each account is associated with its own account ID and resource database. The account ID is stored in the accounts collection in the manner described above. Additional accounts are not present in default IBM Sterling Transformation Extender installations.

- [**Backing up and restoring databases**](#)

Backing up the design databases consists of backing up the master database and each resource database. Similarly, restoring the design databases, whether to a previously used, or to a new MongoDB server instance, consists of restoring the master database and each backed up resource database.

- [**MongoDB migration example**](#)

This documentation describes example commands for MongoDB server instance migration with a native Linux IBM Sterling Transformation Extender installation.

Design databases

IBM Sterling Transformation Extender uses a master database and a resource database to store some of its design data.

The name of the master database was provided during IBM Sterling Transformation Extender installation by the value of /mongo/masterDb in config.yaml. During a Windows installation, the value of this property is usually provided through the installation wizard. The default value for this configuration property in a stand-alone IBM Sterling Transformation Extender installation is mdb. Other names may be used in different contexts.

An IBM Sterling Transformation Extender installation is associated with a default account. The default account is named Default. This name is present as the value of the name property of an account object in the accounts collection of the master database. The default account is associated with an account ID. This account ID is the value of the _id property of the account object in the collection.

A single resource database is created during default installation. The resource database is named using the pattern rdb_<account ID>. The master database and the resource database which is associated with the default account store IBM Sterling Transformation Extender project information and related resources.

An example MongoDB shell session is given below, which shows how the default account ID can be determined. The name of the master database is assumed to be "mdb"

- use mdb
- db.accounts.find({“name”: “Default”})
 - A query result such as the following should be returned
 - [{ _id: ‘61532e49396b627364f9953e’, name: ‘Default’ }]
 - 61532e49396b627364f9953e is the ID of the default account in this example.
 - A resource database named rdb_61532e49396b627364f9953e would be present in this example

Additional accounts

Additional accounts may be associated with a MongoDB server instance. Each account is associated with its own account ID and resource database. The account ID is stored in the accounts collection in the manner described above. Additional accounts are not present in default IBM Sterling Transformation Extender installations.

Backing up and restoring databases

Backing up the design databases consists of backing up the master database and each resource database. Similarly, restoring the design databases, whether to a previously used, or to a new MongoDB server instance, consists of restoring the master database and each backed up resource database.

For Linux Docker installations: For IBM Sterling Transformation Extender installations on a Linux system using Docker, the MongoDB container configuration must be known to perform MongoDB database backups. For default Docker installations, the container name is hip-server-mongo. For example, a backup could be performed by starting an interactive terminal session in the running container and dumping the master and resource databases to a directory within the container. The dumped data could then be copied out of the container.

If MongoDB databases are being migrated to a new MongoDB server instance, you must remember to update the MongoDB connection properties in the appropriate IBM Sterling Transformation Extender configuration file.

In general, it must be considered if IBM Sterling Transformation Extender downtime is necessary when migrating between MongoDB server instances. In the simplest case, IBM Sterling Transformation Extender would be stopped, the design databases would be backed up, and the restore operation in the new MongoDB server instance would be performed. The IBM Sterling Transformation Extender installation configuration update operation would then be updated, and IBM Sterling Transformation Extender could be started with the new configuration.

MongoDB migration example

This documentation describes example commands for MongoDB server instance migration with a native Linux IBM Sterling Transformation Extender installation.

This example shows commands which could be executed to migrate from a local MongoDB server instance to a remote MongoDB server instance. This example assumes that default configuration properties were used when IBM Sterling Transformation Extender was installed natively on the Linux host.

Note: The utilities mongodump and mongorestore might not be appropriate for use with a particular MongoDB server instance. Also, use of these utilities is subject to constraints. These constraints are described in MongoDB database tools documentation.

This example assumes that the target MongoDB server instance has been configured and is accessible using the standard MongoDB port from host <MongoDB host>. It is also assumed that the MongoDB database tools have been installed, so that mongodump and mongorestore are available.

Refer to the following Bash session:

1. Stops IBM Sterling Transformation Extender using the IBM Sterling Transformation Extender utility shell script.

- a. cd <IBM Sterling Transformation Extender installation directory>
 - b. ./ITX stop
 2. Creates a backup of the master database and the resource database. A directory named dump is created in the IBM Sterling Transformation Extender installation directory by mongodump. This directory holds the backups. Standard output and standard errors are appended to file itx_mongodump.log.
 - a. mongodump --db=mdb &>> itx_mongodump.log
 - b. mongodump --db=rdb_<account ID> &>> itx_mongodump.log
 3. Restores the dumped databases into the target MongoDB server instance. Standard output and standard error are appended to file itx_mongorestore.log.
 - a. mongorestore --host=<MongoDB host> &>> itx_mongorestore.log
 4. The configuration file config.yaml in <IBM Sterling Transformation Extender installation directory> must be edited to reflect the change in the location of the MongoDB server. Assuming that the default MongoDB port and a remote host are used, the value of the "host" key of the object value of the top-level "mongo" key (mongo.host) must be set to the remote host.
 5. Starts IBM Sterling Transformation Extender after updating the MongoDB server configuration. The updated values are read.
 - a. ./ITX start
 6. After confirming that IBM Sterling Transformation Extender operates with the new MongoDB server instance as expected, the dump directory should be removed:
 - a. rm -r dump
-

Filesystem data backup and restore

IBM Sterling Transformation Extender uses several file system directories to store design data. They are shown below using JSON Pointer strings to refer to a mapping value in the IBM Sterling Transformation Extender config.yaml file:

- server.persistence.files
- server.persistence.modules
- server.persistence.libs
- rest.persistence.files

Backing up filesystem design data consists of creating a backup of these directories. The backup process varies depending upon the configuration of the filesystems used to store these directories and their contents.

Restoring filesystem design data consists of populating these directories with their backed-up contents. A change in the location of one or more of these directories during a restore operation might be desired. As for a change in the IBM Sterling Transformation Extender configuration for accessing a MongoDB server instance, such a change in location is allowed, and is handled by the general IBM Sterling Transformation Extender installation configuration update operation.

- [Filesystem data migration example](#)
-

Filesystem data migration example

As an example, suppose an IBM Sterling Transformation Extender installation initially used directories on a local filesystem for the values of /server/persistence/files, /server/persistence/modules, /server/persistence/libs and /rest/persistence/files in config.yaml. If a change in the locations of these directories is desired, then the following steps could be used for file system design data migration:

1. Start an administrator command prompt session. Execute the following commands.
 2. Stops the IBM Sterling Transformation Extender installation:
 - a. cd <IBM Sterling Transformation Extender installation directory>\DesignServer
 - b. .\stop.bat
 3. Back up the directories.
 4. Create new directories in their new locations and populate them with the appropriate backed-up data.
 5. Update, as necessary, the values of server/persistence/files, /server/persistence/modules, /server/persistence/libs and /rest/persistence/files in <Link installation directory>\config\config.yaml.
 6. Starts IBM Sterling Transformation Extender after the configuration update.
 - a. .\start.bat
-

Administration

This documentation describes those administration tasks required for Design Server.

User Management

Design Server provides inbuilt support for managing users who may create, deploy and run integrations.

Users are assigned one of these roles:

- Administrator – Full access to all features of Design Server.
- Designers – They can create and modify projects and any artifacts within a project. A designer cannot access user management, cannot deploy projects, cannot import projects and cannot delete projects.

To administer users and user groups, you must have user Administrator privileges. When you first install Design Server an initial administrator user is created named admin with password admin. After installation, the password of this user should be changed.

- [Managing user accounts](#)

You can access the user management interface by performing the following steps:

- [Add a user](#)

These procedures describe how to use Design Server to add a new user.

Managing user accounts

You can access the user management interface by performing the following steps:

1. Login to the Design Server as an administrator.
2. Click the Username at the top of the screen to access the menu.
3. From the drop-down menu select Administration.

From the **Administration** window, an administrator can perform these tasks:

- Create new users
- Edit users details
- Change user's passwords
- Release locks for artifacts locked by a user

Add a user

These procedures describe how to use Design Server to add a new user.

To add a new user, login to Design Server as an Administrator, and then follow these steps on the Welcome page:

1. Click the drop-down arrow next to the Administrator tab in the upper right corner of the Welcome page.
2. Select the Administration option from the menu. The Users window opens.
3. Click the plus sign icon located to the right of the Search box at the top of the page to open Add User.
4. Complete all of the fields in Add User area with the new user's information. Be aware of the following when completing the fields:
 - The password that you assign must be at least eight characters in length and contain at least one number and one character that is neither a number or a letter.
 - In most cases the Role field will remain in its default condition and will display the Designer option.
5. Click the Add button at the bottom of the page. You are returned to the Users page. The user that you added appears in the list of users.

Introduction

Keycloak is an open-source identity and Access Management solution. It is a single sign-on security application for web applications and RESTful web services. Design Server and Runtime REST API supports Keycloak solution of user administration for authentication and authorization.

Existing user databases hold user credentials. Keycloak federates these existing external user databases through the concept of storage providers. By default, Keycloak supports an LDAP and Active Directory storage provider.

When an user tries to access the Design Server Web-console, the browser is redirected to Keycloak authentication system to authenticate user credentials. User gets a reference token from the Keycloak to connect to the server.

Installation

See [Getting Started](#) guide for installation of Keycloak.

User Administration

After you install Keycloak, you must consider how the software manages users and authentication.

You can review the default user management provided by the Keycloak server and decide what additional controls you might want to add. If you manage users and authentication through an existing LDAP/AD server, you can review how to use that server to manage users.

- [Default User Administration](#)

Keycloak uses the concept of a realm to manage and authenticate users. When you install the Keycloak server, a realm called testserver is created for you in Keycloak. All server users belong to this realm and when they log in to the server, they log into that realm.

- [LDAP User Authentication](#)

You are using a Lightweight Directory Access Protocol and Active Directory HTTP server to manage users and authentication. You want to know the best practices to use that LDAP/AD server to manage user access to Design Server.

Default User Administration

Keycloak uses the concept of a realm to manage and authenticate users. When you install the Keycloak server, a realm called testserver is created for you in Keycloak. All server users belong to this realm and when they log in to the server, they log into that realm.

As an administrator, it is important to consider the following points about the Keycloak server administration:

- By default, there is no admin user for Design server. Such an admin user is required for accessing additional Design server functions, which includes claiming ownership of Design server projects and unarchiving them. But you can assign administrative privileges to any user. You must do this by adding the admin role to the user in Keycloak. See [Getting Started](#) guide for more information
- If Keycloak authentication is enabled in Design Server, you need to create the admin user or you can synchronize users from ldap and assign administrative privileges to any user
- Keycloak does not come with a default admin user. You need to create an admin user before using the Design server application. To do this open <http://localhost:8080/auth>, fill the form with your preferred username and password
After you log in to the Keycloak Admin Console, from the Users page, you can search and select the user that you want to make an administrator. From the Groups tab, you can join the user to the Admin group.

For more information about assigning user roles, see [Groups](#) in the Keycloak documentation.

Now that you are the Keycloak server administrator, it is important to consider the following points about the default user management and authentication:

- Minimum password length defaults to 8 characters
- Email verification of new users is turned off
- The Forgot Password feature is turned on by default, but no instructions are sent to the user to reset their password
- Forgotten user passwords are changed by you, if you do not enable Keycloak to send instructions to reset a password

You can review the following sections about changing the default authentication controls.

Email settings

By default, the testserver realm sets the Forgot Password switch on. However, as an administrator, you must enable Keycloak to send an email to the user with instructions to reset their password. If you want to verify an email, you must also enable Keycloak to send an email to the user to verify their email address.

You must provide SMTP server settings for Keycloak to send an email. After you log in to the Keycloak Admin Console, see [Email Settings](#) in the Keycloak documentation.

To set up the email verification, see [Forgot Password](#) in the Keycloak documentation.

User password

Organization can give user access to the account console located at:

<https://<keycloak-url>/auth/realms/<realm>/account>

There is a form to update password (and other useful information about the account). See [User Credentials](#) in the Keycloak documentation.

User deletion

When a user is inactive or no longer access the Design server application, you can delete that user.

See [Deleting Users](#) in the Keycloak documentation.

- [Creating a Realm](#)
- [Configuring Clients](#)
- [Connecting to Design Server](#)

This topic documents how to set up connection between Keycloak and Design server.

Creating a Realm

1. Log in to the Keycloak Admin Console.
2. Click on the Select realm drop-down and click on Add realm button. The Add realm page opens.
3. Import the realm-export.json file from the installation directory. When you import the realm-export.json, it will automatically configure all the settings of realm. You can edit the configuration settings as required.
4. Specify the name of realm in the Name field.
5. Click on Create button to create a realm.
The realm is created with its realm.

Configuring Clients

1. Log in to the Keycloak Admin Console.
2. Click on Clients option from the Configure property. The Clients page opens. User can see the list of application available in the Keycloak.
3. Click on the design-server-rest client from the list. The Design-server-rest page opens. When you select the design-server-rest, it will automatically configure all the properties available in the Settings tab.
4. Go to Credentials tab, click on Regenerate Secret button to create client secret for user.
5. Go to Roles tab to edit the attributes of respective role, if required.

Connecting to Design Server

This topic documents how to set up connection between Keycloak and Design server.

1. Install the Design Server to connect to Keycloak.
2. In Keycloak, make sure that Keycloak Auth Enabled is set to true.
3. In text editor, open config.yaml to confirm the connectivity information. You can see the below information and update as appropriate to your environment:

```
# Flag indicating if keycloak is used for identity & access management
TX_KEYCLOAK_AUTH_ENABLED=true
# Keycloak server url
TX_KEYCLOAK_SERVER_URL=http://localhost:XXXX/auth
# Keycloak realm id
TX_KEYCLOAK_REALM_ID=itx
# Keycloak client id for making rest calls
TX_KEYCLOAK_CLIENT_ID=design-server-rest
# Associated client secret for above client id
TX_KEYCLOAK_CLIENT_SECRET=94a30704-7694-476c-b9c2-5fb78d151892
```

4. Log in to Design Server using Keycloak or LDAP user credentials as configured. To unlock the maps if they are locked, go to Administration in Design Server and delete locks held by a user.

LDAP User Authentication

You are using a Lightweight Directory Access Protocol and Active Directory HTTP server to manage users and authentication. You want to know the best practices to use that LDAP/AD server to manage user access to Design Server.

Design Server supports Keycloak (<https://www.keycloak.org/>) to manage and authenticate users.

Existing user databases hold user credentials. Keycloak federates these existing external user databases through the concept of storage providers. By default, Keycloak supports an LDAP and Active Directory storage provider. By adding a storage provider, you can map LDAP user attributes into Keycloak. You can also configure more mappings.

Before you configure Keycloak to use an existing LDAP/AD provider, you must consider the following best practices:

- Set up your LDAP/AD provider as a read-only repository so that Keycloak Server cannot change it
- Add and remove users in LDAP/AD and not the Keycloak local user database
- Import and synchronize your LDAP/AD users to your Keycloak local database
 - An import for an LDAP/AD user can fail, if the LDAP/AD field chosen for the username mapping in Keycloak is not filled in for that user in LDAP/AD
 - Filter LDAP/AD users by using the Custom User LDAP Filter, so you can import a subset of all your LDAP users. For example, you can set up a Server user group in LDAP and only import those users to Keycloak
- Map a login style name, for example, user1@server.com, by using the UserPrincipalName attribute in LDAP/AD to a username in Keycloak. If you want the full name of the user as your login style, use the cn attribute in LDAP/AD.

Note: The LDAP/AD user name attribute must match the LDAP/AD provider user name attribute (**Username LDAP attribute**) in Keycloak for the LDAP/AD provider to connect with Keycloak.

The following sections use these best practices to guide you to set up Keycloak to connect to your LDAP/AD HTTP server.

- **Selecting LDAP provider in Keycloak**
You can use the Keycloak Admin Console to add an LDAP/AD provider.
- **Required settings for a successful connection to your LDAP/AD provider**
The form includes many fields and several fields include default values.
- **User synchronization**
You must import all users from your LDAP/AD user database by using the option to Synchronize all users. Users are imported based on your saved settings when you set up your LDAP/AD provider.
- **Mappers**
Keycloak uses mappers to map the user attributes defined in the Keycloak user model such as username and email to the corresponding user attributes in the LDAP/AD user database. By default, when you saved your settings and created your LDAP/AD provider, the following mappers were created.

Selecting LDAP provider in Keycloak

You can use the Keycloak Admin Console to add an LDAP/AD provider.

1. Log in to the Keycloak Admin Console.
2. Go to the User Federation page to add your provider.
3. Select the provider named ldap from the list from the Add Provider drop-down list. The ldap provider is created.

Required settings for a successful connection to your LDAP/AD provider

The form includes many fields and several fields include default values.

1. Click on the ldap provider to open the settings to configure.
2. In the Settings tab, select Active Directory from the drop-down list of Vendor. Many fields complete with default values based on this selection.
3. Enter your LDAP/AD URL to connect to your LDAP/AD user database, for example: ldap://<hostname>.<domain>
4. Click on Test connection button to test the connection and confirm that the connection is successful.
5. Provide the directory where the LDAP users are listed, for example: cn=Users,dc=MYCOMPANY,dc=COM.
6. From the Bind Type drop-down list, select Simple option.

7. Provide the LDAP/AD user database administrator user ID for BIND DN and password for the BIND Credential. These credentials are used by Keycloak to access the LDAP/AD user database.
 8. Click on Test Authentication button to test the authentication and confirm that the authentication is successful.
 9. Click on Save button to save the configuration.
-

User synchronization

You must import all users from your LDAP/AD user database by using the option to Synchronize all users. Users are imported based on your saved settings when you set up your LDAP/AD provider.

A successful import is followed by a success message with the number of users imported. A failed import typically results when there is a mismatch between user attributes in the Keycloak user database and the LDAP/AD user database.

You can view all the LDAP/AD database users that were imported and authenticated from the Users page in the Keycloak Admin Console.

Users are listed with ID, Username, Email, Last Name, and First Name. The ID is generated by Keycloak. The value of the other attributes is fetched from the LDAP/AD user database by using mappers.

Mappers

Keycloak uses mappers to map the user attributes defined in the Keycloak user model such as username and email to the corresponding user attributes in the LDAP/AD user database. By default, when you saved your settings and created your LDAP/AD provider, the following mappers were created.

The username attribute that you specified in the Username LDAP attribute must match the username attribute defined in the Keycloak mapper for the LDAP/AD user database to connect with Keycloak.

If you change the Username LDAP attribute from the default value **cn** to **userPrincipalName**, Keycloak would make the same change in the mapper called username to match.

Runtime REST API

Runtime REST API supports Keycloak authentication of user credentials for running maps and flows. Like Design Server, Runtime REST API can be configured to authenticate users with Keycloak.

- [Connecting Runtime REST API server](#)
This topic documents how to set up connection between Keycloak and Runtime REST API server.
-

Connecting Runtime REST API server

This topic documents how to set up connection between Keycloak and Runtime REST API server.

1. Install Design Studio or Runtime & Monitoring.
 2. In Keycloak, make sure that Keycloak Auth Enabled is set to true.
 3. In text editor, open config.yaml to confirm the connectivity information. You can see the below information and update as appropriate to your environment:

```
keycloak.enabled=true
#Keycloak specific properties
keycloak.serverUrl=http://localhost:8080/auth
keycloak.realmId=itz
server.inbound.authentication.keycloak.clientId=itz-runtime-rest
server.inbound.keycloak.clientSecret=9465403f-0ae6-4cf8-ecdf8c4485ed
```
 4. Run Runtime REST API server to service REST client calls.
 5. REST clients invoking the Runtime REST API should include the user credentials that can be authenticated with Keycloak to run the maps or flows, or to fetch any runtime information.
-

Outbound connections to Runtime Servers

Transport Layer Security (TLS) client settings can be configured for outbound connections from Design Server to Runtime Servers.

These settings are intended to allow secure connections to runtime installations. Connecting to a customized runtime installation which uses mutual authentication (mTLS) is supported.

Use of these settings is not required if the previous behavior for outbound connections to runtime installations is sufficient. The default settings provide backwards-compatible behavior.

Connections to a Runtime Server are only allowed if the server is on the Design Server allowlist. See `server.runtime.servers` in config.yaml.

The client TLS settings are configured with the mapping `server.outbound.ssl` in config.yaml. These settings apply to all connections from Design Server to runtime installations.

If TLS is used for connections to Runtime Servers, a trust store must be provided as a PEM file which contains one or more CA certificates. The absolute path of the file should be provided as the value of server.outbound.ssl.caFile.

If a Runtime Server uses mTLS, a client certificate and client private key must be provided as PEM files. Absolute paths to these files should be provided as the values of server.outbound.ssl.certFile and server.outbound.ssl.keyFile, respectively.

The setting server.outbound.ssl.acceptCertificatesStrategy controls certificate verification. If the value **signed** is provided, then only those certificates for which a valid trust chain can be constructed are accepted. This is the recommended setting for production servers when TLS is used. If the value **all** is provided, then all server certificates are accepted. This setting should not be used for production servers. When a value is not provided, self-signed certificates are accepted in addition to valid CA-signed certificates.

If the boolean-valued setting server.outbound.ssl.verifyHostName is set to **true**, hostname verification is performed for the server certificate in addition to the verification indicated by server.outbound.ssl.acceptCertificatesStrategy.

Below is an example of the default values in config.yaml.

```
server:
  outbound:
    ssl:
      certFile: ""
      keyFile: ""
      caFile: ""
      acceptCertificatesStrategy: ""
      verifyHostName: false
```

Design Server concepts

This documentation describes basic Design Server concepts

Projects

This documentation describes Design Server projects.

- **Importing and exporting**

A project organizes related artifacts (connections and actions, schemas, maps, and files) in the Design Server repository. The import and export functions move projects or individual artifacts between the repository and the local file system. By using the import and export functions, you can reuse artifacts in multiple projects or move a project between multiple instances of Design Server.

- **Connections and actions**

A connection defines the properties that are required to connect to an external resource. An action uses a connection to provide source data to a map, or to write data from a map to a target.

- **Files**

You add a file to a project to use it as input to maps in the project. You can upload a file from your local file system to a project with the File designer. Alternatively, you can upload it when you define an action.

Importing and exporting

A project organizes related artifacts (connections and actions, schemas, maps, and files) in the Design Server repository. The import and export functions move projects or individual artifacts between the repository and the local file system. By using the import and export functions, you can reuse artifacts in multiple projects or move a project between multiple instances of Design Server.

For example, you can export a project from the Design Server instance running on a shared server at your office to import it to the Design Server instance running on your local laptop.

Exporting a project creates a .zip archive file and manifest of the project and downloads it to the local file system. Importing a project moves the project from the local file system into the Design Server repository. You can import both Design Server projects and legacy projects into Design Server. Compress legacy artifacts into a .zip archive file before you import them.

You also can select individual artifacts (connections and actions, schemas, maps, and files) to export and import. For example, you might want to export a map to send it to Support to diagnose a problem, or you might want to copy a schema from one project to another.

When you export individual artifacts, you can choose to include dependent artifacts automatically.

- Maps depend on schemas, files, connections, and actions.
- Actions depend on schemas, files, and connections.

Connections and actions

A connection defines the properties that are required to connect to an external resource. An action uses a connection to provide source data to a map, or to write data from a map to a target.

Multiple actions can use the same connection. You can define both connections and actions at either the project level, or at the map level.

- [**Project-level connections and actions**](#)

You can reuse project-level artifacts in multiple maps. You can use the Connections designer in Design Server to define a new project-level connection and its actions, or to select an existing project-level connection and its actions.

- [**Map-level connections and actions**](#)

Map-level artifacts are unique to a particular map. A map-level action can use an existing, project-level connection, or the action can define a new map-level connection.

- [**Connection testing**](#)

You can test both project-level and map-level connections.

Project-level connections and actions

You can reuse project-level artifacts in multiple maps. You can use the Connections designer in Design Server to define a new project-level connection and its actions, or to select an existing project-level connection and its actions.

For example, you can define a project-level connection to a database that is used as a data source, or target of multiple maps. A project-level action might be an SQL inquiry and schema that multiple maps use to look up the same database.

Map-level connections and actions

Map-level artifacts are unique to a particular map. A map-level action can use an existing, project-level connection, or the action can define a new map-level connection.

To define a map-level action.

Alternatively, you can specify the properties of an action directly in the input or output card. It is not necessary to create a map-level action.

Connection testing

You can test both project-level and map-level connections.

When the Design Server has access to an external resource, testing validates the properties of the resource connection. If the Design Server can not access the external resource, the connection test fails, even if the connection properties are correct.

Files

You add a file to a project to use it as input to maps in the project. You can upload a file from your local file system to a project with the File designer. Alternatively, you can upload it when you define an action.

You also can download a file from a project to your local file system. From your local file system, you can upload the file to a different project or to a different instance of Design Server.

Schemas

Schema design

Use the product and documentation to define, modify, and view schemas. A schema describes the syntax, structure, and semantics of your data. The syntax of data refers to its format including tags, delimiters, terminators, and other characters that separate or identify sections of data. The structure of data refers to its composition including repeating substructures and nested groupings. The semantics of data refer to the meaning of the data including rules for data values, relationships among parts of a large data object, and error detection and recovery.

- [**About schemas**](#)
- [**Subtypes**](#)
- [**Schema hierarchy**](#)
- [**Defining data**](#)
- [**Objects, types, and classes**](#)
- [**Testing part of a schema**](#)

You can test a schema by generating test data for an item or a group. The Type Designer creates source and compiled maps that generate and validate the test data. The test-data-generation map automatically runs and produces a text file with the sample data. After that, the test-data-validation map automatically runs on the sample data file.

About schemas

A schema defines the contents of at least one input that you intend to map or one output you intend to map. A schema is the mechanism for defining each element of your data. Similar to a data dictionary, a schema contains a collection of type definitions.

Because data definitions are defined in a schema, you should be familiar with the specifications that define your data before attempting to create one.

A data file is a simple example. The file is made up of records and each record is made up of fields. In this case, there are three kinds of data objects: a file, a record, and a field.

In a file of records, think of the data in terms of the three data objects. For example, one type defines the entire file; another type defines the entire record contained in that file. Other types in the same schema define the data fields of the record.

A schema has three datatype classifications: group, category, and item. The **root** type is the base type from which all other types stem, representing the data objects of all types in the schema. "Root" is the default name, however, you can modify any type name.

Types within a schema are listed in alphabetical order by default. To change the order for new types, modify the root type properties.

IBM Transformation Extender supports large metadata. It supports more than 65,536 types and 65,536 components in a single schema. To reduce the number of types that maps that use large schemas must process, trim these schemas so that they contain only the required types.

Subtypes

Subtypes are created for a number of reasons like identifying distinct data object. Another reason is that specific types of an object may have different properties such as different date formats.

Think of subtypes as different "flavors". Ice cream flavors can be chocolate, strawberry, or vanilla. To create a schema to represent ice cream as data, the different flavors of ice cream could be subtypes of the type **IceCream**.

The type **IceCream** is generic and describes any kind of ice cream. The type **Chocolate**, a subtype of **IceCream**, represents a certain kind of ice cream: chocolate.

A file of purchase order data may have two different kinds of records: header and detail. The schema representing this data might have a **Record** type with **Detail** and **Header** as subtypes of **Record**.

Schema hierarchy

The types in a schema are arranged in a hierarchy. If a type is subordinate to another type, it is called a *subtype*. The type on the branch stemming above a specific type is called its *supertype*.

Subtypes have more specific properties. For example, two different kinds of fields, **Name** and **Date**, may be defined as subtypes of **Field**.

The item type **Field** describes any field. The item types **Date** and **Name** are more specific kinds of fields. In a classification hierarchy such as this, the deeper the subtype is in the type tree, the more specific the data characteristics.

Example scenario

Imagine that a schema represents your house. There is a type for the entire house (**House**). A type for each room (**Bathroom**, **Bedroom**, and so on), and types for the different furnishings in these rooms (**Bed**, **Chair**, and so on). Each type represents a complete object. A house is made up of rooms. Inside these rooms are beds, chairs, couches, and so on. You know this, but you cannot see it by looking at the classification hierarchy view of the schema. You must open one of the types in the group view to see its components.

Defining data

For optimum data transformation, the entire contents of your data must be defined. Using the Type Designer, you can define input data so that each data object of the source data is identified. You can define the output data according to your output specifications.

How specifically you define the data is up to you. For example, if there is a large section of data that you want to process quickly, define it loosely as a chunk of text.

The Type Designer does not define data content as source data or target data. The purpose of the Type Designer is to define your data. Use the Map Designer to define input and output definitions for the data.

Objects, types, and classes

Schemas consist of objects, types, and classes.

Objects

A data object is a complete unit that exists in your input or is built on output. A data object can be simple (such as a date) or complex (such as a purchase order).

A data object is some portion of data in a data stream that can be recognized as belonging to a specific type. When you create types, identify all of the objects that make up the data: input objects and output objects.

Types

A type defines a set of data objects that have the same characteristics. For example, the type "Date" can be defined as representing data objects in the form MM-DD-YY. The type "CustomerRecord" can be defined as representing data objects, each of which consists of a **Company**, **Address**, and **Phone** data object.

Classes

A type is classified according to whether or not it consists of other objects. Each type in a schema must be defined in one of three classes: item, group, or category.

- **Item type**

An item type represents a simple data object that does not consist of other objects.

- **Group type**

A group type represents a complex data object that consists of other objects. For example, "FullName" is a group type that contains the components: FirstName, MiddleInitial, and LastName.

- **Category type**

A category type is used for inheritance and for organizing other types in a schema. For example, you might have a category named "OrderField" to organize the different kinds of order fields in your schema.

Testing part of a schema

You can test a schema by generating test data for an item or a group. The Type Designer creates source and compiled maps that generate and validate the test data. The test-data-generation map automatically runs and produces a text file with the sample data. After that, the test-data-validation map automatically runs on the sample data file.

For choices and partitions, the Type Designer fills the first choice and first partition with data in the generation map. For sequences, the Type Designer generates one index. The Type Designer uses implied default-value and item-value restrictions if they are available; otherwise, it generates sample values for different data types.

- **[Invalid input message from test data generation](#)**

The message One or more inputs was invalid displays when test data generates successfully but the validation map flags some data as not valid. This message can occur when there are data type constraints, component rules, or restrictions that the sample data does not account for. Trace the validation map and inspect the generation map rules to find the data types that are not generating correctly.

Invalid input message from test data generation

The message One or more inputs was invalid displays when test data generates successfully but the validation map flags some data as not valid. This message can occur when there are data type constraints, component rules, or restrictions that the sample data does not account for. Trace the validation map and inspect the generation map rules to find the data types that are not generating correctly.

Importers

There are three types of importers:

- COBOL copybook
- CSV
- XSD or JSON

- **[COBOL Copybook Importer](#)**

The COBOL Copybook Importer imports the definition of one or more COBOL copybooks in a file and generates a type tree that contains the corresponding type definitions. The copybook file might contain more than one COBOL copybook definition.

- **[CSV Importer](#)**

The CSV Importer imports the content from the CSV file to the Design Server UI.

- **[JSON Importer](#)**

COBOL Copybook Importer

The COBOL Copybook Importer imports the definition of one or more COBOL copybooks in a file and generates a type tree that contains the corresponding type definitions. The copybook file might contain more than one COBOL copybook definition.

If your file contains multiple COBOL Copybook definitions, you can import the definitions into a single type tree in the following ways:

- Import a selected subset of the definitions.
- Import all the definitions.
- Import definitions from different COBOL Copybook files.

- **[Running the COBOL copybook importer in Design Server UI](#)**

Use the COBOL copybook importer to automatically generate COBOL copybook type trees.

- **[Runtime options](#)**

Runtime options are Character set and Byte order.

- **[COBOL copybook Specification](#)**

The COBOL copybook importer uses as input a copybook file (.cpy) containing data description entries, as defined by the ANSI X3.23 specification with the following exception and additions:

- **[Type tree Mapping](#)**

The COBOL copybook import operation maps data names to type tree names.

- **Clause**

The COBOL copybook importer supports below clauses in the Design Server UI:

Running the COBOL copybook importer in Design Server UI

Use the COBOL copybook importer to automatically generate COBOL copybook type trees.

The metadata for the type trees is in the form of COBOL copybooks.

The following procedure describes how to use the COBOL copybook importer to generate a type tree from COBOL copybooks.

There is an example of a COBOL copybook file in the *install_dir\examples\general\copybook* folder.

To run the COBOL Copybook Importer in Design Server UI:

1. Create a new project using Design Server UI. See the "Create a project topic" in Knowledge Center for more instructions.
2. Open the project in Design Server UI.
3. Select Schemas > Import and click on Import.
The Import Type window opens.
4. From the Type drop-down list, select COBOL copybook option and click Next.
The Import Properties window opens.
5. Select the COBOL copybook metadata source file that you want to import from the drop-down list of Copybook File Path property.
6. Select runtime options Character set and Byte Order from the selection list.
The Identification window opens.
7. Enter the name of COBOL copybook metadata source file in the Name field and click Import.
The COBOL copybook schema is imported successfully.

Runtime options

Runtime options are Character set and Byte order.

The runtime options, Character set and Byte order, describe the runtime data. The character set code represents a standard collection of letters, numbers, and symbols. The byte order set code represents a convention a machine processor uses to position its lowest byte within a word. The processor positions the lowest byte beginning either from the leftmost or the rightmost position. The default setting for these properties is Native.

COBOL copybook Specification

The COBOL copybook importer uses as input a copybook file (.cpy) containing data description entries, as defined by the ANSI X3.23 specification with the following exception and additions:

- Each entry must have either a data-name or the reserved FILLER clause. (This is an exception to the ANSI X3.23 specification).
- The COBOL copybook importer also supports these USAGE clauses:
 - COMPUTATIONAL-3
 - COMPUTATIONAL-4
 - COMPUTATIONAL-5
 - COMPUTATIONAL-X
 - COMP-3
 - COMP-4
 - COMP-5
 - COMP-X

Trees that are generated by the COBOL copybook importer can be used immediately for map development. However, depending on the contents of the copybook file, it might be necessary for the generated type tree to be modified using the Schema Designer.

Type tree Mapping

The COBOL copybook import operation maps data names to type tree names.

The COBOL copybook data names are converted to type tree names according to the following rules:

Characters that are not valid in type tree element names are replaced with the underscore (_) character. For example, the invalid \$ character in **monthly\$total** is replaced with the underscore (_) character. Therefore, **monthly\$total** becomes **monthly_total**.

Duplicate names are made unique by appending one or more numeric digits to the names. For example, there are two instances of **year**. You can make one of the instances **year1** and leave the other instance as **year**.

The import operation maps FILLER items by appending a number incrementally to the end of the name. Examples are FILLER1, FILLER2, FILLER3, and so on.

Clause

The COBOL copybook importer supports below clauses in the Design Server UI:

- VALUES clause
- USAGE clause
- PICTURE clause
- ORDER clause
- ORDER DEPENDING ON clause

The COBOL copybook importer does not support **REDEFINES** clause in the Design Server UI.

CSV Importer

The CSV Importer imports the content from the CSV file to the Design Server UI.

- [Running the CSV Importer in Design Server UI](#)
Use the CSV Importer to automatically generate CSV type trees.
- [CSV Importer Properties](#)
This topic lists the properties of CSV importer.

Running the CSV Importer in Design Server UI

Use the CSV Importer to automatically generate CSV type trees.

To run the CSV Importer in Design Server UI:

1. Create a new project using Design Server UI. See the Create a project topic in Knowledge Center for more instruction.
2. Open the project in Design Server UI.
3. Select Schemas > Import and click on Import.
The Import Type window opens.
4. From the Type drop-down list, select CSV option and click Next.
The Import Properties window opens.
5. Select the following properties to import the CSV:
 - Sample File Path
 - Sample Row Limit
 - Has Header
 - Logging
 - Format
 - Line Break
 - Delimiter
 - Quote
 - Escape
6. Click Next.
The Identification window opens.
7. Enter the name of CSV source file as you want in the Name field and click Import.
The CSV schema is imported successfully.

CSV importer Properties

This topic lists the properties of CSV importer.

Sample File Path

Specifies the path to the file to use for detecting and evaluating the schema structure.

Sample Row Limit

Specifies the number of rows to process in the specified sample file when detecting and evaluating the schema settings.

Has Header

Indicates the first line in the data is header line when it is set to true.

Logging

Specifies the level of logging to use. The default is Off. The value Informational means log informational, the value Errors Only means log error messages only, and the value Verbose means log debug and trace level messages along with the informational and error messages.

Format

Specifies the data format. Default is Strict CSV. The other data format is Custom.

Line Break

Specifies the line break (end of the line). Default is CR LF (Carriage Return Line Feed). Other options are as follows:

- LF: Line Feed
- CR: Carriage Return
- Newline: Platform Specific

Delimiter

Specifies the field delimiter. Default is Comma. Other options are as follows:

- Pipe
- Colon
- Semicolon
- Exclamation Mark
- Equals Sign
- Tilde
- Space
- Tab
- Null
- Custom

Quote

Specifies the enclosing character. Default is Double Quote. Other options are Single Quote and None.

Escape

Specifies the escape release character. Default is Double Quote. Other options are as follows:

- Backslash
- Exclamation
- Single Quote
- Null
- None
- Custom

JSON Importer

Using a JSON document to define a schema is termed as importing the JSON document. A schema defined through this process is referred to as a native JSON schema. Two types of JSON documents can be utilized for defining a native JSON schema:

- An imported JSON document can include example data, which is employed to establish the schema. Schemas resulting from these documents are termed JSON template schemas.
- Alternatively, the imported JSON document can adhere to the JSON Schema format. In such instances, the resultant schema embodies the specifications outlined in the JSON Schema document. These schemas are referred to as JSON Schema schemas.
- [**Importing JSON Schema in Design Server UI**](#)
Import the JSON schema in Design Server UI:
- [**Importing JSON Template in Design Server UI**](#)
Import the JSON template in Design Server UI:
- [**The representation of JSON types in schemas**](#)
- [**Using native JSON schemas in a map**](#)

Importing JSON Schema in Design Server UI

Import the JSON schema in Design Server UI:

1. Open the project in Design Server UI.
2. Go to Schema and click on XSD/JSON.
XSD/JSON where you can upload the XML schemas, JSON templates and JSON schema.
- The Import XSD or JSON window opens.
3. In the Import XSD or JSON window do as follows:

- In the File field, browse a JSON schema from your local machine to Design Server UI.
 - In the Name field, enter the name of the JSON schema.
 - In the Folder field, enter the folder path where you want to import the JSON schema.
 - In the Description field, add description of the JSON schema, if required.
4. Click OK to import the JSON schema.

The JSON schema is imported in Design Server UI. You can see it in the Schema list.

Note: The JSON schema is not editable. To edit the JSON schema, you have to edit it in a local machine and re-upload it into design server.

Importing JSON Template in Design Server UI

Import the JSON template in Design Server UI:

1. Open the project in Design Server UI.
2. Go to Schema and click on XSD/JSON.
XSD/JSON where you can upload the XML schemas, JSON templates and JSON schema.
3. The Import XSD or JSON window opens.
4. In the Import XSD or JSON window do as follows:
 - In the File field, browse a JSON template from your local machine to Design Server UI.
 - In the Name field, enter the name of the JSON template.
 - In the Folder field, enter the folder path where you want to import the JSON template.
 - In the Description field, add description of the JSON template, if required.
5. Click OK to import the JSON template.

The JSON template is imported in Design Server UI. You can see it in the Schema list.

5. Edit the JSON template if you want.
Note: You can edit the JSON template unlike the XSD schema and JSON schema.

The representation of JSON types in schemas

In a native JSON schema, a JSON object is portrayed as a group type. The group type comprises components that directly correspond to the properties found within the JSON object. Each component's name, reflecting the type of the component, aligns with the respective property name in the JSON object.

Meanwhile, a JSON array is depicted as a component having an unbounded number of repetitions. For instance, when a component possesses an unlimited repetition count and represents a number item type, it effectively signifies a JSON array consisting of numeric values.

When it comes to JSON numbers, they are symbolized using number item types within the schema. Likewise, JSON strings and Boolean values are both depicted using text item types.

Using native JSON schemas in a map

You can use a native JSON schema in either an input card or an output card.

Differing from standard schemas, you cannot manually choose a specific type from a native JSON schema for a card. Instead, the system automatically selects the group type located under the root type for the card.

- [JSON cards and text encodings](#)

JSON cards and text encodings

When utilizing a native JSON schema in an input card, the automatic determination of text encoding for input data takes place during map execution.

When an output card uses a native JSON schema, the text encoding of card output is determined during map definition. The allowed text encodings for output data are UTF-8, UTF-16LE, UTF-16BE, UTF-32LE, and UTF-32BE.

Previously, the text encoding for the output of a native JSON output card was designated through a rule specified on a text item component termed "encoding." The component's path was denoted as encoding:<card name>. For instance, the rule might be '=NONE' resulting in the use of UTF-8. Alternatively, the rule could be '='UTF-16LE'" signifying the utilization of UTF-16LE.

The encoding item component has been eliminated from native JSON schemas. Currently, the text encoding for an output card is determined by selecting a UTF encoding for the Data Language card property. This property exclusively pertains to output cards employing a native JSON schema that lacks the encoding item type. In the absence of a selection, UTF-8 is defaulted for the output card. Existing maps and schemas utilizing the encoding item type remain valid.

Type properties

The properties of a type define the characteristics of the data objects of that type.

For *item* types, properties define whether that item is text, a number, a date and time, or a syntax value. Properties include such characteristics as size, pad characters, and justification.

For *group* types, the properties are related to the format of that group. The format of a group may be explicit or implicit. In addition, type properties include syntax objects that appear at the beginning or end of the object, as well as release characters.

- [Defining type properties](#)
 - [Basic type properties](#)
 - [Item properties](#)
 - [Group properties](#)
 - [XML properties in the schema](#)
-

Defining type properties

You can define specific properties for each type. Type properties generally consist of the following:

- Definition of a type's **Class**, **Group** or **Item** properties
 - Type **Initiator**, **Terminator**, and **Release** character
 - A list of where a type is used in the definitions of other types
-

Basic type properties

The following type properties are common to categories, groups, and items:

- [Name](#)
- [Class](#)
- [Description](#)
- [Intent](#)
- [Partitioned](#)
- [Order Subtypes](#)
- [Initiator](#)
- [Terminator](#)
- [Release Characters](#)
- [Empty](#)
- [National Language](#)
- [Document Type](#)
- [Where Used](#)

Item-specific properties are discussed in more detail in ["Item Properties"](#). Group-specific properties are discussed in more detail in ["Group Properties"](#).

- [Name](#)
 - [Class](#)
 - [Description](#)
 - [Intent](#)
 - [Partitioned](#)
 - [Order subtypes](#)
 - [Initiator](#)
 - [Terminator](#)
 - [Release characters](#)
 - [Empty](#)
 - [Document Type](#)
 - [Where used](#)
 - [Symmetric swapping](#)
 - [Orientation](#)
 - [Shaping](#)
-

Name

The name of the type should be as descriptive as possible and reflect the information that the type represents.

A type name must conform to the following guidelines:

- A type name cannot be more than 256 characters long.
- A type name cannot be a reserved word or contain a reserved symbol. For a list of reserved words, refer to Design Studio Introduction.
- A type name cannot contain only digits and/or periods.
- A type name may contain numbers and special characters.
- A type name cannot contain spaces. Use an underscore instead.
- A type name can contain the following:
 - Letters
 - Digits
 - ASCII characters 128-255
 - Special characters: #, ~, %, _, ? or \
 - Double-byte characters (Japan edition only)
- A type name cannot have the same name as another type on the same level in the same schema.

Class

The class of a type describes whether the type represents a simple data object (item), a complex data object (group), or whether it is used to organize types (category). Define the class of the selected type by selecting the class from the drop-down list in the **Value** column.

Category

Categories organize types that have common properties. Each category has a set of item properties and a set of group properties. Subtypes of the category inherit these properties.

A category does not define data objects in detail and it does not represent data to be used in a map. You never map a category type, so a category type does not have components. A category type may be a component of a partitioned group, after a component with the identifier attribute. A category cannot be a component of a non-partitioned group.

Item

The item type represents a simple data object that does not consist of any objects. An item does not have components.

Group

The group type represents a complex data object that consists of components.

To change a type's class, select the type and choose the desired class for the **Class** property in the Properties window.

If you change a type to an item, all the types in its subtree become items. If you change a type to a group, all the types in its subtree become groups. This is because the schema has a classification hierarchy and all the nested subtypes of an item must be items. Similarly, all the nested subtypes of a group must be groups.

Description

Use this property to record a brief description of the type. The description entered is for informational purposes only. It is not used for identifying data objects at runtime.

It is good practice to add a meaningful description of the type when you are defining it.

Intent

Indicates whether the type is a general type or an XML type.

The type can only be an XML type if the type tree was created using the XML DTD Importer or the XML Schema Importer.

Refer to ["XML Properties in the Type Tree"](#) for more information about type properties created from XML.

Partitioned

If the data of this type can be divided into mutually exclusive subtypes, it can be partitioned. For information about partitioning, see ["Partitioning"](#).

Yes

Partitioning is enabled for this type.

No

Partitioning is not enabled for this type.

Order subtypes

Choose the method in which the subtypes of this type will be added or viewed in the schema.

Ascending

Add and view subtypes of this type in alphabetic or numeric order.

Descending

Add and view subtypes of this type in reverse alphabetic or numeric order.

Add First

Add subtypes of this type to the top of the type list.

Add Last

Add subtypes to the bottom of the type list.

To manually reorder the subtypes in a schema, the **Order Subtypes** property must be either **Add First** or **Add Last**.

Initiator

An initiator is a syntax object that appears at the beginning of a data object. Defining an initiator for a type specifies that when data of that type appears, the initiator appears at the beginning of the data object. The initiator becomes part of the data type definition.

None

There is no initiator.

Literal

The initiator is literal. Expand the **Initiator** property to enter the literal initiator value and data language of the initiator value.

Variable

Allow for possible values. Expand the **Initiator** property to define the variable terminator **Default**, **Item**, and **Find** properties.

If each record begins with an asterisk *, define the * as a literal initiator of the record type.

The following data represents the classes in a college English department. An asterisk * is displayed at the beginning of each **ClassRecord**.

For information about other symbols used in the Value field of the Initiator property, see the Type Tree Importers documentation .

Terminator

A terminator is a syntax object that appears at the end of a data object. The terminator becomes part of the data type definition.

None

There are no terminators.

Literal

A constant value. Expand the **Terminator** property to define the literal terminator **Value**.

Variable

Allow for possible values. Expand the **Terminator** property to define the variable terminator **Default**, **Item**, and **Find** properties.

For example, a carriage return/linefeed (CR/LF) at the end of a record is the record's terminator.

Generally, if the data ends with a given syntax object, you should define a literal terminator. For example, it is very common to define a CR/LF as a terminator of a record when you know that the record always ends with a CR/LF, regardless of where the record appears.

Release characters

A release character is a one-byte character in your data indicating that the character(s) following it should be interpreted as data, not as a syntax object. The release character is not treated as data, but the data that follows it is treated as actual data.

- [Building release characters for output data](#)
 - [Guidelines for using release characters](#)
 - [Release character example](#)
-

Building release characters for output data

If a release character is defined for a type, a release character is inserted for each occurrence of a syntax object in the data of any item contained in that type.

Guidelines for using release characters

Guidelines for using release characters include the following:

- Release characters apply to character data only, *not* binary data.
 - Characters defined as pad characters are not released.
 - The maximum size of an item does *not* include the release characters.
-

Release character example

The group type **Record** type has a literal delimiter of , and a release character of ?. The group type **Record** has three item components. Data for the record looks like the following:

Miller?, MD,Harkin Hospital,1996

The ? releases the comma after Miller.

In the first field, the actual data value is **Miller, MD**. Because the comma appears as part of the data, it is necessary to have the release character ?, which indicates that the , following it is data, *not* a delimiter. This data would be interpreted as the following:

Data for component #1: **Miller, MD**

Data for component #2: **Harkin Hospital**

Data for component #3: **1996**

A release character can apply to a delimiter, a terminator, or even the release character itself.

If a release character appears in the data and it is not followed by a syntax item, the release character is ignored.

Empty

The **Empty** property provides alternative type syntax for groups or items when they have no data content.

When the **Empty** property is specified for a type and there is no data content, the **Empty** syntax is displayed. For example, this can be used for XML data that contains either start and end tags or an empty tag.

You can use the **Empty** type property instead of syntax object items to potentially improve the schema runtime processing time (during data validation).

The following options are available for the **Empty** property.

None

Default setting. Select None if you do not have an initiator, terminator, or release character.

Literal

A constant value. When **Literal** is selected, the **Value**, **Ignore Case**, **Required**, and **National language** sub-fields become available. You can use only one literal to indicate a zero-length data item.

- [Empty type property example](#)

Empty type property example

Consider an XML element called **Comment**. Presuming that this element has a simple content of an arbitrary length, it can be represented in a schema as a text item, with the initiator value `<Comment>` and the terminator value `</Comment>`.

This item can then be used to validate the following data:

```
<Comment>Some comment...</Comment>
```

The resulting value for the item will be:

```
Some comment...
```

As another example, the following data is also successfully validated:

```
<Comment></Comment>
```

The resulting value for the item will be a zero-length string.

The problem occurs with the following data: `<Comment/>`

From the XML perspective, this data is equivalent to the `<Comment></Comment>` data shown above, as it represents the **Empty** element **Comment**. However, from the schema perspective, this data is invalid because it does not start with `<Comment>` and does not end with `</Comment>` as required by the initiator and terminator item property values.

This is where the **Empty** property is useful. By defining the **Empty** property for the item to have a value of `<Comment/>`, it will be possible to validate the data `<Comment/>`.

Even if the data does not match the syntax described by the initiator and terminator properties, it will be validated because it will match the **Empty** property value. It will be treated as an *empty item*, that is, the resulting value for the item will be a zero-length string, similar to the example shown previously.

Document Type

The **Document Type** setting is a component of Document Verification. A requirement of using Document Verification is to set the **Document Type** properties of the associated schema object.

To use the Document Verification option for XML documents, you must do the following:

- Set the **Document Type** property to **XML**.
- Set the Document Type Metadata value to **Schema** or **DTD**.
- Specify the **Location** of the DTD or Schema file.

Default

This is the default value, which indicates a non-XML document type.

XML

Indicates an XML document type. When this value is set to **XML** and the appropriate **DocumentVerification** map settings are in place, the XML document will be validated by an external program in addition to the standard data validation process.

See the Map Designer documentation for information about using the Document Verification option.

Where used

Where Used shows how and where the type is used within other types in the tree.

Symmetric swapping

When the input and output objects have different symmetric swapping characteristics, the Symmetric Swapping property is considered "on" for the related mapping rule.

Examples

Original text	Resulting text
abcF (E) Dghi Input Object No swapping Output Object Swapping	abcF (E) Dghi
abcF (E) Dghi Input Object Swapping Output Object No swapping	abcF (E) Dghi
abcF (E) Dghi Input Object No swapping Output Object No swapping	abcF) E (Dghi
abcF (E) Dghi Input Object Swapping Output Object Swapping	abcF) E (Dghi

Orientation

Orientation is to change characters from one direction to a different direction.

Example

A simple orientation example is to define English characters "abc" as a bidirectional object with left-to-right (LTR) orientation. If you moved the object to a right-to-left (RTL) object, the output should be "cba". The following example demonstrates a string of characters that change in orientation.

Original text	Resulting text
ABC DEF Ā Ā Ā Input left-to-right (LTR)	Ā Ā Ā FED CBA
Output right-to-left (RTL)	

Shaping

Text shaping is the concept of combining or separating characters where possible. The Arabic usage is with ligatures (such as the LAM, ALEF, and LAM-ALEF characters). For example, in a non-shaped word, both LAM and ALEF characters would appear separately. In a shaped word, the two characters would be replaced with a single LAM-ALEF character.

Numeric shaping is the idea that normal regular numerals (0–9) can be shaped to be Arabic-Indic

Examples

- **Ù, = LAM**
- **í° = ALEF**
- **Ûm'i»» = LAM ALEF**

Original text	Resulting text
---------------	----------------

Original text	Resulting text
abc>>>def	abcÜ,,i°def
Input	
Shaped	
Output	
Unshaped	
Original text	Resulting text
abcÜ,,i°def	abc>>>def
Input	
Unshaped	
Output	
Shaped	

Item properties

Item types define data for a selected type. Item types are divided into the following subclasses: **Number**, **Text**, **Date & Time**, and **Syntax**. Each item subclass has a specific set of properties.

- [Item subclass](#)
- [Number item subclass properties](#)
- [Text item subclass properties](#)
- [Date & Time item subclass properties](#)
- [Syntax item subclass properties](#)

Item subclass

The subclass of an item represents the characteristics of the data. Both category and item types have the **Item Subclass** property.

To define the subclass of an item type:

1. Access the **Properties** for the selected category or item type.
2. For the **Item Subclass** property, select a value that defines the data for the type:

Number

Any number excluding active syntax objects. The interpretation can be either character or binary. See "[Number Item Subclass properties](#)".

Text

Any character excluding active syntax objects. The interpretation can be either character or binary. See "[Text Item Subclass properties](#)".

Date & Time

A valid date and time format based on the data. The interpretation can be either character or binary. See "[Date & Time Item Subclass properties](#)".

Syntax

Syntax objects are used as separators between portions of data. A restricted set of values used to define a dynamic delimiter, initiator, terminator, or release character. The interpretation is character. A syntax object cannot be longer than 120 bytes. Syntax objects, used as syntax, cannot be mapped. See "[Syntax Item Subclass properties](#)".

For text and syntax objects, character size constraints can be in bytes or characters. Numbers are considered in digits and date-time formats are sized by the format, such as CCYYMMDD.

Number item subclass properties

Number item subclass properties can be of **Character** or **Binary** value. When you choose **Number** as the **Item Subclass** value, you must choose an **Interpret as** value: **Binary** or **Character**.

The **Interpret as** value defines how the data should be interpreted. Items with a subclass of **Number**, **Text**, and **Date & Time** can be interpreted as either character or binary. Items with a subclass of **Syntax** can be interpreted as character only.

- [Interpret as binary](#)
- [Length \(bytes\)](#)
- [Byte order](#)
- [Interpret as character](#)
- [Size \(digits\)](#)
- [Separators](#)
- [Pad](#)
- [Restrictions](#)
- [National language](#)
- [NONE and Zero](#)
- [Places](#)
- [E-Notation](#)

E-Notation gives you the ability to process and build decimal numbers expressed as E-Notation.

Interpret as binary

Binary text items have content size and pad properties. Binary data is required to be sized or of a fixed size.

Binary number items interpret the data as a binary number or as a byte stream.

Items with an item subclass of **Text** can be interpreted as character or binary.

When **Interpret as** is set to **Binary**, the binary presentation options are Integer, Float, Packed, or BCD.

- [Binary integer presentation](#)
- [Binary float presentation](#)
- [Binary packed presentation](#)
- [Binary BCD presentation](#)

Binary integer presentation

The **Integer** value is a whole number.

Length(bytes)

Can have a length of **1**, **2**, **4**, or **8** bytes

Byte order

See "[Byte order](#)".

Sign

See "[Sign](#)".

Binary float presentation

The **Float** value is a number with decimals in a location as needed.

Length(bytes)

Can have a length of **4**, **8**, or **10** bytes

Binary packed presentation

The **Packed** value is a number that has size, decimal places, and sign properties.

The binary packed properties include Length, Implied places, and Sign.

Length

Can have a length of **1 - 16** bytes

Implied places

Can be from **0 - 31**

Sign

Can be **Trailing-** or **Trailing+**

For **Packed** numbers, the **Implied places** cannot exceed the **Length**.

Binary BCD presentation

The **BCD** value is a binary coded decimal number that has size.

Length (bytes)

Use the **Length (bytes)** property to select the number of bytes that equals the length of the item.

The **Length(bytes)** property is an option when the **Number** item subclass property is interpreted with a **Binary** value.

Byte order

Use the **Byte order** property to define the way that bytes in the data are ordered by selecting one of the following values from the Byte order drop-down list.

Value	Description
BigEndian	The most significant byte has the lowest address. (Usually systems such as IBM, HP, and Solaris.)

Value	Description
Little Endian	The least significant byte has the lowest address. (Usually systems such as Intel and VAX.)
Native	The order is dictated by the platform.

Byte order is an option when the **Number** item subclass is interpreted with a **Binary** value.

Interpret as character

Character number items interpret the data as symbolic data. Symbolic data has the same meaning on different computers. For example, a comma is symbolic because it has the same meaning, regardless of the computer type.

Character text items can have content size and pad properties.

When the **Interpret as** value is defined as **Character**, the **Release** property is specific to the **Character** value and is *not* available as an option for a **Binary** value. For more information, see "[Release characters](#)".

When **Interpret as** is set to **Character**, integer, decimal, and zoned presentation options are available.

- [Character integer presentation](#)
- [Character decimal presentation](#)
- [Zoned character presentation](#)
- [Places > implied](#)

Character integer presentation

The **Integer** value is a whole number.

Further information on the **Integer** value is found in "[Integer Separators](#)".

Character decimal presentation

The **Decimal** value is a number that contains decimals.

More information about the **Decimal** value is found in section "[Decimal Separators](#)".

Zoned character presentation

The **Zoned** value is a number that has a size, decimal place, pad, and sign properties.

Use the "Size (digits)" property to specify the size of the zoned number. The size of a zoned number can also be specified in terms of minimum and maximum. The minimum size must be less than or equal to the maximum size.

When the content size of a character-zoned number is used, the last digit and the sign are combined into the same digit but the content size does *not* include the initiator, terminator, release characters, or pad characters.

When the content size of a character-zoned number is important (for example, when the **SIZE** function is used on the character zoned number), the content size includes the sign but does not include the initiator, terminator, release characters, or pad characters.

Use the "[Sign](#)" property to define whether the zoned number will be signed. The default value is **No**.

The size of a zoned number is specified in terms of the minimum and maximum number of digits. If the size is specified with the **Min** and **Max** properties, the sign is *not* included.

Example

Zoned Number Not Signed	Signed Zoned Number
1230	123{
1231	123A
(Length = 4 digits)	(Length = 4 digits)

Places > implied

Numbers have implied decimal places. The number of decimal places cannot exceed the length specification. The list of available decimal places varies with the length selected. In the **Value** column, enter a total number of decimal places (not to exceed 31).

Size (digits)

Use the **Size (digits)** property to specify the minimum and maximum size of a number item.

Min

The minimum number of digits of the data object.
The default value is 0.

Max

The maximum number of digits of the data object.

If there is no maximum size, a **Max** value is not required.

- [Excluded from min and max size](#)
- [Size example](#)

Excluded from min and max size

For character number items, there are certain objects excluded from the **Min** or **Max** count. For each **Presentation** option, the following table lists what is excluded from the minimum or maximum **Size (content)** count.

Presentation:	Integer	Decimal	Zoned
	initiators	initiators	initiators
	pad characters	pad characters	pad characters
	release characters	release characters	release characters
	separators	separators	terminators
	symbols	symbols	
	terminators	terminators	

Size example

In the following ASCII number there are a total of 12 characters:

+1234567.999

However, because the initiator (+) and separator (.) are not counted, the number would validate for a maximum size of 10 (**Max=10**).

Separators

Integer and decimal number items can have a separator for thousands and fractional places. If you opt to have a separator, choose **Yes** and sub-options will appear that enable you to choose the format and syntax.

The **Zoned** value of the **Presentation** property does not have separators.

- [Integer separators](#)
- [Separator > format](#)
- [1000's syntax > value](#)
- [1000's syntax \(literal\) > value](#)
- [1000's syntax \(variable\) > default](#)
- [1000's syntax \(variable\) > item](#)
- [1000's syntax \(variable\) > find](#)
- [Decimal separators](#)
- [Separators > format](#)
- [Separators > 1000's syntax](#)
- [Separators > fraction syntax](#)
- [Sign](#)

Integer separators

For integer numbers, the separator is the thousands separator.

Integer character numbers have separator properties of:

- Format
- 1000's Syntax
- Value

These properties specify the placement and value of the thousands separator.

Separator > format

Use the Separators Format property to specify the placement and value of the thousands separator.

To define the separator format, select one of the separator formats from the drop-down list in the **Value** column.

#[.]###

The thousands separator is not required on input, but if present, it must be in the proper location. The separator is not built on output.

#.###

The thousands separator is required on input in the proper location. The separator is built on output.

1000's syntax > value

Define the thousands separator as either literal or variable. Select one of the following from the drop-down list in the **Value** column:

Literal

The thousands syntax is a constant literal specified in the **1000's Syntax > Value** column.

Variable

The thousands syntax allows for variable values. Use the **1000's Syntax > Default**, **Item**, and **Find** properties to define the variable fraction separator.

1000's syntax (literal) > value

For the literal thousands separator, define the literal separator by typing the literal separator character in the **Value** column or by clicking the browse button to display the Symbols dialog box from which you can insert any non-printable value.

A separator can be from one to 120 bytes in length. A separator cannot start or end with a digit and cannot be the? character. The ? character is a reserved character that can be used as a wildcard to represent any single valid character.

1000's syntax (variable) > default

Use the **1000's Syntax > Default** property to define the default variable separator value for the thousands place. The default literal separator value for thousands syntax is a comma.

Enter the default variable separator character or click the browse button to display the Symbols dialog box in which you can insert any non-printable value.

1000's syntax (variable) > item

For integer numbers with a variable thousands separator, the **Separator** property is **Yes**, and the **1000's Syntax** is **Variable**, you can specify an item type for the variable thousands separator.

You can select the desired syntax item from the drop-down list. Or, with the focus on the **Item** property (in the **Value** column), press **Alt** and drag the default separator type item from the schema editor into the **Value** column.

An item type with a class of **Syntax** must exist in the schema to be a valid **1000's Syntax Item**.

1000's syntax (variable) > find

For integer numbers with a variable thousands separator, the **Separator** property is **Yes** and the **1000's Syntax** is **Variable**. The value of the separator can be the current value or the value of the separator can be determined each time an occurrence of that type is found.

Yes

Determine the value of the separator each time an occurrence of that type is found. After the value of that separator is found, that particular value is used until it is reset either by another Find or by the occurrence of that separator as a component.

No

The system uses the value to which the separator item is currently set or, if not set, it uses the default value.

Decimal separators

For decimal numbers, the separator is the thousands separator and the fractional separator.

Decimal character numbers have separator properties of:

- Format
- 1000's Syntax
- Fraction syntax
- Value

These properties specify the placement and value of the fractional separator.

The item is assumed to have an implicit number of decimal places. To specify the number of implied decimal places, use the Item Subclass Places property to specify the number of implied decimal places.

Sepators > format

Use the Separators Format property to specify the placement and value of the fraction separator.

To define the separator format, select one of the separator formats from the drop-down list in the **Value** column.

#####[.##]

The fractional separator is present if there are fractional digits. There is no thousands separator.

####,##

The fractional separator is always present, even if there are no fractional digits. There is no thousands separator.

#[,###][.##]

The fractional separator is present if there are fractional digits. There is an optional thousands separator.

#[,##][.##]

The fractional separator is always present. There is an optional thousands separator.

#,###[.##]

The fractional separator is always present. There is always a thousands separator if there are more than three whole number digits.

Sepators > 1000's syntax

Use the Separators 1000's Syntax to define the thousands syntax. Select one of the following from the drop-down list in the **Value** column:

Literal

The thousands syntax is a constant literal specified in the **1000's Syntax > Value** column.

Variable

The thousands syntax allows for variable values. Use the **1000's Syntax > Default**, **Item**, and **Find** properties to define the variable fraction separator.

See the following topics for additional information:

- ["1000's Syntax \(Literal\) > Value"](#)
- ["1000's Syntax \(Variable\) > Default"](#)
- ["1000's Syntax \(Variable\) > Item"](#)
- ["1000's Syntax \(Variable\) > Find"](#)

Sepators > fraction syntax

Use the **Fraction syntax** property to define the fraction syntax as either a constant literal value or a variable value.

You can define the **Fraction syntax** as literal or variable.

Literal

A constant value.

Use the Fraction syntax...Value property to select a literal fraction separator from the **Symbols** dialog box.

Variable

Allow for variable fraction separator values. The following options are available to define the variable fraction separator:

- **Default** - Use this property to define the default variable separator value for fractions. The default literal separator value for fractions is a period. Enter the default variable separator character or click the browse button to display the Symbols dialog box in which you can insert any non-printable value.
- **Item** - For decimal numbers with a variable fractional separator, the **Separator** property is **Yes** and the **Fraction syntax** is **Variable**, you can specify an item type for the variable fraction separator.
Note: An item type with a class of **Syntax** must exist in the schema to be a valid **Fraction syntax Item**.
- **Find** - For decimal numbers with a variable fraction syntax, the **Separator** property is **Yes** and the **Fraction Syntax** is **Variable**, the value of the separator can be either the current value or the value of the separator can be determined each time an occurrence of that type is found. Select an option:
 - Yes - Determines the value of the separator each time an occurrence of that type is found. After the value of that separator is found, that particular value is used until it is reset by another find or by the occurrence of that separator as a component.
 - No - The system uses either the value that the separator item is set to currently, or, if it is not set, uses the default value.

Sign

Use the Item Subclass Sign property to define whether the number is signed for either the **Integer** or **Decimal** value. A sign is a symbol that identifies a number as being either positive or negative. A positive sign is plus (+); a negative sign is negative (-).

Yes

The number is signed. Expand the **Sign** property to define the sign values. If the **Sign** property is **Yes**, a minimum of at least one sign value (**If number is +**, **If number is -**, or **If number is 0**) must be specified and at least one value must be required on input.

No

The number is not signed. This is the default setting.

When you select Yes, you can further define the sign values. The following sub-options are available:

Leading-

The sign precedes the number when it is negative only. For input, a sign is required for negative data, but is optional for positive data.

Trailing-

The sign follows the number when it is negative only. For input, a sign is required for negative data, but is optional for positive data.

Leading+

The sign always precedes the number. A sign is required for input and output data.

Trailing+

The sign always follows the number. A sign is required for input and output data.

Custom

Defaults to **Leading-** but can be changed.

Sign values may be specified for the following:

- **If Number is +** (positive numbers)
- **If Number is -** (negative numbers)
- **If Number is 0** (value of zero)

For each value (**If number is +**, **If number is -**, or **If number is 0**), specify the following:

Leading sign

Enter the symbol to be placed before a number.

Trailing sign

Enter the symbol to be placed after a number.

Required on input

If the number has a sign, at least one value (**If number is +**, **If number is -**, or **If number is 0**) must be required on input. Select an option from the drop-down list:

Yes

The sign is mandatory on input.

No

The sign is optional on input.

Pad

Pad implementation determines how the pad value is sized during validation and how the pad value is sized when written as output.

When the data value to be mapped to the target item is smaller than the minimum length of that item, pad characters can be used to pad the data to that minimum length. Input data can contain both content and pad characters. Output data is built according to the pad definitions of the types. Bytes are the default measurement for sizing, however, both byte and character sizing options are available.

Yes

Enables the **Pad** option. Allows the item to contain both content and pad characters.

No

All data is assumed to be content on input; no pad characters are built on output.

- [Pad > value](#)
- [Pad > Padded to](#)
- [Padded to > length](#)
- [Padded to > SizedAs](#)
- [Padded to > CountsTowardMinContent](#)
- [Padded to > Allow Excess Trailing Pads](#)
- [Pad > justify](#)
- [Pad > apply.pad](#)
- [Pad > fill](#)

Pad > value

Use the Pad Value property to define a 1-character pad character. The default pad value is 0.

Type the pad character symbol between angled brackets <> in the **Value** field or click the browse button to display the Symbols dialog box to insert any non-printable value.

For example, to enter a **FormFeed** value for the pad character, in the **Value** field enter <FF> or click the browse button to select the **FF (FormFeed)** symbol from the Symbols dialog box.

Pad > Padded to

Use the Pad > Padded to property to define whether the data item is padded to a fixed size or to the minimum content size defined for the data object.

Fixed size

The data object is padded to a fixed size (in bytes or characters) that you specify in the Length field.

For any item padded to a fixed size, the item must have a value specified for the Size (content)>Max property and the Padded to>Length value must be greater than or equal to the Size (content)>Max value.

Min Content

The data is padded to the value specified as the minimum size (in the Size (content)>Min field).

You can further specify whether to count pad characters toward the length using the **CountsTowardMinContent** subproperty.

Padded to > length

To pad to a fixed size, you must specify the length.

Input data can contain both content and pad characters but when an item is built for output, the item is padded to the number of bytes or characters specified in this field.

The Padded to > Length value must be greater than or equal to the Size > Max value.

You can specify the method of measurement (bytes or characters) using the **SizedAs** subproperty.

Padded to > SizedAs

Use the Padded to > **SizedAs** property to specify that padding is measured in either bytes or characters. The default setting is Bytes.

Trace file and audit log lengths are always reported in bytes.

Table 1. Pad implementation example (bytes compared to characters)

Scenario for empty field:	Result:
Data language is UTF-8 Pad > Value = Å (hexadecimal representation = 0xC3 0x80) Pad > Padded to = Fixed Size Padded to > Length = 5 Padded to > SizedAs = Bytes	Two Å pad characters using 4 bytes.
Data language is UTF-8 Pad > Value = Å (hexadecimal representation = 0xC3 0x80) Pad > Padded to = Fixed Size Padded to > Length = 5 Padded to > SizedAs = Characters	Five Å pad characters totaling 10 UTF-8 bytes.

Padded to > CountsTowardMinContent

When you select Padded to > Min Content, the **CountsTowardMinContent** field is displayed. Use this setting to specify if pad characters should count toward the length of an object when determining if it meets its minimum content length.

Yes

Pad characters are included in the count for the length of an object.

No

Pad characters are not counted toward the length of an object.

You can use the **Propagate** function here to propagate the present **CountsTowardMinContent** value to the subtypes of the present type, if present.

If you are using the trace option, there might be cases where in the trace file the content length appears to be different than the input length. A trace file lists the input length (the length used to validate an object), which includes the input length of each object plus the pad characters. In the case of numbers, signs and separators are also counted in the length.

CountsTowardMinContent > AcceptAllPads

The **AcceptAllPads** property is applicable to data objects that contain only pad characters. Use this property when you need an object that contains only pad characters (no content) to either pass or fail validation depending on whether it is a mandatory or an optional type, regardless of the minimum size requirement.

Yes

When an object contains only pad characters, the minimum size requirement is ignored and the object passes size validation. The object then passes or fails type validation depending on whether it is a mandatory or an optional type.

No

(Default setting) Pad characters do not count toward the minimum size requirement, therefore an object that contains only pad characters and that does not meet the minimum size (content) requirement fails size validation.

Example

The results of the following examples are based on these values:

Size (content)

5

Pad

Yes
Pad > Padded to
 Min Content
Padded to > CountsTowardMinContent
 No

Default Behavior: When **AcceptAllPads** is set to **No**, the following results occur:

Input Data (X = pad character)
Valid?

ABCDE
 Yes
ABC
 No
ABCXX
 No
XXXXX
 No

When **AcceptAllPads** is set to **Yes**, the following results occur:

Input Data (X = pad character)
Valid?

ABCDE
 Yes
ABC
 No
ABCXX
 No
XXXXX
 Yes (when type is optional)
No (when type is mandatory)
X
 Yes (when type is optional)
No (when type is mandatory)

Padded to > Allow Excess Trailing Pads

When you select Padded to \geq Min Content, the Allow Excess Trailing Pads property is displayed. Use this setting to flag an element as not valid when the following conditions apply:

- The element is padded to a minimum size.
- The number of pad characters exceeds the number that is required to achieve the minimum size.

Pad > justify

Use the **Justify** property to specify whether the data is padded to the left or right.

Left

The data will be on the left and will be padded (if necessary) on the right.

Right

The data will be on the right and will be padded (if necessary) on the left.

Use the **TRIMLEFT** and **TRIMRIGHT** functions to exclude pad characters from the justified side.

In the following example, xxxxxxxxxxx represent 10 spaces:

For an input of 1234567891xxxxxxxx with a right justified pad, the rule =SIZE(input) returns 20. For the same input, the rule =SIZE(TRIMRIGHT(input)) would return 10 (removing the padding on the right).

Pad > apply pad

Use the **Apply pad** property to specify when to apply the pad character. Choose a value from the drop-down list.

Fixed Group

Apply the pad characters only when the item appears in a fixed group.

Any context

Apply the pad characters when the item appears in any context.

Each item in a fixed group must be padded to a fixed size or have the same value for the minimum and maximum content size.

For example, suppose the item **Name** has a space pad character with this value:

Mary<sp><sp>

If the **Apply pad** property is **Fixed Group**, the two spaces at the end are treated as pad characters only when the item appears in a fixed group. If the **Apply pad** property is **Any context**, then the spaces are always treated as pad characters.

Each item in a fixed group must be padded to a fixed size or have the same value for the minimum and maximum content size.

Pad > fill

For padded signed numbers, this option determines placement of pad characters. Choose a value from the drop-down list.

After sign

Pad characters are placed after the sign.

Before sign

Pad characters are placed before the sign.

Restrictions

Restrictions of an item are the valid values of that item. When the **Interpret as** value is defined as **Character**, the **Restrictions** property is specific to the **Character** value and is not available as an option for a **Binary** value.

Depending on other **Item Subclass** settings, the **Restrictions** property can be set to **Value**, **Character**, or **Range**.

- [Restrictions > ignore case](#)
- [Restrictions > rule](#)

Restrictions > ignore case

By default, restrictions are case-sensitive. To enable or disable the **Ignore case** property of the **Restriction**, choose from one of the following options:

Yes

Ignore case of restrictions in item type properties.

No

Do not ignore case-sensitive restrictions.

For example, if **Ignore Case = Yes** and the restriction **Value** is **ft**, the data values **ft**, **fT**, **Ft**, and **FT** are all valid.

Restrictions > rule

This setting indicates whether you are going to include or exclude the criteria (as "valid" or "invalid") specified for the restrictions.

Include

Includes the restriction values as valid data.

Exclude

Excludes the restriction values as invalid data.

National language

The National language default value is Western. For initiator, terminator, and release character **Literal** values, you can optionally specify a ["Data language"](#).

When the **Interpret as** value is defined as **Character**, the **National language** property is specific to the **Character** value and is not available as an option for a **Binary** value.

- [National language > data language](#)
- [Supported code pages](#)

National language > data language

Use the National language > Data language property to define the data language or character set of this character text item.

Supported code pages

IBM Transformation Extender supports the following code pages:

Arabic
ASCII (deprecated)
Big5
Big5-HKSCS (IBM)
BOCU-1
CESU-8
CII Kanji (deprecated)
Cyrillic
EBCDIC (deprecated)
ebcdic-ar
ebcdic-cp-ar1
ebcdic-cp-ar2
ebcdic-cp-be/ch
EBCDIC-CP-DK/NO
ebcdic-cp-es
ebcdic-cp-fi/se/sv
ebcdic-cp-fr
ebcdic-cp-gb
ebcdic-cp-he
ebcdic-cp-it
ebcdic-cp-roece/yu
ebcdic-de
ebcdic-he
ebcdic-is
EBCDIC-JP-kana
ebcdic-xml-us
EUC (deprecated)
EUC-CN
EUC-JP
euc-jp-2007
EUC-TW
euc-tw-2014
GB_2312-80
gb18030
GBK
Greek8
Hebrew
hp-roman8
HZ-GB-2312
IBM Kanji (deprecated)
ibm-037 (ebcdic-cp-us/ca/wt/nl)
ibm-1006
ibm-1025
ibm-1026
ibm-1047
ibm-1047-s390
ibm-1097
ibm-1098
ibm-1112
ibm-1122
ibm-1123
ibm-1124
ibm-1125
ibm-1129
ibm-1130
ibm-1131
ibm-1132
ibm-1133
ibm-1137
ibm-1140 (ebcdic-us-37+euro)
ibm-1140-s390
ibm-1141 (ebcdic-de-273+euro)
ibm-1141-s390
ibm-1142 (ebcdic-dk/no-277+euro)
ibm-1142-s390
ibm-1143 (ebcdic-fi/se-278+euro)
ibm-1143-s390
ibm-1144 (ebcdic-it-280+euro)
ibm-1144-s390
ibm-1145 (ebcdic-es-284+euro)
ibm-1145-s390
ibm-1146 (ebcdic-gb-285+euro)
ibm-1146-s390
ibm-1147 (ebcdic-fr-297+euro)
ibm-1147-s390
ibm-1148 (ebcdic-international+euro)
ibm-1148-s390
ibm-1149 (ebcdic-is-871+euro)
ibm-1149-s390
ibm-1153

ibm-1153-s390
ibm-1154
ibm-1155
ibm-1156
ibm-1157
ibm-1158
ibm-1160
ibm-1162
ibm-1164
ibm-1250
ibm-1251
ibm-1252
ibm-1253
ibm-1254
ibm-1255
ibm-1256
ibm-1257
ibm-1258
ibm-12712-s390
ibm-1276 (Adobe Standard Encoding)
ibm-1363 (korean)
ibm-1363_P110-1997
ibm-1364
ibm-1371
ibm-1373_P100-2002
ibm-1386_P100-2001
ibm-1388
ibm-1390
ibm-1399
ibm-16684
ibm-16804-s390
ibm-33722_P120-1999
ibm-37-s390
ibm-437
ibm-4517
ibm-4899
ibm-4909
ibm-4971
ibm-5123
ibm-5346
ibm-5347
ibm-5348
ibm-5349
ibm-5350
ibm-5351
ibm-5352
ibm-5353
ibm-5354
ibm-5471_P100-2006
ibm-720
ibm-737
ibm-775
ibm-803
ibm-813
ibm-8482
ibm-850
ibm-851
ibm-852
ibm-855
ibm-856
ibm-857
ibm-858
ibm-860
ibm-861
ibm-862
ibm-863
ibm-864
ibm-865
ibm-866
ibm-867
ibm-868
ibm-869
ibm-874
ibm-875
ibm-901
ibm-902
ibm-9067
ibm-916
ibm-921
ibm-922

ibm-930
ibm-933
ibm-935
ibm-937
ibm-939
ibm-9447
ibm-9448
ibm-9449
ibm-949_P110-1999
ibm-949_P11A-1999
ibm-950_P110-1999
ibm-954_P101-2000
ibm-971_P100-1995
ibm-eucKR
IBM-Thai
IMAP-mailbox-name
ISO_2022,locale=ja,version=0
ISO_2022,locale=ja,version=1 (JIS)
ISO_2022,locale=ja,version=2
ISO_2022,locale=ja,version=3 (JIS7)
ISO_2022,locale=ja,version=4 (JIS8)
ISO_2022,locale=ko,version=0
ISO_2022,locale=ko,version=1
ISO_2022,locale=zh,version=0
ISO_2022,locale=zh,version=1
ISO_2022,locale=zh,version=2
JIS (deprecated)
KOI8-R
KOI8-U
Latin-9
Latin1
Latin1 (deprecated)
Latin2
Latin3
Latin4
Latin5
Latin6
Latin8
LMBCS-1
macintosh
MS_Kanji
Native
SCSU
Shift_JIS
shift_jis78
SJIS (deprecated)
Thai8
UNICODE Big Endian (deprecated)
UNICODE Little Endian (deprecated)
US-ASCII
UTF-16
UTF-16 BigEndian
UTF-16 LittleEndian
UTF-16 OppositeEndian
UTF-16 PlatformEndian
UTF-16,version=1
UTF-16,version=2
UTF-16BE,version=1
UTF-16LE,version=1
UTF-32
UTF-32 BigEndian
UTF-32 LittleEndian
UTF-32 OppositeEndian
UTF-32 PlatformEndian
UTF-7
UTF-8
UTF-8 (deprecated)
Windows 874
Windows 949 (korean)
x-iscii-be
x-iscii-de
x-iscii-gu
x-iscii-ka
x-iscii-ma
x-iscii-or
x-iscii-pa
x-iscii-ta
x-iscii-te
x-mac-ce
x-mac-cyrillic

NONE and Zero

When the Item Subclass has a Number value and the Interpret as property has a Character value, the NONE and Zero properties are available. Expand the NONE or Zero property to specify an override data value to define the Special Value and Required on input properties.

- [**NONE > Special Value and Zero > Special Value**](#)
Enter a value in the Special Value property if the item is NONE or Zero.
 - [**NONE > Required on input and Zero > Required on input**](#)
Select a value from the Required on input property list.
-

NONE > Special Value and Zero > Special Value

Enter a value in the Special Value property if the item is NONE or Zero.

For example, if you specify * in the Special Value property for NONE, and a number object has the value *, it is interpreted as NONE.

If you enter a value in the Special Value property for NONE or Zero, enter the exact characters to be validated in the input data and built in the output data.

The maximum element length that is allowed for the Special Value property is 3000 bytes.

NONE > Required on input and Zero > Required on input

Select a value from the Required on input property list.

Yes

If the data object is NONE or Zero, the map uses the value in the Special Value property when it validates the item in the input.

No

Do not use the value in the Special Value property. No special values or input requirements are defined when data is NONE or Zero.

When the map builds the item in the output, the item might be either the value in the Special Value property or the default value for NONE or Zero. For example, if an item contains all pad characters and no actual data, the default value for NONE is used when that item is built in the output.

Places

The **Places** property allows you to specify the number of implied decimal places.

For item types defined with an **Item Subclass of Number**, a **Decimal** presentation, and a **Separator**, the **Places** property allows you to specify the minimum and maximum number of decimal places and whole number places. The minimum number of places must be less than or equal to the maximum number of places.

If the decimal number has no separator, use the **Places** property to specify the number of implied decimal places.

E-Notation

E-Notation gives you the ability to process and build decimal numbers expressed as E-Notation.

This is the common way to represent data that looks like the following:

"1.314E1" (value is 13.14)

or like this:

"-7.001e -2" (value is -.07001).

In E-Notation, a literal "E" is used to separate the coefficient from the exponent (coefficient E exponent).

Note: The data described in this documentation is "E-Notation data", and should not be confused with Scientific, or exponential notation.

- [**Data parsing expectations**](#)
 - [**Generating a data item defined with E-Notation**](#)
The following are expectations when generating a data item defined with E-Notation
 - [**E-Notation output**](#)
-

Data parsing expectations

- Data can be expressed as either "e" or "E", for example, "7.1e20" and "7.1E20 are both accepted.

- Positive exponents can have or omit the "+" sign, for example "1.314E+1" and "1.314E1" are both accepted.

To accept E-Notation from the input, toggle the E-Notation toggle on the Input Schema.

The following table shows examples of the input and the output:

Table 1.

Input	Output
1.234567890E9	1234567890
1.2E1	12
1.2E1	12
1.2E1	12
1.201E1	12.01
1.21E1	12.1
1.22345678991E10	12234567899.1
1.234E1	12.34
1.234E2	123.4
1.234E3	1234
1.2340E4	12340
1.23400E5	123400
-1.234E1	-12.34
-1.234E0	-1.234
-1.234E-1	-1.234
-1.234E-2	-0.1234
-1.234E-3	-0.01234
5.2E-4	.00052
0000.000054	.000054
0.052	.052
1.23	1.23
2222.356	2222.356

Generating a data item defined with E-Notation

The following are expectations when generating a data item defined with E-Notation

- Data is generated with the exponent as a capital "E" "1.314E1"
- Data is generated without spaces surrounding the capital "E" exponent indicator- "1.314E-4"
- Data is generated without signing the positive exponent values ("1.8E8" not "1.8E+8")

E-Notation output

On the Output Schema, toggle E-Notation to the active position.

An example of output data is shown in the table below:

Table 1.

Input	Output
12	1.2E1
12.00	1.2E1
12.000000	1.2E1
12.01	1.201E1
12.10	1.21E1
12234567899.10	1.22345678991E10
12.34	1.234E1
-12.00	-1.2E1
-12.34	-1.234E1
12	1.2E1
12.00	1.2E1
12.34E	1.234E1
-12.00	-1.2E1
-12.34	-1.234E1
12E	1.2E1

Text item subclass properties

Text item subclass properties can be interpreted with a **Character** or **Binary** value. Depending on the value, character or binary, the following are text item subclass properties:

- ["Interpret as Binary"](#) or ["Interpret as Character"](#)
- ["Size \(content\)"](#)
- ["Pad"](#)
- ["Restrictions"](#)
- ["National language"](#)

- [Interpret as binary](#)
- [Interpret as character](#)
- [Size \(content\)](#)
- [Pad](#)
- [Restrictions](#)
- [National language](#)
- [NONE](#)
- [Bidirectional](#)

Interpret as binary

Binary text items have content size and pad properties. Binary data is required to be sized or of a fixed size.

Binary number items interpret the data as a binary number or as a byte stream.

Items with an item subclass of **Text** can be interpreted as character or binary.

When **Interpret as** is set to **Binary**, the binary presentation options are Integer, Float, Packed, or BCD.

- [Binary integer presentation](#)
- [Binary float presentation](#)
- [Binary packed presentation](#)
- [Binary BCD presentation](#)

Interpret as character

Character number items interpret the data as symbolic data. Symbolic data has the same meaning on different computers. For example, a comma is symbolic because it has the same meaning, regardless of the computer type.

Character text items can have content size and pad properties.

When the **Interpret as** value is defined as **Character**, the **Release** property is specific to the **Character** value and is *not* available as an option for a **Binary** value. For more information, see ["Release characters"](#).

When **Interpret as** is set to **Character**, integer, decimal, and zoned presentation options are available.

- [Character integer presentation](#)
- [Character decimal presentation](#)
- [Zoned character presentation](#)
- [Places > implied](#)

Size (content)

Each text or syntax object can be defined with the minimum and maximum sizes in bytes and/or characters. The product determines size for text or syntax objects according to the following rules.

- When the size of an object is in bytes only, size and validation are calculated according to the size limitations set for bytes.
- When the size of an object is in characters only, size and validation are calculated according to the size limitations set for characters.
- When an object has size values defined for both bytes and characters, the object size is determined by characters (based on any character size limitations). Bytes are used to verify that the size of the data did not exceed any byte constraints that were set.

When the content size of a text or syntax item is specified in bytes, initiators, terminators, release characters, and pad characters are excluded from the count.

Min

The minimum size of the data object. You can enter values for bytes and/or characters.
The default value is 0. The largest value allowed is 65535.

Max

The maximum size of the data object. You can enter values for bytes and/or characters.
The default value is 0. The largest value allowed is 65535.
When there is no maximum content size limitation, this value is not required.

During schema analysis, checks are performed on byte and/or character Min and Max values according to the following methods:

- When values are present for both MinBytes and MinCharacters, the MinBytes value must be greater than or equal to MinCharacters.
- When values are present for both MaxBytes and MaxCharacters, MaxBytes must be greater than or equal to MaxCharacters.
- When no size limitations are present, no verification takes place.

Remember:

- Some character sets use two or more bytes per character.
- The trace file and audit log lengths are reported in bytes.

Pad

Pad implementation determines how the pad value is sized during validation and how the pad value is sized when written as output.

When the data value to be mapped to the target item is smaller than the minimum length of that item, pad characters can be used to pad the data to that minimum length. Input data can contain both content and pad characters. Output data is built according to the pad definitions of the types. Bytes are the default measurement for sizing, however, both byte and character sizing options are available.

Yes

Enables the **Pad** option. Allows the item to contain both content and pad characters.

No

All data is assumed to be content on input; no pad characters are built on output.

- [Pad > value](#)
- [Pad > Padded to](#)
- [Padded to > length](#)
- [Padded to > SizedAs](#)
- [Padded to > CountsTowardMinContent](#)
- [Padded to > Allow Excess Trailing Pads](#)
- [Pad > justify](#)
- [Pad > apply.pad](#)
- [Pad > fill](#)

Restrictions

Restrictions of an item are the valid values of that item. When the **Interpret as** value is defined as **Character**, the **Restrictions** property is specific to the **Character** value and is not available as an option for a **Binary** value.

Depending on other **Item Subclass** settings, the **Restrictions** property can be set to **Value**, **Character**, or **Range**.

- [Restrictions > ignore case](#)
- [Restrictions > rule](#)

National language

The National language default value is Western. For initiator, terminator, and release character **Literal** values, you can optionally specify a ["Data language"](#).

When the **Interpret as** value is defined as **Character**, the **National language** property is specific to the **Character** value and is not available as an option for a **Binary** value.

- [National language > data language](#)
- [Supported code pages](#)

NONE

When the Item Subclass has a Text value and the Interpret as property has a Character or Binary value, the NONE property is available. Expand the NONE property to specify an override data value to define the Special Value and Required on input properties.

- [NONE > Special Value](#)
Enter a value in the Special Value property if the item is NONE.
- [NONE > Required on input](#)
Select a value from the Required on input property list.

NONE > Special Value

Enter a value in the Special Value property if the item is NONE.

For example, if you specify * in the Special Value property for NONE, and a text object has the value *, it is interpreted as NONE.

If you enter a value in the Special Value property for NONE, enter the exact characters to be validated in the input data and built in the output data.

The maximum element length that is allowed for the Special Value property is 3000 bytes.

NONE > Required on input

Select a value from the Required on input property list.

Yes

If the data object is NONE, the map uses the value in the Special Value property when it validates the item in the input.

No

Do not use the value in the Special Value property. No special values or input requirements are defined when data is NONE.

When the map builds the item in the output, the item might be either the value in the Special Value property, or the default value for NONE. For example, if an item contains all pad characters and no actual data, the default value for NONE is used when that item is built in the output.

Bidirectional

Use the Item subclass > Bidirectional property to define the data as bidirectional.

Certain languages that are read from right-to-left often contain numeric or other phrases that are read from left-to-right. This type of data is called "bidirectional" because it changes direction in the middle of a line.

The default setting is **No**.

To enable this setting, select **Yes**. When you enable this setting, additional options are available to further define the bidirectional data.

Text items

The Bidirectional subproperties for text item types provide options for symmetric swapping, ordering, orientation, and shaping.

Symmetric Swapping

Use this property to maintain the orientation of directional pairs of characters (such as parentheses, greater than/less than symbols, brackets, braces).

Ordering Scheme

This property pertains to the order of the data as stored in memory.

Logical, the default setting, indicates that the data is stored in memory from left-to-right, regardless of the orientation of the text. Using the **Visual** ordering scheme, the data is read and stored with the start character in the right-most position both visually and in memory.

Orientation

Orientation pertains to the direction of the text as read visually. The default direction is left-to-right (LTR), meaning the text object is read starting from the left. The direction can be right-to-left (RTL), meaning the object is read starting from the right. When you select RTL, the behavior of some text functions (such as LEFT, RIGHT, MID) can change.

Use the Contextual LTR and Contextual RTL settings when the orientation should be taken from the context of the data because the data contains "strong" characters that are either orientation left or orientation right. When no strong characters are present in the data, the orientation will be based on this setting.

For example, when you set the orientation to Contextual LTR and no strong characters are encountered (meaning the data is orientation-neutral), the data orientation will be considered left-to-right.

Text Shaping

Use this property to produce output in a different glyph (shape or bit pattern). The default setting is off, or **Unshaped**.

To enable text shaping, select **Shaped**. The output will be shaped according to the code page of the target object.

Character number items

For character number item types, you can additionally define a numeric shaping characteristic.

Numeric Shaping

Use this property to indicate whether the output of numeric values will have the shapes that are used in English (Arabic digits 0123456789) or the national numerical shapes.

The default setting is **Regular**, which means Arabic digits (0123456789). Otherwise, select **Arabic-Indic** (۹۸۷۶۵۴۳۲۱۰).

The Bidirectional property is not applicable to binary numbers.

Date & Time item subclass properties

The Date & Time properties provide the flexibility to define multiple combinations of date-time formats.

Date & Time items can be interpreted as either binary or character. (See sections "[Interpret as Binary](#)" and "[Interpret as Character](#)".)

For binary **Date & Time** items, the **Presentation** property has two different values: **Packed** and **BCD**.

Packed

The **Packed** value is a number that has size, decimal places, and sign properties.

BCD

The **BCD** value is a binary coded decimal number that has size.

Other **Item Subclass** properties for **Date & Time** are displayed depending on the **Interpret as** setting (**Binary** or **Character**).

- [Date](#)
- [Time](#)
- [Format](#)
- [Time zones](#)
- [Time zone format string for XML](#)

- [Optional time segments of the time format string](#)
 - [Date and time format examples](#)
 - [NONE and Zero](#)
-

Date

For binary **Date & Time** items, use the **Date** property to define whether the date format is enabled.

Yes

Date format is enabled for this type. Expand the **Format** property to select the date format.

No

Date is not defined for this type.

- [Date > format](#)
-

Date > format

Select the date format that the data interpretation will be based on. For binary **Date & Time** items, the Date->Format property has four values.

CCYYDDD and **YYDDD** Julian date formats are supported.

- CCYYMMDD
 - YYMMDD
 - CCYYDDD
 - YYDDD
-

Time

For binary **Date & Time** items, use the **Time** property to define whether the time format is enabled. Select one of the following options:

Yes

Time format is enabled for this type. Expand the **Format** property to select the time format.

No

Time is not defined for this type.

- [Time > format](#)
-

Time > format

Select the time format that the data interpretation will be based on. For binary **Date & Time** items, the Time->Format property has several values. The following choices are available from the drop-down list:

- HH24MMSS
 - HH24MM
 - HH24:MM:SS
 - HH24:MM
 - Custom
-

Format

You can define a date-time format from within the type properties, except for native schema XML.

The output format for the native schema XML date/time field is always:

{CCYY-MM-DD}T{HH24:MM:SS.3-3+/-ZZ:ZZ}

To specify a different output format, use Xerces XML instead.

To define the Date & Time format:

1. Open the type properties.
 2. In the Item Subclass->Format property, click the browse button.
The Date Time dialog box is displayed.
 3. Make your format selections and click OK.
You can use alphabetical characters as separators. In the example,
2001-04-02T10:32:59-0500, T is the separator.
- Separators are limited to 60 characters.

Supported date formats:

- CCYYMMDD
- YYMMDD
- MMDDCCYY
- MMDDYY
- CCYYDDD
- YYDDD
- DDMMCCYY
- DDMMYY

Note: The DDMMCCYY and DDMMYY formats did not exist prior to version 6.5 of the Design Studio. If you had defined either one of these formats prior to 6.5 as a custom format, it will be recognized in 6.5 as the respective new format, instead of the previously defined custom format.

- [Use of format elements](#)
- [Custom date format](#)
- [Custom Time Format](#)

Use of format elements

If you select the same format element more than once for the same item, that item may not be validated separately. If you specify **MON** twice, the day of the month is not validated separately for both months. For example, if the date format element **MON** is used twice for a single item, with the following format string:

MON D-MON D

Suppose the data is this:

Feb 28-Mar 31

28 is not validated for the month of February.

Some combinations of reserved words are invalid. A separator must follow reserved words representing a variable number of digits (**D** and **M** for date) if data follows.

For example:

- D/M/CCYY is valid.
- CCYYM/D is invalid.
- DMCCYY is invalid.

See the Design Studio Introduction for a list of reserve words and symbols.

Custom date format

After choosing **Custom** from the date format drop-down list, click browse. The Date Format dialog box is displayed.

You can only use non-alphanumeric characters (excluding the {, } and [,] non-alphanumeric characters) as separators in the custom Date Format dialog box.

The following table provides examples of date formats.

Date Format	Description	Example
CCYY	4-digit Century + Year	1999
YY	2-digit Year (00-99)	99
MM	2-digit Month (01-12)	12
M	1- or 2-digit Month (1-12)	8
MON	3-character Month (Jan to Dec)	JAN
MONTH	Full name of Month	January
DDD	3-digit Day of year (001-366)	32
DD	2-digit Day of Month (01-31)	31
D	1- or 2-digit Day of Month (1-31)	7
DY	3-character Day of Week (Sun-Sat)	Fri
DAY	Full Name of Day of Week (Sunday-Saturday)	Friday
WW	1- or 2-digit Week of Year	13
Qn	Quarter of Year (Q1-Q4)	Q2
Custom	User defined custom date format	

After you define and save a custom format, the custom format string is displayed in the Properties window. For example, the custom date format **CCYYMMDD**, and the custom time format **HH24MMSS** display as:

{CCYYMMDD}{HH24MMSS}

Custom Time Format

After selecting **Custom** from the time format list, click browse. The Time Format window is displayed.

You can only use non-alphanumeric characters as separators, excluding the {, } and [,] non-alphanumeric characters, in the Time Format window.

The following table provides time format examples.

Time Format	Description	Example
HH24	2-digit hour in 24 hour format (00-23)	23
H24	1- or 2-digit hour in 24 hour format (0-23)	11
HH12	2-digit hour in 12 hour format (00-12)	08
H12	1- or 2-digit hour in 12 hour format (0-12)	8
MM	2-digit minute (00-59)	09
M	1- or 2-digit minute (0-59)	9
SS	2-digit second (00-59)	05
S	1- or 2-digit second (0-59)	5
AM/PM	Meridian (AM/PM)	AM
ZZZ	A time zone abbreviation. See " Time Zones " for a list of supported time zones.	EST
+/-ZZZZ	Hours and minutes before or after the Greenwich Mean Time (GMT). GMT is now referred to as Coordinated Universal Time (UTC).	+0500
+/-ZZ:ZZ	4-digit time where the format is a 2-digit hour and 2-digit minute, separated by a colon.	+05:00
+/-ZZ[;ZZ]	4-digit time where the format is a 2-digit hour and an optional 2-digit minute, separated by colon.	+05:00
+/-ZZ[ZZ]	4-digit time where the format is a 2-digit hour and an optional 2-digit minute.	+0500
TZD	4-digit time where the format is a 2-digit hour and 2-digit minute, separated by a colon. Z on output, if value is +/-00:00.	+05:00

After you define and save a custom format, the custom format string is displayed in the Properties window. For example, the custom date format CCYYMMDD, and the custom time format HH24MMSS display as:

{CCYYMMDD}{HH24MMSS}

Time zones

The following table lists the supported time zones.

Time Zone	Abbreviation	Hours before Greenwich Mean Time	Hours ahead of Greenwich Mean Time
Greenwich Mean Time (Zulu)	GMT	0	0
West African Time	WAT	-1	+23
Azores Time	AT	-2	+22
No name; Brasilia Time	###	-3	+21
Atlantic Standard Time	AST	-4	+20
Eastern Standard Time	EST	-5	+19
Central Standard Time	CST	-6	+18
Mountain Pacific Time	MST	-7	+17
Pacific Standard Time	PST	-8	+16
Yukon Standard Time	YST	-9	+15
Hawaii Standard Time	HST	-10	+14
Nome Time	NT	-11	+13
New Zealand Time	NZT	-12	+12
No name; no location	###	-13	+11
Guam Standard Time	GST	-14	+10
Japan Standard Time	JST	-15	+9
China Coast Time	CCT	-16	+8
West Australia Time	WAT	-17	+7
Zulu + 6 (Russia zone 6)	ZP6	-18	+6
Zulu + 5 (Russia zone 5)	ZP5	-19	+5
Zulu + 4 (Russia zone 4)	ZP4	-20	+4
Baghdad Time	BT	-21	+3
Eastern European Time	EET	-22	+2
Central European Time	CET	-23	+1

Time zone format string for XML

Note: The output format for the native schema XML date/time field is always:

{CCYY-MM-DD}T{HH24:MM:SS.3-3+/-ZZ:ZZ}

To specify a different output format, use Xerces XML.

The following time zone strings support the specification of a single character Z to indicate Coordinated Universal Time (UTC), as described by the World Wide Web Consortium (www.w3.org) and the ISO 8601 standard:

- +/-ZZZZ
- +/-ZZ:ZZ
- TZD

When the +/-ZZ:ZZ or +/-ZZZZ format string is specified, the data validation process works in the following manner:

- Valid data that corresponds to this format string will contain either a character literal Z (representing UTC) or a zone in the appropriate format: +/-ZZ:ZZ or +/-ZZZZ, as specified.

- If a character literal Z is present, it shall be interpreted, and treated at mapping time, as UTC, which is equivalent to +00:00 or +0000.

When the TZD format string is specified, the data validation and output generation processes work in the following manner:

- Valid data that corresponds to this format string will contain either a character literal Z, representing UTC, or a zone, in the +/-ZZ:ZZ format.
- If a character literal Z is present, it shall be interpreted, and treated at mapping time, as UTC, which is equivalent to +00:00.
- If the value is +/-00:00, it generates Z in the output. Otherwise, it generates the output in +/-ZZ:ZZ format.

The following time zone strings support the omission of the minute portion of the difference from UTC:

- +/-ZZ[:ZZ]
- +/-ZZ[ZZ]

Optional time segments of the time format string

This section discusses how you can specify a portion of a time-format string to be optional. For example, you can specify that either the *time zone* portion or the *hours, minutes, seconds, fractional seconds, Meridian* portion is optional.

This flexibility is also applicable to time-format strings used in functions or in schema scripts imported by the Type Tree Maker.

Only the time portion **or** the zone portion can be optional—not both.

To specify a portion of the time-format string as optional:

1. The following procedure assumes that the **Item Subclass** property of your schema is **Date & Time**.
2. From within the schema properties, navigate to Item Subclass.>.Format.
3. Click the browse button to open the Date Time dialog.
4. From a Format field that is set to **Time**, go to the field directly below it and choose Custom from the drop-down menu.
5. Next to the Format field with **Custom** selected, click the browse button. The Time Format dialog is displayed.
6. Choose from the following tasks:
 - To specify the *hours, minutes, seconds, fractional seconds, and Meridian*-portion of the time format string as optional, enable the check box on the far left side of the dialog.
 - To specify the *time zone* portion of the time format string as optional, enable the check box.
 Only one segment of the time format string can be specified as optional.

Date and time format examples

The following list includes examples of popular standard date and time formats:

Format	Description
X12 EDI	Fractional seconds format: HH24MM[SS[0-2]] .

SWIFT

Time Zone Designator or UTC Designator format: **HH24MSS[TZD]**.

TimeStamp format: **CCYY-MM-DDTHH:MM[SS[.0-6]][TZD]**, which is a combination of date and time.

HL7

Time format: **HH24MM[SS[.0-6]][+/-ZZZZ]**.

TimeStamp format: **CCYYMMDDHHMM[SS[.0-6]][+/-ZZZZ]**, which is a combination of date and time.

SAP

Time format: **HHMMSS**.

ODBC

Time format: **HH:MM:SS[.0-9]**, which is the form most commonly used in **ODBC** mapping. The fractional part is optional and not often used.

NONE and Zero

When the Item Subclass has a Date & Time value and the Interpret as property has a Character value, the NONE and Zero properties are available. Expand the NONE or Zero property to specify an override data value to define the Special Value and Required on input properties.

- [NONE > Special Value and Zero > Special Value](#)
Enter a value in the Special Value property if the item is NONE or Zero.
- [NONE > Required on input and Zero > Required on input](#)
Select a value from the Required on input property list.

NONE > Special Value and Zero > Special Value

Enter a value in the Special Value property if the item is NONE or Zero.

For example, if you specify * in the Special Value property for NONE, and a data and time object has the value *, it is interpreted as NONE.

If you enter a value in the Special Value property for NONE or Zero, enter the exact characters to be validated in the input data and built in the output data.

The maximum element length that is allowed for the Special Value property is 3000 bytes.

NONE > Required on input and Zero > Required on input

Select a value from the Required on input property list.

Yes

If the data object is NONE or Zero, the map uses the value in the Special Value property when it validates the item in the input.

No

Do not use the value in the Special Value property. No special values or input requirements are defined when data is NONE or Zero.

When the map builds the item in the output, the item might be either the value in the Special Value property or the default value for NONE or Zero. For example, if an item contains all pad characters and no actual data, the default value for NONE is used when that item is built in the output.

Syntax item subclass properties

Syntax objects are characters that precede, separate, or follow a particular data object. Item types with an **Item Subclass** of **Syntax** are defined and used to specify delimiters, initiators, terminators, and release characters. Syntax objects with variable values are defined as the syntax object's **Variable** property. Syntax objects appearing as actual data to be mapped as output data are defined as components of a type.

- [Syntax objects with variable values](#)
- [Example of variable syntax object as an item type](#)
- [Syntax objects as data](#)
- [Example of syntax objects as components of a type](#)
- [Delimiter > find](#)

Syntax objects with variable values

Using a syntax item to specify delimiters, initiators, terminators, and release characters is required when the value of syntax objects (delimiters, initiators, terminators, and release characters) may vary.

The restrictions of syntax items define the variable literal values.

To define a variable syntax object:

1. Create an item type with an **Item Subclass** of **Syntax** to represent the syntax object.
2. Define the restrictions for the syntax item type.
3. For the item or group type with the variable syntax object, for the **Variable** delimiter, initiator, terminator, and release character, select the item from the **Item** drop-down list in the Properties view.
4. For the **Find** property, select Yes.

Only item types defined with an **Item Subclass** of **Syntax** appear in the drop-down list.

Example of variable syntax object as an item type

In this example, the group type **Header** has variable delimiter values of: , * + ~

1. Create an item type with the **Item Subclass** of **Syntax** and the name **Delimiter**. (The name can be any valid type name, but naming this item **Delimiter** is useful).
2. To define the item restrictions, double-click the **Delimiter** type. The item view is displayed.
3. Define the restrictions in the **Include** column.
4. In the Properties window, select Syntax for the **Item Subclass** property.
5. Open the properties for the delimited group type **Header**.
6. For the **Syntax** property, choose **Delimited**.
7. For the **Delimiter** property, choose **Variable**.
8. Expand the **Delimiter** property.
9. Define the Delimiter:
Default literal value.
10. For the variable Delimiter Item property, choose the item type **Delimiter** from the drop-down list.
11. Indicate that the variable delimiter should be determined for each occurrence of the object by selecting Yes for the Delimiter:
Find property.

Syntax objects as data

The value of a syntax object (delimiters, initiators, terminators, and release characters) may appear as actual data that can be mapped. Syntax objects that appear as actual data are defined as components of a type.

Example of syntax objects as components of a type

The following is an example of defining a syntax object as a component of a group type. The following data represents a message received from multiple departments.

Each message is made up of segments. Each department uses different segment delimiters and terminators. The message format specifies that the delimiter and terminator are the first two bytes of the message, followed by the actual data.

To define syntax objects for the delimiter and terminator:

1. Define two separate syntax objects with an **Item Subclass** of **Syntax**, one for the message delimiter and one for the message terminator. Appropriate type names might be **SegmentDelimiter** and **SegmentTerminator**.
2. Define the possible message delimiter values as restrictions of the **SegmentDelimiter** item type. Define the possible message terminator values as restrictions of the **SegmentTerminator** item type.
3. Define these item types as the first two components of the group type **Message**.
4. Expand the **Delimiter** property.
5. For the variable **Delimiter.>Item** property, select SegmentDelimiter from the drop-down list.
6. For the variable **Terminator.>Item** property, select SegmentTerminator from the drop-down list.

During the data validation process, the component **SegmentDelimiter** appears in the data with a value of *. The group type **Segment** has a variable delimiter specified as the item type **SegmentDelimiter**, with the value *. Therefore, it is understood that the segment has * as the delimiter.

When the value of a syntax object appears as actual data, it can be mapped as data. For example, source data may use different delimiters from several different sources. Acknowledgments of this data must be sent back to the source using the delimiter used in the original data. This data can be mapped from the input to the output if these syntax objects are defined as components within the data.

Delimiter > find

When syntax objects are **Variable**, the **Find** property must be defined. The **Find** property determines whether the value of syntax object is set on each occurrence or whether the current setting (or default value) is used.

When the **Initiator**, **Terminator**, **Release**, or **Delimiter** property value is **Variable**, select one of the following from the drop-down list in the **Value** column:

Yes

Determine the value of the syntax object each time an occurrence of that type is found. After the value of that syntax object is found, that particular value is used until it is reset by another **Find** or by the occurrence of that syntax item as a component.

INPUT: The value of the object in the input data is determined by the location in the data stream and the restrictions of the syntax item.

OUTPUT: The default value is used for building the syntax object in the output data.

No

The variable syntax object is defined as the current value or, if it is not set, as the default value.

INPUT: If the syntax object is not encountered in the data stream, the value of the syntax object is the default value.

OUTPUT: If the value of the syntax item has *not* been previously set, the default value is used.

Group properties

Group properties include the group's **Subclass** and **Format**, which describes how to distinguish one component of that group from another component of that group.

- [Group subclass](#)
- [Sequence group formats](#)
- [Explicit format](#)
- [Implicit format](#)
- [Specifying a delimiter](#)
- [Defining exclude characters](#)

During map run time, the data validation process uses the Exclude Character List group property to define a set of characters that are not allowed to exist in the data stream as data content.

- [Escaping an excluded character in the data content](#)

Select Release exclude characters when you configure the exclude character list to enable the release character to escape excluded characters. When enabled, a release character that precedes an excluded character in data content escapes the excluded character, and the data content passes validation. Likewise, Link precedes an excluded character with the release character when building the output stream.

Group subclass

Group types have a subclass of **Sequence**, **Choice**, or **Unordered**.

Property	Description
Sequence	A partially-ordered or sequenced group of data objects. Each component of a Sequence group is validated sequentially.
Choice	Choice groups provide the ability to define a selection from a set of components like a multiple-choice question on a test. A Choice group is valid when the data matches one of the components of the choice group. Validation of a Choice group is attempted in the order of the components until a single component is validated. If the Choice group has an initiator, the initiator is validated first.

Property	Description
Unordered	An unordered group has one or more components. Unordered groups can only have an Implicit format property, with the same syntax options as sequence groups: None or Delimited .

- [Properties of group subclasses](#)
- [Choice group components](#)
- [Unordered group components](#)

Properties of group subclasses

The distinction between Sequence and Choice group subclasses is that Choice groups have no **Partition** or **Format** properties.

Type syntax properties such as **Initiator**, **Terminator**, **Release**, and **Empty** can be applied to **Choice** groups. If a release character is defined, by default it applies to the **Terminator**.

Choice group components

A Choice group can have both items and groups as components. A partitioned Sequence group can only have group subtypes.

Components of a Choice group must be distinguishable from each other. The components of a Choice group cannot have a component range other than (1:1). Only one component of a Choice group built in the output data.

A Choice group data type is only one of the group components. For example, the data type **Record** is a group type with a **Group Subclass of Choice**. The group type **Record** has three components: **Order**, **Invoice**, and **Sales**. The data validation of **Record** will be only one of the components: **Order**, **Invoice**, or **Sales**. For this reason, the components of a Choice group must have a component range of (1:1).

Unordered group components

An unordered group has one or more components that can appear in the data stream in any order. They allow many SWIFT and FIX message types, for example, to be defined in a more natural way.

Unordered groups have no partitioned property. They have **Implicit** format properties with the same syntax options as sequence groups: **None** and **Delimited**.

When a group is defined as **Unordered**, any component can appear in the data stream. A component can be an item or a group.

Unordered group components have a range property. For example, if the unordered group, **A**, has the following component list:

```
B (1 : S)
C
D (S)
```

then **A** must have one **C**, at least one **B**, and possibly some **Ds**. They could appear in any order. For example, data for **A** could have the pattern: **CDBBDDD** or **BBBDDDCBD**.

Component rules of an unordered group cannot reference other components of the same group. They can only reference the component to which the rule refers and the objects contained in that component.

Sequence group formats

A sequence group has either an **Explicit** or **Implicit** format.

Explicit

The explicit format relies on syntax to separate components. Each component can be identified by its position or by a delimiter in the data. Delimiters appear for missing components.

Implicit

The implicit format relies on the properties of the component types. The format is not fixed. If delimiters separate components, they do not appear for missing components.

For example, if each component of a fixed group has a fixed size, the component is distinguished from the next component by its position in the data. Or, a group may have delimiters that appear for missing components. In these cases, the format is apparent; the group has an explicit format.

If a group does not have an explicit format, it has an implicit format. An implicit format relies on the properties of the component types. In this example, the components make some pattern in the data and it is possible to distinguish between them, but the format is not fixed and if delimiters separate components, they do not appear for missing components.

When deciding what format a group has, it may help to ask first whether it is clear where one component ends and another begins. Generally, a group has an explicit format if the position of each component in the data stream is always the same or if a delimiter always marks the place for each component.

Explicit format

To specify that a group has an explicit format, choose **Explicit** for the **Format** property. Select a setting for the **Track** property, and choose the group's syntax: **Fixed** or **Delimited**.

- [Track](#)
 - [Fixed syntax](#)
 - [Explicit delimited syntax](#)
-

Track

The **Track** property indicates whether the system should track only the components that have content or all components, including those that do not have content. The settings are **Content** and **Places**.

For example, if a group **StudyGroup** has the component **Name(s)**, and Places is specified for the Track property, any empty occurrences of **Name** are tracked.

Suppose **Name** has an * initiator and a space pad character. This is the data for **StudyGroup**:

*Carolyn * *Stuart *Margaret

Notice that the second **Name** is missing.

If Places is specified for the Track property, and you want to map the third **Name**, the empty **Name** would be counted as an occurrence, so the third **Name** would be Stuart. However, if Content is specified for the Track property, the empty **Name** would not be counted as an occurrence because it does not have content and the third **Name** would be Margaret.

Fixed syntax

If a group data object is always the same size (in bytes), it has a fixed syntax. For example, a record that is always 160 bytes has a fixed syntax.

Each component of a fixed group must be fixed. If you break down a fixed group, it ultimately consists of items that are fixed. Each is padded to a fixed size or its minimum and maximum content size are equal. Do not specify the size of a fixed group. The size is automatically calculated based on the size of the group's components.

- [Guidelines for defining a fixed group](#)
-

Guidelines for defining a fixed group

- Each component must be a group with a fixed syntax or a fixed item. The item is padded to a fixed size or its minimum and maximum content size are equal.
 - Each component must have a specified range maximum. The maximum cannot be s.
 - If a component range minimum is not equal to the maximum, content is not required for optional component occurrences. For example, if a component is the item **ShippingAddress** (0:1) and **ShippingAddress** has a minimum content size of two characters, data may either contain: 1) all pad characters, or 2) if content is in the data stream, there must be a minimum of two characters for a **ShippingAddress**. If a component is a group, no content is required for any of the items contained in an optional occurrence of that component.
-

Explicit delimited syntax

An explicit-format group with a delimited syntax is one whose components are separated by a delimiter and the delimiter appears as a placeholder even when a component has no content.

About delimiters and explicit-format groups:

- The only time a delimiter can be missing is when all components following the delimiter are optional and there is no data for the optional components.
 - A delimiter is a character or series of characters that separates data objects.
 - A delimiter cannot be longer than 500 bytes.
 - The delimiter of a group appears inside the group, separating its components. When a group is delimited, that indicates something about the components of the group. The delimiter inside a group is delimiting the components.
-

Example

The group **Employee** has an explicit-delimited format because a comma delimiter appears between **Employee's** components, the items that make up the **Employee** object. In addition, the delimiter is displayed when a component is missing and there is data for components following it.

The components of **Employee** are the items **ID#**, **Name**, **Department**, **Address**, and **Age**.

- [Delimiter](#)
-

Delimiter

Use the **Delimiter** property to specify the value and location of the delimiter. For specific information on specifying the delimiter, see "["Specifying a Delimiter"](#)".

In an explicit-delimited group, if the delimiter is whitespace <WSP>, the delimiter is interpreted as one byte long; each <WSP> character is interpreted as another delimiter. In the output, a <WSP> delimiter is one space.

The limit for <WSP> is 512 bytes.

Implicit format

In a group with an implicit format, the components are distinguishable not by delimiter or position, but by their pattern; something in the definition of the component types themselves. The group has no syntax property that distinguishes one component from another.

For example, the type **File** consists of **Record**(s). <CR><LF> appears at the end of each **Record**. It has been defined as the terminator of **Record**. **File**, however, has no syntax of its own. The **Record** terminator distinguishes one **Record** from another.

To specify that a group has an implicit format, choose **Implicit** for the **Format** property. Optionally, define a comment type, and choose the group's syntax: **Delimited** or **None**.

- [Floating component](#)
- [Implicit whitespace syntax](#)
- [Implicit delimited syntax](#)
- [No syntax](#)
- [Distinguishable components of an implicit group](#)

Floating component

The floating component represents an object that may appear after any component of the group.

An implicit group can have a floating component; an explicit group cannot. If the group is prefix or infix delimited, the floating component is displayed before the delimiter. If the group is postfix delimited, the floating component is displayed after the delimiter.

A floating component can be an optional component that may appear after any other component. However, it is not included in the component list because it does not appear at a specific location.

If a group has a floating component, a component must be distinguishable from a floating component. For example, components and floating component could start with different initiators.

Note: When a floating component appears in the input data, it is validated during mapping. If there are floating components in your output data, define them as actual components of the output.

A floating component can appear after the initiator (when the type has an initiator), after each component, or both. For EDI and other existing floating components, trees will be converted as "after each component".

A floating component can be specified for implicit sequence group, choice group, and unordered group definitions.

Implicit whitespace syntax

Select WhiteSpace for the **Component Syntax** value when white spaces are not allowed in the data. When you select the **WhiteSpace** option, define the **Build As** and **Character Set** properties.

Note: Helpful for XML data.

- [Build as](#)
- [Character set](#)

Build as

Enter the characters that will replace white spaces.

The Build As property is only available when the Component Syntax property is defined as WhiteSpace.

Character set

Select a character set for the component from the drop-down list.

The default value is Native, where the literal values use the native character set of the computer.

[Supported character sets \(code pages\)](#)

Implicit delimited syntax

If a delimiter separates the components of a group, but the delimiter does not appear when a component is missing, the group has an implicit format with a delimited syntax.

In certain data, the delimiter may not be a placeholder.

- [Delimiter](#)

Delimiter

Use the **Delimiter** property to specify the value and location of the delimiter. For specific information on specifying the delimiter, see "["Specifying a Delimiter"](#)".

In an implicit-delimited group, when the delimiter is whitespace <WSP>, all contiguous <WSP> characters are treated as one delimiter. In the output, a <WSP> delimiter is built as one space.

The limit for <WSP> is 512 bytes.

No syntax

To specify a group with an implicit format that has no syntax, choose **None** for the **Component Syntax** value.

Distinguishable components of an implicit group

Each component of an implicit group needs to be recognizable. If either of two different components appears at the same place in a data stream, there must be a distinguishable difference between one component and the other.

Sometimes components of an implicit group may be distinguished because there is something in the data that distinguishes them.

After the first item **Line** of the **Form**, the next data object could be another item **Line**. Or, it could be a **Trailer Line**. When looking at a particular **Line**, there is something in the data that identifies that **Line** either as a **Header Line**, an item **Line**, or a **Trailer Line**. The type **Line** has been partitioned to distinguish between the different kinds of **Lines**.

Specifying a delimiter

For the **Delimiter** property, specify whether the delimiter is a literal or a variable value.

- [Literal](#)
- [Variable](#)
- [Location](#)
- [Delimiter value appears as data](#)
- [Allow Excess Trailing Delimiters](#)

Literal

If the delimiter is a constant value, enter the delimiter value in the **Value** field. To enter a non-printable value, click the browse button and select a value from the Symbols dialog box.

- [National language](#)
- [Data language](#)

National language

The National language default value is Western. For initiator, terminator, and release character **Literal** values, you can optionally specify a "["Data language"](#)".

When the **Interpret as** value is defined as **Character**, the **National language** property is specific to the **Character** value and is not available as an option for a **Binary** value.

- [National language > data language](#)
- [Supported code pages](#)

Data language

Use the **National language** > **Data language** property to define the data language or character set.

[Supported code pages](#)

Variable

Sometimes you do not know what delimiter is used in the data source, especially if you receive that data from an outside source. However, you know all of the possible values that the delimiter could be. You can create an item to represent this delimiter and specify all of the possible values as restrictions of that item. The value of the delimiter in the data is found.

To specify an item as a Variable Delimiter:

1. In the Properties view for the delimited group, click in the Delimiter, Item value field.
2. Press Alt and drag the item from the schema editor into the **Item** field.

Location

The **Location** property specifies the location of the delimiter with respect to the components. The options are prefix (the delimiter appears before the component), postfix (the delimiter appears after the component), and infix (the delimiter appears between components).

The following table explains each option.

Property	Description	Example
Prefix	Before each component. Before each member of a component in a series.	$*a*a*b*c$
Postfix	After each component. After each member of a component in a series.	$a*a*b*c*$
Infix	Between components. Between members of a component in a series.	$a*a*b*c$

Delimiter value appears as data

Delimiters do not appear as part of the actual data. For example, if the delimiter is specified as a comma, the comma in the following text item would be considered a delimiter, rather than part of the data itself:

Tom Smith, Jr.

If the data does contain the delimiter value, there are ways to format the data so that both the data and the delimiter are distinguishable. For example, text items can be enclosed in quotation marks. Or, a release character may be used. For information about release characters, see ["Release Characters"](#).

Allow Excess Trailing Delimiters

The schema validation process allows data that contains excess trailing delimiters on sequence groups.

You can change the default setting for the Allow Excess Trailing Delimiters property to No so that the schema validation process, instead, fails data that contains excess trailing delimiters.

The schema validation process interprets a delimited sequence group that contains optional components at the end of its sequence, as having excess trailing delimiters.

For example, GROUPA contains the following components: COMPA (mandatory), COMPB (optional), COMPC (optional), and COMPD (optional). The group is infix-delimited by a tilde. The following data stream examples are valid for GROUPA:

A~B~C~D
A~B~C~
A~B~~
A~~~

For this example, you want the schema validation process to disallow all of the data stream examples that contain unnecessary or excess trailing delimiters. To disallow the excess trailing delimiters, you can change the setting for the Allow Excess Trailing Delimiters property to No. The result is that the validation process fails all of the examples except the first one.

Defining exclude characters

During map run time, the data validation process uses the Exclude Character List group property to define a set of characters that are not allowed to exist in the data stream as data content.

The data validation process allows these excluded characters to exist in the data stream as syntax, such as delimiters, if they are not being used as data content.

The group exclude characters might be derived from previously parsed values in the data stream or might be defined as literal values.

When you define exclude characters at the group level, the data validation process handles these specified characters as if they were in-scope data syntax.

For example, you have a schema which defines two groups, GROUPA and GROUPB. The tree defines the asterisk (*) character as a group delimiter for GROUPA. It defines the at-sign (@) character as a group delimiter for GROUPB. The tree also defines the FILE group, which contains both GROUPA records and GROUPB records.

The data stream contains content corresponding to the FILE group, which is composed of a series of both GROUPA and GROUPB records.

When a map begins to run, it parses and validates this data stream. Since the asterisk is the delimiter of GROUPA, the data content of GROUPA records is only allowed to contain asterisk characters that are escaped. If the data content contains non-escaped asterisk characters, those GROUPA records fail data validation. Similarly, since the at-sign is the delimiter of GROUPB, the data content of GROUPB records is not allowed to contain non-escaped at-sign characters. If it does, those GROUPB records fail data validation.

The schema defines the delimiters, or markup, for the GROUPA and GROUPB records. Asterisk characters are considered to be in-scope markup while the data validation process handles GROUPA records. At-sign characters are considered to be in-scope markup while the data validation process handles GROUPB records.

Conversely, GROUPA data content is allowed to contain at-sign characters, as the GROUPB at-sign delimiter does not apply to GROUPA records. Therefore, at-sign characters are considered out-of-scope markup for GROUPA records. GROUPB data content is allowed to contain asterisk characters, as the GROUPA asterisk delimiter does not apply to GROUPB records. Therefore, asterisk characters are considered out-of-scope markup for GROUPB records.

You might require the data validation process to exclude asterisk and at-sign characters in all of the data content for both GROUPA and GROUPB records, except when those characters are used as delimiters.

If so, use the Exclude Character List property to specify the asterisk and at-sign literal values at the FILE group level. When the validation process evaluates the data, it throws an exception if there is an asterisk (*) or an at-sign (@) character in the data content within the FILE group. The asterisk and at-sign characters are still valid if they are used as group delimiters; they are invalid if they are used in data content.

Performance tip: The Exclude Character List property causes the map validation process to use more resources as it searches for each of the exclude group characters in every data element under the group on which it is defined.

Open the Exclude Character List window by clicking the Value field on the Exclude Character List property, and then clicking the button that appears at the end of the field. Specify the characters in the list as literal characters or syntax items. You can use the Add Literal, Add Syntax Item, Edit, Remove, and Remove All functions to add literal characters and syntax items to the list, edit them, and remove them from the list.

Escaping an excluded character in the data content

Select Release exclude characters when you configure the exclude character list to enable the release character to escape excluded characters. When enabled, a release character that precedes an excluded character in data content escapes the excluded character, and the data content passes validation. Likewise, Link precedes an excluded character with the release character when building the output stream.

Related tasks

- [Adding literal](#)
- [Adding syntax type](#)

Related reference

- [Release characters](#)

XML properties in the schema

When you open a schema in the Type Designer after you imported it from a DTD or XML Schema, the XML-specific properties are displayed as read-only fields.

For each type in the resulting schema, the Intent property is either General (indicating non-XML) or XML. All category types that are created during the import process are considered non-XML properties. All group and item types that are created during the import process are considered XML properties.

Any types that you manually add to the XML type tree are considered non-XML and have General intent. In such a case where there are both XML and non-XML types present (such as EDI data), the appropriate method of validation is determined automatically.

See the Type Tree Maker for information about creating schemas by importing XML Schemas and DTDs.

See Utilities for XML for information about tools that can assist you with upgrading your XML schemas and maps to this version of IBM Transformation Extender.

- [Support for XML constructs](#)
- [XML schema datatypes](#)
- [Identity constraints](#)
- [Namespaces](#)
- [Element type declarations](#)
- [Element and attribute wildcards](#)

Support for XML constructs

IBM Transformation Extender validation supports the following XML constructs.

Character Data

During validation, character data is mapped by the XML parser to IBM Transformation Extender types. This data includes both parsed character data (PCDATA) and unparsed character data (CDATA).

Comments and Processor Instructions

XML comments and processing instructions (PI) are mapped to floating components in schemas.

Namespaces

The XML Schema importer supports the specification of arbitrary prefixes for namespaces declared in the input grammar.

XSDL Hints

The xsi:schemaLocation and xsi:noNamespaceSchemaLocation attributes are collectively known as XML Schema Definition Language (XSDL) hints. These attributes specify the location of the XML Schema(s) that the XML parser uses for validation.

An XML Schema allows either or both of these attributes to display within any element tag. But the XML parser respects the location values only when the XSDL hint is specified in the root element of the schema.

The Doc group type of schemas that are created with the XML Schema or XML DTD importer contains the location of the Xerces DTD or Xerces XML Schema that is used for Xerces XML validation. The Intent > Validate As > Location property can be modified. However, use the Schema > Type > Metadata card setting in the map to change the location of the DTD or Schema if needed.

Empty Elements

Only one set of initiators and terminators for the Empty element is required for validation.

Nillable Elements

A nillable element can have one of three states: absent, present with content, or present with nil content. Both the DTD and XML Schema allow the definition of optional elements in the instance document. Additionally, the XML Schema allows nillable elements where the content can be empty when it contains an xsi:nil attribute with a value of "true", despite the fact the element's content is mandatory.

The XML Schema Importer can create the content of a nillable element within a group with a range of (0:1), which makes the element content optional.

Mixed Content

When parsing elements with a mixed content model (containing both character data and child elements), the DTD and XML Schema importers generate text items within the mixed content (instead of character sequences).

Regular Expressions

An XML Schema allows the restriction of the values of simple types that are based on regular expressions or pattern facets. During XML validation, the parser enforces the pattern facet. The XML Schema Importer fills the appropriate type properties with pattern facets encountered in the input schema, which are viewable in the Type Designer.

XML schema datatypes

This section describes the schema constructs that correspond to XML Schema datatypes.

The following table lists some common XML Schema datatypes with their corresponding type property values in the Type Designer.

For detailed information about XML attributes and elements as type properties in the schema, refer to the Type Tree Importers documentation .

XML Schema Datatype	Type Property	Value
simpleType	Intent > Validate As	XML_SIMPLETYPE
complexType	Intent > Validate As	XML_COMPLEXTYPE
element	Intent > Validate As	XML_ELEMENT
attribute	Intent > Validate As	XML_ATTRIBUTE
group	Intent > Validate As	XML_GROUP

- [Simple types](#)
- [Complex types](#)
- [Elements](#)
- [Attributes](#)
- [Groups and substitution groups](#)

Simple types

Elements that have assigned simple types have character data content but not child elements or attributes.

Simple type elements are represented in the Type Designer type properties with the value XML_SIMPLETYPE.

Complex types

Elements that have assigned complex types can have both child elements and attributes. The order and structure of the child elements of a complex type are known as its content model. Content models are defined using a combination of model groups, element declarations or references, and wild cards. There are three kinds of model groups: **all**, **choice** and **sequence**.

Complex type elements, including the complex types using compositor elements, are represented in the type properties with the value XML_COMPLEXTYPE.

Elements

Element declarations are either local or global. Local elements are represented in the Type Designer type properties with the value: XML_ELEMENT.

When the schema defines several global elements, the generated schema includes a Global choice group that contains all of the global elements. A Global choice group is represented in the type properties with the value XML_BODY.

Attributes

Attributes are another building block of XML. The import process uses attribute declarations to name attributes and associate them to particular simple types. Attribute declarations can be either local or global.

Attributes, including both local and global, are represented in the type properties with the value XML_ATTRIBUTE.

Groups and substitution groups

One way an XML Schema allows the creation of reusable content is by using named model groups. These global groups can be referenced throughout a schema.

Substitution groups are a flexible way to designate element declarations as substitutes for other element declarations in a content model. New element declarations can easily be designated as substitutes from other schema documents or namespaces without changing the original content model.

Groups and substitution groups are represented in the type properties with the value XML_GROUP.

Identity constraints

The XML parser in IBM Transformation Extender enforces declared identity constraints. The XML Schema Importer sets these properties based on the XML Schema. In general, the identity constraint definition serves in one of the roles listed below:

Identity Constraint	Description
Unique	Enforces that a value (or combination of values) is unique within a given scope. For example, all product numbers must be unique within a catalogue.
Key	Enforces uniqueness and requires that all values be present. For example, every product must have a number and it must be unique within the catalogue.
Key References	Enforces that a value (or combination of values) corresponds to a value represented by a key or uniqueness constraint. For example, for every product number that is displayed as an item in a purchase order there must be a corresponding product number in the product description section.

Namespaces

The XML Schema Importer supports namespace and prefix properties for all elements and attributes. The importer assigns the values of these properties based on the input grammar. For each namespace in the input grammar, the importer allows you to specify the namespace prefix to be used for output documents.

During the import process, a **Location** value is specified in the schema without a fully qualified path. In this case, a map execution process would, by default, look for the XML Schema at this location. However, if you need to change this path, you can do so.

For example, if you refer to the proper location of the schema in the XML document, then you can manually remove the path location from the type properties. When the map execution process does not find a **Location** value specified in the schema, the process then looks to the "[XSDL Hints](#)" specified in the XML document. When XSDL hints are not present, validation fails.

The value specified in the schema takes precedence over the XML document.

Element type declarations

The XML DTD and XML Schema importers can generate a simple schema when encountering character content within mixed data. During the import process, when character data (CDATA/PCDATA) is encountered, the importer incorporates it into existing text items within the schema.

Element and attribute wildcards

DTDs and XML Schemas allow the specification of wildcard elements within a grammar. The IBM Transformation Extender XML Schema and DTD importers recognize element wildcards, build the appropriate types, and set the appropriate properties for those types to represent element wildcards in the schema.

During the import process, when a wildcard element resolves to an element that is not defined in the imported grammar, it is represented by a text item.

The value of these text items is the text string from the input buffer that runs from the start of the open tag to the end of the close tag for that element.

XML Schemas allow the specification of an attribute wildcard that matches any number of undeclared attributes within the element tag. The XML Schema Importer recognizes the attribute wildcard and creates a sequence of name and value text item pairs within the attribute list of the enclosing element. The XML validation library fills this sequence with the names and values of attributes that do not match any declared attributes in the attribute list.

Type propagation for Category, Group, and Item Properties

The Propagate option enables to propagate individual and its children property values of a type to its child types where applicable.

Note: This property can only propagate the changes from parent properties to its child properties. Child properties do not have the Propagate option. The Propagate option is disabled if the parent property gives the validation message.

- [To Propagate in Category/Group/Item property](#)
 - [To Propagate Restrictions in Category/Group/Item property](#)
-

To Propagate in Category/Group/Item property

To propagate the changes from the parent element to its child elements using Propagate option.

1. In the Schema workspace Dictionary, open the parent element where you want to propagate its changes to its child elements.
2. Make the desired changes to the property values combo box or text box of the Properties dialog.
3. Hover the cursor in the property values combo box or text box and click on the More icon.
4. Select Propagate.

This will propagate the property values in all the child elements.

To Propagate Restrictions in Category/Group/Item property

To propagate the changes from the parent element to its child elements using Propagate Restrictions option.

1. In the Schema workspace Dictionary, open the parent element where you want to propagate its changes to its child elements.
2. In the Properties dialog, select Restrictions from the drop down.
3. Make the Values from file field false.
4. Enter the text in the Include/Exclude and Description text field.
5. Hover the cursor in the Rule combo box and click on the More icon.
6. Select Propagate Restrictions.

This will propagate restrictions in the Include/Exclude and Description field of all the child elements.

Item restrictions

Item restrictions are an optional feature you can use to specify valid or invalid data objects for an item type. Item restrictions are grouped into three categories: **Value**, **Character**, and **Range**.

Restrictions of an item type are the valid or invalid values for that item. "Include" restrictions are all of the valid values for an item. "Exclude" restrictions specify all of the invalid values. For example, a unit of measure field in the data must be one of a set of values: CN, BX, PK, BR. These values should be defined as "include" restrictions of the item **UnitOfMeasure**.

Defining restrictions for an item restricts the valid data for that item. Partial lists of the valid values are not possible. For example, it is not possible to define the values for a certain item to be {A, B, C, "some other possible values"}.

While you can define restrictions for any item, most items will not have restrictions. For example, you would not want to restrict the valid values of a name field because you would probably want to accept any name as valid data for that field. However, you could assign restrictions to it if needed.

- [Restrictions settings](#)
 - [Ignoring restrictions](#)
-

Restrictions settings

Using the Restrictions property, you can create a "restriction list" to limit an item to a particular value or set of values. Depending on the Item Subclass, you can set value, character, or range restrictions.

- [Value restrictions](#)
 - [Character descriptions](#)
- In the character restrictions view for an item type, there is either an Include or Exclude column, depending on the Restrictions > Rule setting.
- [Range restrictions](#)
 - [Inserting symbols](#)
-

Value restrictions

In the value restrictions view for an item type, there is either an Include or Exclude column, depending on the Restrictions > Rule setting.

Include column

Use the Include column to specify the restrictions to use in determining if input data is valid or invalid for the item. One-to-many values are allowed. The order in which restrictions appear in the list is not relevant.

"Include" values are considered valid. An input item is considered invalid if a character other than those specified in the **Include** column appears in the data.

Exclude column

Use the Exclude column to specify the text character values to be excluded.

"Exclude" values are considered invalid. This option provides a means to specify a default set of restrictions.

To add value restrictions to an item:

1. Go to the Item Subclass  Restrictions property.
 2. Choose Value from the drop-down list.
 3. The restriction list is case-sensitive by default. If you do not want case-sensitivity, set the Ignore case subproperty to Yes to ignore case.
 4. Set the Rule subproperty to Include or Exclude the values that you will provide.
 5. Enter the value restrictions Include or Exclude column.
 6. (Optional) Enter corresponding descriptions in the Description column.
 7. Analyze the schema.
 8. Save changes.
-

Character descriptions

In the character restrictions view for an item type, there is either an Include or Exclude column, depending on the  Rule setting.

- Include character restrictions:
 - Include First column - Use this column to define the first character of the character list.
 - Include After column - Use this column to define the character list of the characters to follow the first character, as defined in the Include First column.
 - Exclude character restrictions:
 - Exclude column - Use this column to specify the substrings in the input data that must be excluded and replaced with the associated reference string.
 - Reference String column - Use this column to define the characters that are to replace the excluded character strings. On output, a character text item is built by using the corresponding reference string when the content contains any of the Exclude character substrings listed. For example, you can define XML item content that excludes markup delimiters by using the corresponding reference strings. This is because XML character text items cannot contain the markup delimiters <, > or & in the raw form.
- [To add character restrictions to an item](#)
-

To add character restrictions to an item

1. Go to the Item Subclass  Restrictions property.
 2. Choose Character from the drop-down list.
 3. The restriction list is case-sensitive by default. If you do not want case-sensitivity, set the Ignore case subproperty to Yes to ignore case.
 4. Set the Rule subproperty to Include or Exclude the values that you will provide.
 5. Enter the character restrictions in the Include First or Exclude column.
 6. Enter corresponding values in the Include After or Reference String column.
 7. Analyze the schema.
 8. Save changes.
-

Range restrictions

Include range restrictions define the minimum and maximum values valid values of a range. *Exclude range restrictions* define the minimum and maximum values of the range that are to be excluded or considered invalid.

Note: Unbound restrictions are not supported.

- [Value not in range](#)
 - [Value not in range](#)
The minimum value and the maximum values can be specified as not included in the range. For example, when a number in the Include Minimum field displayed the Value NOT In Range icon, it indicates that the number is not included as the minimum value.
 - [To add range restrictions to an item](#)
-

Value not in range

The minimum value and the maximum values can be specified as not included in the range. For example, when a number in an Include Minimum field displayed the **Value NOT In Range** icon, it indicates that the number is not included as the minimum value. The range will therefore extend from any value that is greater than that number to the maximum value specified. Similarly, if the **Include Maximum** value is designated as "not in range", the valid range would extend up to any value this is less than the maximum value.

To specify a value as "not in range":

1. Select the field (cell) that contains the value to be designated as "not in range".
 2. Right click on the value and choose the Value NOT In Range menu item.
-

Value not in range

The minimum value and the maximum values can be specified as not included in the range. For example, when a number in the Include Minimum field displayed the Value NOT in Range icon, it indicates that the number is not included as the minimum value.

The range will therefore extend from any value that is greater than the number to the maximum value specified. Similarly, if the Include Maximum value is designated as 'not in range', the valid range would extend up to any value that is less than the minimum value.

To add range restrictions to an item

1. Go to the Item Subclass \geq Restrictions property.
2. Choose Range from the drop-down list.
3. The restriction list is case-sensitive by default. If you do not want case-sensitivity, set the Ignore case subproperty to Yes to ignore case.
4. Set the Rule subproperty to Include or Exclude the values that you will provide.
5. Enter the range restrictions in the Include Minimum and Include Maximum or Exclude Minimum and Exclude Maximum columns.
6. (Optional) Enter corresponding descriptions in the Description column.
7. Analyze the schema.
8. Save changes.

Inserting symbols

Symbols are used to indicate non-printable characters. You can insert a symbol by typing it or by using the Symbols dialog.

For example, to define a carriage return/line feed as an item restriction, from the item restriction view area, right-click and select Insert Symbols from the context menu. From the list of symbols, double-click CR and double-click LF and select OK. The carriage return/line feed symbols are displayed in angle brackets in the item restriction view area: <CR><LF>

Ignoring restrictions

There might be instances when you do not require the data for an item to match any of its restrictions. Suppose you defined restrictions for your data, but you want to run a test on some test data, and you do not want to use the restrictions for this time only. You can ignore the restrictions for a given execution of a map.

In another example, you have a list of valid part numbers that can appear in any data circulated within your company. Suppose you receive data from some other company in the same format as your internal data. The other company may be using different part numbers, so you do not want to make their data conform to the restrictions. When you map the other company's data, you could ignore the restrictions.

For instructions on ignoring restrictions during map execution, see Map Designer and Command Server documentation.

Components

A component represents a data object that is part of another data object.

- [Components are required for group types](#)
- [Defining components](#)
- [Required and optional data](#)
- [Defining component rules](#)
- [Component attributes](#)

Components are required for group types

Categories and groups can have components. Components of group and category types display in the group and category views. Item types do not have components.

Group types represent actual data objects, so groups must have components. The one exception is a partitioned group which is explained in ["Partitioning"](#).

By contrast, a category is used for organizing types and for type property inheritance reasons. Categories do not define actual data objects in detail. A category does not need components.

Each group must have at least one component, unless it is partitioned.

IBM Transformation Extender supports large metadata. It supports more than 65,536 types and 65,536 components in a single schema. To reduce the number of types that maps that use large schemas must process, trim these schemas so that they contain only the required types.

- [Components must be in the same schema](#)
- [Importance of component order](#)
- [Component range](#)

Components must be in the same schema

A component must be a type in the same schema as the type that contains the component.

You cannot define the components of a type by opening up a different schema and dragging components from that tree. You can, however, copy types from one schema to another.

Importance of component order

In the group view, components are listed from top to bottom in the order they appear in the data stream. The component in the first cell appears first in the data stream. The component in the second cell appears next, and so forth.

Component range

A component range can be specified for any component. A component range defines the number of consecutive occurrences of that component. The range (s) represents some or any number greater than one.

When a component is selected, the component name is displayed in the rule bar. Click in the rule bar after the component name to type the component range.

The range is displayed in parentheses immediately after the component name. The range is two numbers separated by a colon. The first number indicates the minimum number of consecutive occurrences of that component. The second number indicates the maximum number of consecutive occurrences of that component. The syntax of the component name and range is:

Component (MIN:MAX)

To enter a range after a component name, type in the rule bar or use the **Set Range** command.

The maximum component range is 2147483647.

Whenever a component has a range, each occurrence of that component may be referred to as a "member" of a series. The words "occurrence" and "member" are used interchangeably in the documentation.

- [Indefinite number](#)
- [Single occurrence](#)

Indefinite number

When there is no maximum number of occurrences of a component (the maximum is indefinite) use the letter s to stand for "Some - you do not know how many". Therefore, a file that contains at least one record and has no maximum number of records has this component:

Record(1:s)

If the range minimum is zero, omit the 0 and only enter s. If a file has a minimum of zero records and no maximum number of records, it would have this as its component:

Record(s)

Remember that when you see (s), the minimum of zero is implied. Therefore,

Record(0:s) is the same as Record(s)

If you enter any number alone in the parentheses, the range changes to a minimum of 0 and a maximum of that number. For example, if you enter (5) for a range and save and close the group view, the next time you open it, you see the range displays:

(0:5)

Single occurrence

If there is a minimum and a maximum of one consecutive occurrence of a component, its range is (1:1). This is the default. If there is no range after a component name, the range (1:1) is implied. When you drag a type to make it a component, it is automatically created with the default component range of (1:1).

Defining components

Most groups need at least one component. Categories do not have components, however, you can define components for a category for inheritance purposes.

To determine the components of a group, ask the question: *This group type consists of what?*

For example, "The file consists of what?" The answer might be "records". So, **Record(s)** is a component of that group or category type.

Suppose you have a data file containing order records for an office supply store. You need to define the components of the type **File**.

- [Guidelines for defining components](#)
- [Complete type name](#)
- [Relative type names](#)

- [Ambiguous type names](#)
 - [Specifying minimum and maximum consecutive occurrences in the component list](#)
 - [Fixed and variable ranges](#)
 - [Specifying a component range](#)
 - [Variable component names](#)
-

Guidelines for defining components

The guidelines below are numbered to allow references to each other. The numbering does not suggest a priority or a sequence to be followed in observing the guidelines.

1. Categories cannot have components.
A category is only used for organizing your schema and for setting common properties.
 2. A partitioned group cannot have components.
A partitioned group always represents a choice among the subtypes of that group. You never map a partitioned group without its subtree, so it does not need components.

Note: A *subtree* is a branch of a schema that includes a type and all of the subtypes that stem underneath it.
 3. If a group is not partitioned, it must have at least one component.
Nonpartitioned groups are sequences of data objects rather than choices. A sequence must contain at least one component.
 4. A type and one of its subtypes cannot be in the same component list.
 5. If a type has components, a subtype can inherit any of those components or any type in the subtree of one of those components.
 6. If a type has no components, a subtype can inherit any type that could be in that type's component list.
 7. A type that has an initiator and a terminator can have itself or one of its ancestors as a component.
 8. A type cannot have one of its subtypes as a component.
-

Complete type name

The complete name of a group is displayed in the title bar of its window. A type's complete name is similar to a path name, beginning at the type and including all types in the path up to the root. Spaces appear between types in a complete type name.

You can have duplicate type names as long as they are not on the same level. Each type has a unique complete name, so there is no doubt as to which particular type is being referenced.

Relative type names

A component name is similar to a relative path name. Component names exclude types that the defined type and component type have in common in their complete names.

The relative type name excludes the names that appear in the complete names of both the type and the component type. For example, because **ROOT** is included in the complete type name of both **Row ROOT** and **Product Column ROOT**, it is excluded from the relative type name **Product Column**.

If more than one type in a schema has the same relative type name as another in a group, then the full path of the type is exported instead of the relative type name. No two types that have different full type names but identical relative type names can be added to a component list.

- [Moving types with the same relative type name](#)
-

Moving types with the same relative type name

In a schema, there could be two types with the same name that exist in different places within the tree. The relative type names could both evaluate to be the same, with respect to the component.

For example, the relative type **Group Category ROOT** might have a type named "Item" in both **ROOT** and **Category ROOT**. Both evaluate to "Item" with respect to the **Group** component. In this scenario, you can drag only one component named **Item** into the component list. An attempt to add a second component with the same name will be blocked.

Ambiguous type names

A component name can refer to more than one type. In this situation, you must change type names so that the component name is no longer ambiguous. The Type Designer does not allow components with the same relative type name to be added to component lists.

If you delete a type that is used as a component of another type, and then perform an "Undo", thus adding the type back, and the component name is ambiguous; the component name remains unresolved in the component list. In this scenario, the "Undo" operation does not result in a complete reversal of the action.

Specifying minimum and maximum consecutive occurrences in the component list

The following table contains examples of how to specify in the component list the minimum and maximum consecutive occurrences for specific data objects:

Data Object	Min	Max	How to Specify
DateField	1	5	DateField (1:5)
DetailRecord	1	100	DetailRecord (1:100)
AddressField	2	3	AddressField (2:3)

Fixed and variable ranges

Sometimes the range of a component is described as fixed or variable. As an example, a fixed range has the same minimum and maximum, (5:5). A variable range has a different minimum and maximum, (1:10) or (s) for example.

Specifying a component range

You can specify a component range by using Set Range to enter the minimum and maximum number of occurrences.

You can select multiple components and use Set Range to apply the same range to each component. A range of (1:s) means that there will always be at least 1 occurrence, but could occur an infinite number of times. A range of (0:s) means that an occurrence is optional, but could occur an infinite number of times.

Variable component names

A component can refer to more than one type. To refer to all possible types whose names could appear at a certain place in the component name, use the word ANY.

The word ANY is like a wild card. It represents any type whose name could appear in that place.

The use of ANY is restricted to **Categories** and partitioned groups. For more information about using ANY in a partitioned group, see "[Partitioning](#)". For a list of reserved words and symbols, see the Design Studio Introduction.

Required and optional data

Sometimes, a certain data object is optional; it does not have to be present in the data. For example, in purchase order data, there might be a billing address and a shipping address. If the company wants the items shipped to the billing address, the shipping address would not appear in the data. The shipping address would be optional; it might not appear in the data.

Another example is a middle name field. Some people do not have a middle name, so the middle name field might be optional.

The Type Designer needs to know what data is optional. This is evident from the component range. The range minimum tells how many occurrences of that object must be present in the data. These are the required occurrences. Optional occurrences are the ones that are not required.

For example, for the following component, the range minimum is zero. No occurrences must be present. It is optional data:

DateField (0:1)

Suppose the component looks like this:

DateField (1:5)

The range is between one and five occurrences. This means that one occurrence of **DateField** is required and the remaining four occurrences are optional.

The following table lists examples of components and explanations of their status.

Component	Status	Reason
LineItem (1:s)	1 occurrence required	Range minimum is 1
Note Field (5:5)	5 occurrences required	Range minimum is 5
RecordID	1 occurrence required	Range minimum is 1
ShipTo (0:1)	0 occurrences required (Optional)	Range minimum is 0
OrderRecord (s)	0 occurrences required (Optional)	Range minimum is 0

- [Significance of required data](#)

Significance of required data

If you define an occurrence of a component as required, you are saying that, for the data containing the component to be *valid*, this component must exist. If it does not exist, the data is *invalid*.

For example, if you define **Record** as having the component **Field (3:3)**, you are saying that there must be three **Fields** in the **Record**. If there are not three **Fields**, then either it is a **Record** in error or it is not a **Record**.

There are other factors, besides the existence of all required components that make data valid. The existence of required components is necessary, but not sufficient, for data to be valid.

Defining component rules

A component rule is an expression about one or more components. It indicates what must be true for that component to be valid. For given data, it evaluates to either "true" or "false". A component rule is similar to a test. If the data does not pass the test, it is invalid.

Component rules are used for validating data. Some important points about component rules are:

- Only components of a group can have component rules. Components of a category cannot have component rules.
- A component rule cannot be longer than 32K.
- If a component is optional and does not appear in the data, the component rule is evaluated after determining that the data is missing. In a component rule, you can specify relationships that depend on existence or nonexistence of data.

Sometimes components have relationships among each other. For example, an address field in purchase order data is preceded by a qualifier field which tells whether the address is a bill-to or a ship-to address. If the value of the qualifier field is `BT`, the address field following it is a bill-to address. If the value is `ST`, the following address is a ship-to address.

The qualifier and address fields are dependent on each other. They only make sense as a pair. If one of these fields is optional and is missing, the other field is not meaningful. If the qualifier is missing, you do not know whether the address is a bill-to or a ship-to. If the address is missing, the qualifier does not qualify anything!

You might want to define this kind of relationship and other relationships among data objects. To do this, use a component rule. For example, use a rule on the address field component to indicate that the qualifier must be present if the address is present.

- [Examples of component rules](#)
- [Component rule syntax](#)
- [Entering object names in component rules](#)
- [Shorthand notation](#)
- [Component rules are context-sensitive](#)
- [Special characters in component rules](#)
- [Comments in component rules](#)

Examples of component rules

A component rule can limit the acceptable values of a component as shown in the following example:

```
Quantity < 10000
Interest Rate > .13 & Interest Rate < .20
WHEN (PurposeCode != "PF", ShipValue = 200|ShipCode < 0)
```

The following example illustrates how a component rule can make the presence of one component mandatory, if another component is present:

```
WHEN (PRESENT (Address Field), PRESENT (Qualifier Field))
WHEN (PRESENT (PhoneNumber), PRESENT (AreaCode), ABSENT (AreaCode))
```

A component rule can compare a component to the result of an arithmetic operation as shown in the following example:

```
SUM (((QuantityOrdered:Item Record:Detail) = TotalQuantity:Summary Record:Detail
Account Balance = Credits - Debits
Extension = Quantity*Price
#Items Field = COUNT (Item Record IN Invoice)
```

Component rule syntax

A component rule is a complete expression that evaluates to either "true" or "false". It can contain functions (for example, PRESENT, COUNT, and SUM). It can also contain arithmetic operators (such as `- + * /`).

A component rule is a statement, so it does not start with an equal sign.

For more information about the syntax of expressions, see the Functions and Expressions documentation.

Entering object names in component rules

A component rule can refer to a component within a component. The syntax for a component is the component name, followed by a colon (`:`), followed by the object of which the component is a part. Whenever you see the colon (`:`) in an expression, it can be interpreted as "component of" or simply "of".

For example, the following expression means "The item number of the order record of the order":

```
Item#:OrderRecord:Order
```

All of the objects that can be used in component rules are shown in the group view. You can enter an object name into a component rule by pressing `Alt` and dragging the object into the edit area. The complete object name is automatically entered.

A component rule can refer to:

- The component it applies to and any nested components.
- Any component above the given component in the component list, and its nested components.
You can enter an object name in a component rule by typing it; however, the recommended method is to press Alt and drag the component into the edit area.

Shorthand notation

In a component rule, the dollar sign (\$) represents the component itself. When you enter an object name in a rule by pressing Alt and dragging the object, the dollar sign is automatically entered in the rule to represent the given component.

Component rules are context-sensitive

Component rules apply to components, not types. It applies to data in a certain context when the data is a component of a given group.

Suppose you have some order data that contains two kinds of records: a regular order record and a bulk order record. In the bulk order record, the quantity ordered must be greater than 1000. In the regular order record, the quantity can be any number. You could put a rule on the quantity ordered component of a bulk order, but not on the quantity ordered of a regular order.

Special characters in component rules

To enter the actual value of a special character in a component rule, you must enter the Hex value for one of the characters. For example, to enter the text value <WSP>, enter the Hex value for the less than sign <<3C>>, and then the rest:

<<3C>>WSP>

See the Design Studio Introduction for a list of non-printable Hex and decimal values.

Comments in component rules

You can add comments to component rules. Comments do not affect how component rules are evaluated.

A comment begins with the characters /* and ends with the characters */. A comment can appear anywhere in a rule as long as it does not separate object names.

For example, the following component rule has a comment:

```
SIZE (Phone# Field:$) >=7
/* Phone numbers must include area code */
```

Component attributes

Attributes can be assigned to a component in a component list.

- [Identifier attribute](#)
- [Restart attribute](#)
- [Sized attribute](#)
- [Include self in size](#)

Identifier attribute

The identifier attribute can be used on a component of a group. The identifier indicates the components that can be used to identify the type to which a data object belongs. All the components, from the first, up to and including the component with the identifier attribute, are used for type identification.

When this data is validated, it knows that, when it reaches the identifier, it has found a specific group. That group, therefore, is known to exist, even if part of the group following the identifier is missing. The Map Designer documentation discusses how data validation occurs.

The identifier attribute is also useful when identifying an object of a partitioned group. For information about using an identifier with partitioned data, see ["Partitioning"](#).

In a component list, there can be only one identifier attribute.

Restart attribute

To continue processing your input data when a data object of a component is invalid, assign the restart attribute to that component.

Do not put the restart attribute on a required component. There must be a sufficient number of valid instances to cover all required components. If you have a required component that is not valid, the restart attribute does not validate the data.

For additional information about using the **Restart** attribute , see the Map Designer documentation.

Sized attribute

The sized attribute is used on a component in which the value specifies the size (in bytes) of the component immediately following it. The sized attribute can be used on more than one component of a group.

For example, you might have a variable length component with a number immediately preceding it that indicates the length of the component: **10Washington**. The size of the component would be 10.

Some important points about using the sized attribute are:

- The component with the sized attribute must be defined as an unsigned integer.
- If a binary byte stream item does not have a fixed size, the component preceding it must specify its size and the sized attribute must be used on that component.

The size of a component is the number of bytes from the beginning of that component, up to and including the end of the component. If a component has a series range (such as [1:3]), the size includes all of the members in the series of that component. If a delimiter separates each member of that series, the delimiters must be included in the size. Also, if release characters appear in the component, they must be included in the size.

The size does *not* include delimiters that separate one component type from the next.

Include self in size

If the value of the component with the sized attribute includes the size of the component, use the **Include Self in Size** option. For example, suppose the component with the sized attribute has a length of seven bytes. The value of the component with the sized attribute would be seven. So, its data would be one byte long, to hold the character seven. If its value includes the length of itself, its value would be $7 + 1 = 8$.

For example, a sized component is 7 bytes.

+ 1 byte (the length of the character 7)

If **Include Self in Size** is selected: 8 bytes

Partitioning

By partitioning, you can define your data to distinguish the difference between data objects based on values in the data or differences in the syntax.

Partitioning is a method of subdividing objects into mutually exclusive subtypes. The partitioned type maintains the same class.

- [Determining when to partition](#)
- [Partitioning types](#)

Determining when to partition

There are some cases in which you will *need* to partition your data and others in which you will *want* to partition your data. You are *required* to partition for unordered data when a data object at a certain place in the data stream can be any number of types and each type has different definitions. Choose to partition for convenience, either to simplify mapping rules or to put additional logic into your data's definition.

- [Required partitioning](#)
- [Partitioning for convenience](#)
- [Benefits of partitioning](#)

Required partitioning

Partitioning is required when components are randomly or partially ordered. The following example represents unordered data:

BGI - 13100,REM,931104,19970424...

AXR - 10930,INV,003X114,19970422...

PVY - 19496,ORD,PO-104-1499,19970425...

BGI - 13100,ORD,PO-182-2587,19970425...

AXR - 10930,INV,003X-114,19970422...

PVY - 19496,REM,931104,19970424...

The file would contain three different types of transactions: **Invoice**, **Order**, and **Remittance**. Each transaction has a different definition based on the type of information it represents.

Partitioning for convenience

You might decide to use partitioning in your schema to build additional logic into the definition of your data. You may also use partitioning to simplify the rules needed in your map.

The following is an example of partitioning to simplify rules. The example compares the differences between rules needed *with* and *without* partitioning. In the rule without partitioning, you would specify a condition for each state abbreviation in each region. This could make your mapping rules long, difficult to read, and difficult to maintain. The mapping rule with partitioning is more concise, self-documenting, and easier to maintain.

Map rule without partitioning:

```
=IF(ShipToCode Field::Input="NY" |  
    ShipToCode Field::Input="NJ" |  
    ShipToCode Field::Input="PA",  
    F_MapEast (Record:Input), NONE)
```

Map rule with partitioning:

```
=F_MapEast (EXTRACT (Record:Input,  
PARTITION (ShipToCode Field::Input, East)))
```

Benefits of partitioning

Using the [previous example](#), explore the benefits of partitioning.

- The rule is shorter than the rule without partitioning. The knowledge of which states belong to each region is maintained in the type tree rather than the map rule.
- It is easy to read this map rule and understand the mapping function being performed. For example, if the state belongs to the list of states in the eastern region, execute the **MapEast** functional map.
- The partitioning method is easier to maintain. If a value for **State** is added or moves from one region to another, it can be easily changed in the schema and automatically reflected in any mapping rules that reference the partitioned object.
- Partitioning using a restriction list used with the **Ignore Case** setting eliminates the need for PROPER, LOWERCASE, or UPPERCASE functions to compare each state with a literal.

Partitioning types

When data objects of different types appear in the same place in the data, the types must be distinguishable. This means that the data needs to be distinguishable by their definitions in the schema.

When the data object at any given point in the data may belong to any of a number of different types, there must be some way to tell the difference between them. To do this, you create a type and define a mutually exclusive subtype for each data object that may appear in the same place in the data. Once the subtypes are created, the subtypes also need to be distinguishable. Subtypes are distinguishable based on a value in the data or in the syntax of the different types.

- [Partitioning items](#)
- [Partitioning an item type using initiators](#)
- [Partitioning an item type using restrictions](#)
- [Partitioning an item type by format](#)
- [Partitioning groups](#)

Partitioning items

Use one of the following three methods to partition items in schemas:

- Initiators
- Restrictions
- Format

Partitioning an item type using initiators

To partition by initiator, each subtype must have an initiator and the value of the initiator must be unique for each subtype.

Partitioning by initiator is the most efficient method of partitioning.

Partitioning an item type using restrictions

If an item has restrictions, you can partition that type, create mutually exclusive subtypes, and divide the restrictions between subtypes. A restriction cannot appear in more than one subtype of that item.

- [Example of using restrictions](#)

Example of using restrictions

You have new data that needs to be entered into the schema designer. Each new record contains the name of the employee and his or her department.

The **Employee List** schema illustrates components of **Record**.

The following is the new data containing the name of the employee and the department.

Steven Barlow,Doc

Heather Proust,Qa

Mary Whiting,Doc

Genie Elks,Sup

Francine Maxwell,Dev

Mark Brown,Sup

Daryl Schwartz,Acc

Harry O'Brian,Sal

Ellen Randolph,Dev

Paula Keller,Qa

Define the values of **Department** as a character text item.

Define the example data as valid restrictions (enter in the **Include** column) of **Department**:

If the departments are located in different offices, you can divide the data into separate files; one file per office. To do this, map the data from the main office to one file, the data from the development office to one file, and the data from the support office to another file. Then create subtypes of **Department: MainOffice**, **DevelopmentOffice**, and **SupportOffice** and partition **Department**. When you partition **Department** and create subtypes, you are saying that a given department data object belongs to only one of the subtypes based on its value.

The subtypes of **Department** inherit the restrictions of **Department**. Now allocate restrictions among subtypes. To do this, delete the restrictions from the subtype that do not apply to that particular office. For example, the **MainOffice** item has only the departments in that office, **DevelopmentOffice** item has only the departments in that office, and the **SupportOffice** item has only the departments in that office.

Partitioning an item type by format

Subtypes that differ by their format are distinguishable from each other.

Partitioning groups

Use one of the following three methods to partition groups in schemas:

- Initiators
- Identifiers
- Component rules
- [Partitioning a group type using initiators](#)
- [Partitioning a group type using identifiers](#)
- [Partitioning a group type using component rules](#)

Partitioning a group type using initiators

To partition by initiator, each subtype must have an initiator and the value of the initiator must be unique for each subtype.

The method of partitioning by initiators for group types is similar to item types.

Partitioning a group type using identifiers

In a component list, only one component can have the identifier attribute.

The identifier attribute distinguishes the components that can be used to identify the type to which a data object belongs. Typically, use this technique to distinguish group partitions when components following the identifier are different for each partition, or, if you have a multilevel partitioned subtree. In the latter case, using an identifier accelerates data validation.

A partition is valid when each component up to and including the identifier is validated. If the set of components is valid, the partition exists.

If the identifier set of components is not valid, the partition is determined not to exist. Either validation occurs for the next partition at the same level (if there is one) or it is determined that the partitioned group does not exist.

If the partition exists, what occurs next depends on the position of the partition type in the subtree.

- If the partition is partitioned (that is, it has subtypes), the rest of the components are skipped and the process begins to validate subtypes until a subtype is valid, exists and is in error, or does not exist.
- If the partition has no subtypes, the remaining components are validated. If all remaining components are valid, the partition not only exists, but also is valid. If one or more components are found to be in error, the partition exists, but its type is in error. If the partition exists, but its type is in error, the error is propagated back up the partitioned subtree until the group being validated is reached. When a partition is found to exist, the system will not continue to search for partitions.

To specify a component as the identifier:

1. From the schema editor, double-click a component.
The Component view opens.
2. In the Component column, right-click on the component and choose Identifier from the context menu.

The identifier symbol appears in the component column next to the component name.

Partitioning a group type using component rules

Component rules are used to partition data when a value or range of values can be used to distinguish one partition from another.

Type inheritance

Properties, components, and restrictions of a type can be inherited by the types created as subtypes under it. Some properties of a type can also be propagated to already existing subtypes.

When you create a type, it becomes a subtype of whatever type is selected at the time. Everything that defines a type gets passed down: properties (with the exception of the Partitioned property), components, and restrictions. After a type is created, you can then modify any aspect of its definition.

When a group is created within a category, the item properties do not apply, so they are not inherited. When an item is created under a category, the group properties are not inherited.

- [Inheritance of item properties and restrictions](#)
- [Inheritance of category properties and components](#)
- [Propagation of type properties](#)

Creation time is not the only time type properties can be passed from a type to its subtypes. You can use propagation to pass along certain type properties.

Inheritance of item properties and restrictions

Properties and restrictions of items are inherited when a new item is created.

For example, an item named **Department** is defined as a character text item that has a content size minimum of 2 and maximum of 3. Department has a list of "include" restrictions that consists of valid departments: ACC (Accounting), SLS (Sales), and MKT (Marketing).

The type **MainOffice** is created as a subtype of **Department**. **MainOffice** inherits the properties and restrictions of **Department**.

You can delete any restrictions that are not applicable to **MainOffice**.

Inheritance of category properties and components

Categories can be used for organizing types and for inheritance reasons. Generally, you would use categories when you want to put items, groups, and possibly other categories under it as subtypes.

- [Organizing types under a category](#)
- [Using categories for inheritance](#)
- [When not to use categories](#)

Organizing types under a category

If you have two tables defined in one schema, you could divide the types of the two tables into different categories. A benefit of using a category type is that you can have any class of subtype: groups, items, and other categories.

Using categories for inheritance

The properties of a category include group and item properties. Assign properties to a category that you want types beneath the category to inherit.

Any group created under a category inherits the property of the category. Any item created under a category inherits the category's item properties. A category created under a category inherits both the group and item properties. Each type created under a category inherits the other properties of the category, such as initiator and terminator.

As an example, if most groups in your **LabInfo** data are infix delimited with ~ and most items in your **LabInfo** data are unsigned integers, you can define these as the properties of the category **LabInfo**. Any types you create under **LabInfo** inherit its properties.

You can change the properties of the root type, which is a category. Categories can have components, so you can also use them for inheriting components.

When not to use categories

It is best not to use a category type when the subtypes will specifically be item types or specifically be group types.

For example, you have a type named **Field** for which the only subtypes will be items. If you make **Field** a category type, each time you create a subtype, the new type will also be a category type by default. You will have to change the **Class** property from **Category** to **Item** for each subtype that you create. Instead, make **Field** an item type so that the **Class** property for any subtypes you create will automatically be defined as item types.

Propagation of type properties

Creation time is not the only time type properties can be passed from a type to its subtypes. You can use propagation to pass along certain type properties.

Propagation passes the setting of a particular type property from the given type to all of the types in the subtree as applicable and defined by their context. When you select a type property for propagation that does not apply to a given subtype, it is not propagated.

- [**Propagation of type properties common to all types**](#)

You can propagate some type properties that are common to all types such as Item Subclass, Description and Type Syntax, to each of its subtypes, without the process checking additional criteria to determine if the type property can be propagated.

- [**Propagation of type properties for specific types**](#)

You can propagate some other type properties such as Separators and Restrictions, to each of its subtypes for specific types, with the process checking additional criteria to determine if the type property can be propagated.

Propagation of type properties common to all types

You can propagate some type properties that are common to all types such as Item Subclass, Description and Type Syntax, to each of its subtypes, without the process checking additional criteria to determine if the type property can be propagated.

The type properties that fall into this category are the properties in the root of the tree, or are children properties that apply to all types to which their parent property is set, and are global in that these properties apply to all items.

The propagation process checks that the settings for some type properties such as the Item Subclass property, the Interpret as property, and if applicable, the Presentation property in a type, match the settings for these same properties in its subtypes, and if they match, the selected property is propagated. But, the propagation process does not check the settings of other properties to determine if the selected property exists in the item, because the selected property in this case, applies to all item types.

Examples of propagating a type property that has no children properties

A given category type that is in the root of the tree has the following type property setting: Description. \geq An example. The Description type property has no children properties. When you select the Description type property setting of An example to be propagated, the process propagates the Description setting of An example to all types in the tree. The Description property is a type property that is common to all types.

A given category type has the following type property settings: Item Subclass. \geq Text and for its child property, Interpret as. \geq Character. When you select the child Interpret as type property to be propagated, the process propagates the Interpret as setting of Character to each subtype of that category that also has the same Text type property setting. The Interpret as property is a type property that is common to all types except syntax types.

Examples of propagating a type property that has children properties

A given category type has the following type property settings: Item Subclass. \geq Number, and for its children properties, Interpret as. \geq Binary, and Presentation. \geq Integer. When you select the parent Item Subclass type property to be propagated, the process propagates the Item Subclass setting of Number, as well as the settings for its children properties, Interpret as setting of Binary, and the Presentation setting of Integer, to each subtype of that category. The Item Subclass property is a type property that is common to all types.

A given category type has the following type property settings: Item Subclass. \geq Text and for its child property, Interpret as. \geq Character. When you select the parent Item Subclass type property to be propagated, the process propagates the Item Subclass setting of Text, as well as the setting for its child property, Interpret as setting of Character, to each subtype of that category. The Item Subclass property is a type property that is common to all types.

Propagation of type properties for specific types

You can propagate some other type properties such as Separators and Restrictions, to each of its subtypes for specific types, with the process checking additional criteria to determine if the type property can be propagated.

The type properties that fall into this category are children properties that do not apply to all types to which their parent property is set, and are specific to a type, in that these properties do not apply to all item types.

The propagation process checks that the settings for some type properties such as the Item Subclass property, the Interpret as property, and if applicable, the Presentation property in a type, match the settings for these same properties in its subtypes, and if they match, the selected property is propagated. But, depending on the property being propagated and the context, the process also checks other property settings. If you are propagating a child property, depending on the context, the process also checks the settings of the parent property, because the selected property might only apply to specific item types.

Examples of propagating a type property that applies to a specific type

A given category type has the following type property settings: Item Subclass. \geq Number, and for its children properties, Interpret as. \geq Character, and Presentation. \geq Decimal. When you propagate the Separators child property, the process affects only types in the schema that also have those same Number, Character, Decimal type property settings. The Separators property is a type property that applies to the specific Presentation types, Decimal and Integer. For those types that have the Presentation type property setting of Zoned, the Separators property does not apply; it is not an available type property.

A given category type has the following type property settings: Restrictions. \geq Value and for its child property, Ignore case. \geq No. The Ignore case type property has no children properties. When you select the child Ignore case type property setting of No to be propagated, the process propagates the Ignore case setting of No to all types in the category whose parent Restrictions property setting is Value. The Ignore case property is a type property that applies to a specific Restrictions type, Value. For those types that have the Restrictions type property setting of Range, the Ignore case property does not apply; it is not an available type property.

Error detection

During map execution, the input data is compared to the data definition in the type tree. If the data does not match the definition, it is invalid or "in error".

To validate a data object as belonging to a certain type, the data must be matched to its type definition. For data to be valid, the following must be true:

- The data must have the properties that were defined for the type.
- If the type is an item that has restrictions, the data object must match one of the restrictions.
- If the type is a group, the components of the data object must match those defined in the group view, and each component rule must evaluate to "true" at map execution time.

Ultimately, all data objects, no matter how complex, consist of items because items represent the smallest unit of data. When all of the items that collectively comprise a component are found, the component has been "found."

When executing a map, invalid data is recorded in the trace file. You can decide what you want the system to do when errors are encountered: you can map the errors to an output and also have the system ignore invalid data. For methods on ignoring invalid data, see ["Restart attribute"](#).

- [How error detection works](#)
 - [Existence indicators](#)
 - [Existence versus presence of components](#)
-

How error detection works

When data is validated, the system may encounter data that does not match its type definition. A combination of how that type is defined and what data shows up determines what happens next. As data is validated, the system tracks valid data objects, data objects that exist but are in error, and data that cannot be recognized as belonging to any type.

If a data object is valid, the information is recorded, as needed, and validation continues.

If the system is looking for a data object of a particular type, and something in the data does not exactly match the type definition, the system bases the determination on whether to continue validation on whether that data object is known to exist.

If a data object contains enough information to determine its existence, errors associated with that data object are recorded and validation continues.

If a data object exists but contains errors, it is marked as an error. If a restart point is specified for the component of the type in error, these errors are ignored and validation continues.

If a data object does not contain enough information to determine its existence, the following occurs:

- If no validation recovery mechanism is specified, validation is stopped because it will get lost if it continues. Validation recovery mechanisms are discussed in the section ["Error Recovery"](#).
- If recovery mechanisms are specified, the system returns to the nearest restart point and resets what it is looking for. It proceeds by examining the data byte-by-byte until it either recognizes something or reaches the end of the data stream. All rejected bytes (from the first byte not associated with a type to the last unrecognizable byte) are collectively marked as an unidentified foreign object (UFO). A UFO is data in error with no valid data contained within it.

In summary, the data object of an input card could be valid, yet contain errors. If the data object of any input card is invalid, output data is not built. If all input card data objects are valid, the input is mapped to the output based on the map rules. Operators, most functions, and map references operate on valid data. To map invalid data, use the **REJECT** function.

Existence indicators

When the source or destination of a data object exists, the system knows that the entire data object of that source or destination exists. For example, if a source specified to contain a transaction is a file, and that file exists, the transaction exists. On the other hand, if you get a `Source not available` message, the data object of that source does not exist because the source itself does not exist.

When the data object of a source or destination exists, specific information about errors will appear in the data. For example, you might have any of the following:

- The data object is valid because it conforms to its type definition.
- The type of the data object exists, but has no content.
- The data object is in error because it does not conform to its type definition.
- The data object is valid but contains errors. You indicated that you wanted to ignore certain errors.

In addition to these conditions, the system can also tell if there is any unknown data remaining after the card object has been recognized. This can occur if there really is "junk" at the end. It may also be that enough data was identified to determine the status of the entire object and the data at the end could not, for some reason, be determined to have anything to do with the data object.

When a data object exists, the status of that object is determined by validating its type properties. If an entire data object is an item, it is easy to determine what the status is: whether it is valid or whether it does not match its item format, whether its value is not one of the specified restrictions, or whether it is missing an initiator or terminator. If the source or destination exists and the entire data object is an item, there is nothing else it could possibly be.

If the type of the entire data object is a group, the status of each component must be determined. The system uses existence indicators to determine whether a group component exists. If a component exists, the system can also tell you its status. If a component is required in another component that exists, the system will notify you if that required component is missing.

Existence versus presence of components

A component exists if the data object in which it is contained exists and if the component takes up space (if there is at least one byte in the data stream representing that component).

For example:

- If a delimiter appears as a placeholder for that component.
- If the type of that component has an initiator that distinguishes that component from any other component that may appear at that position in a data stream, the component exists if the initiator is there.
- If the type of that component is a group with an identifier and all components up to and including that identifier have been found to be valid.
- If the type of the component is an item with a restriction list and one of its restrictions appear in the data.
- If the type of the component is an item with a **Padded To** length specified and the item contains at least the number of bytes in the **Pad To** length.
- If the type of an optional component is valid, but the component rule evaluates to FALSE, the component only exists if there is a syntactic placeholder.

Data might exist, but might not be present. The presence of data is determined by whether the data has content. The following definitions are used in other parts of the documentation:

EXISTS

Something in the data represents that data object.

CONTENT

The data contains at least one byte other than a syntactical placeholder, such as a pad character or delimiter.

PRESENT

The data object exists and has content.

ABSENT

The data object does not exist or it exists and has no content.

Required components must exist. They do not necessarily have to be present. For example, a required item with no minimum content size can exist, be absent, and still be valid. However, if some minimum content size is specified, a required component is not valid if there is no minimum content.

The existence of optional components depends on the context in which they are used. For example:

- In a delimited group, an optional component must exist if data follows.
- In a fixed group, an optional component must exist if trailing white space is required. If no trailing white space is required, an optional component must exist if data follows.
- In an implicit group, an optional component is not required.

Unlike a required component, when an optional component exists, it might not have content - even when the minimum content specification is greater than zero bytes. The following table explains the conditions that must be met for required and optional components.

	EXISTS	CONTENT	PRESENT	ABSENT
Required Component	Must exist in any context	Must conform to content specification	Must be present if minimum content specification is greater than zero bytes.	Can be absent only if minimum content specification is zero.
Optional Component	Depends on context	Need not have content	Not necessary	Can be absent

Error recovery

It is good practice to map invalid data when it is encountered. For example, you have a file of records and some records are invalid. If your structure is designed to map invalid data when it is encountered, processing can continue without interruption.

To map invalid data of a particular object, you can assign the Restart attribute. The Restart attribute allows the processing of input data to continue when a data object of a component is invalid. Errors are ignored for invalid data object during validation. Using error-handling functions, you can map the invalid data.

- [Using the Restart attribute](#)
- [How the Restart attribute works](#)
- [Mapping invalid data](#)

Using the Restart attribute

The Restart attribute provides instructions for handling errors encountered in a data stream. The Restart attribute gets assigned to a component and takes effect during data validation.

During data validation, the Restart attribute identifies a "start over" point when an unidentified foreign object (UFO) is encountered in the data. All unrecognized data is considered an error of the type with the Restart attribute assigned.

If you are mapping data of a component with a Restart attribute in place, only the valid occurrences of that component are mapped. For example, suppose your input is a file of records, and the **Record** component of **File** has the Restart attribute. Suppose some of the records are invalid. When you map the **Record** objects, only the valid records are mapped. To map the invalid records (for example, to an error file) use an error-handling function, such as REJECT, in the map rule.

For information about error-handling functions, see the Functions and Expressions documentation.

Typically, you would assign the Restart attribute to a component that has a series range. For example, (1:20) or (s), and the objects in the series are independent of one another. If each record in your file is independent from the others, it makes sense to ignore an error if it encounters an invalid one. A bad record does not make the next record meaningless. But suppose the records are related. Maybe they are records related to one purchase order; if any record is bad, you want to stop after validation because it is important to have all records. Losing any records would make the overall data incomplete.

For additional information about using the **Restart** attribute, see the Map Designer documentation.

Tip: To capture all errors of the component using Restart, place the Restart attribute at the highest level possible in the schema.

How the Restart attribute works

The Restart attribute can do the following tasks:

- Identify valid data objects that contain objects in error.
- Tell the system where to start over when a unidentified foreign object (UFO) is encountered in the data during the validation process.
All unrecognized data is considered to be an error of the type with Restart assigned to it.
- Identify the UFOs and existing data objects in error that are ignored when mapping input to output.

When an invalid data object is a component that does not have the Restart attribute assigned, that component is marked in error. If components from the beginning of the data are marked in error because no Restart attribute is assigned, the following results occur. Processing stops after validation, and input data is not mapped to output.

Do not put the Restart attribute on a required component. There must be a number of valid instances that is enough to cover all of the required components. If you have a required component that is not valid, the Restart attribute does not validate the data.

When the Restart attribute is added to components that have only one mandatory element, it is ignored. Adding Restart to the parent level works only when there are no children that have one mandatory value. In this scenario, the ISERRORS and REJECT functions would report the first error. But if there are other errors, they are not captured because Restart stopped and restarted at the next parent level.

The Restart attribute is supported on components of Xerces schemas.

For more information about using the **Restart** attribute, see the Map Designer documentation.

Mapping invalid data

You can map the invalid or rejected data which you can use to determine what is wrong with the data. To do this, you use the restart attribute, in conjunction with the **REJECT**, **CONTAINSERRORS**, and **ISERROR** functions in map rules. For information on using these functions, see the Functions and Expressions documentation.

Schema analyzer

Use the schema analyzer to analyze your type definitions.

The schema analyzer analyzes type definitions and ensures internal consistency. For example, if you defined a group as fixed, but accidentally defined one of its items with no **Padded To** length, errors occur during analysis.

The analyzer checks your data definitions for logical consistency. It does not compare your definitions to your actual data. The resulting analyzer messages indicate whether your schema definitions are acceptable; not whether they match your data.

- [Mapping effects](#)
- [Internal consistency](#)
- [Logical analysis](#)
Logical analysis addresses the integrity of the relationships that you define.
- [Structural analysis](#)
Structural analysis addresses the integrity of the underlying database.
- [Error and warning messages](#)

Mapping effects

The analyzer helps you define your data by locating objects in your input data and creating the objects in your output data. The analysis indicates whether there is something in your definition that may prevent a correct mapping of your data.

If you do not analyze a tree before you map the data or you analyze a tree and do not resolve the errors, you will be warned that you may receive unpredictable results when you map.

Internal consistency

The analyzer checks the logic of your data definitions. For example, suppose you defined a group **PO** as fixed and consists of **Line(s)**. For the **PO** to be fixed, it cannot have an indefinite number of **Lines**. An analysis error would occur, indicating that you defined **PO** as fixed but it has a variable number of components.

Logical analysis

Logical analysis addresses the integrity of the relationships that you define.

Logical analysis detects, for example, undefined components, components that are not distinguishable from one another, item restrictions that do not match the properties of that item, and circular type definitions. The analyzer also checks delimiter relationships to each other and to components, undefined inherited relationships, and logic errors contained in component rules.

Structural analysis

Structural analysis addresses the integrity of the underlying database.

Generally, you should not encounter structural analysis errors. Structural analysis might be able to detect and possibly correct defects caused by system environment failures.

Error and warning messages

Analysis results might contain error and warning messages.

Warnings are relatively insignificant. They indicate an inconsistency that occurred when you changed something in the tree that was resolved. For example, if you change a group to an item, the analyzer removes the components of that type because items do not have components. To remind you of this change, the analyzer issues a warning.

Errors are important. An error is a problem in your type definitions that you should correct. An error may result in unpredictable results in your mapping.

Occasionally there are errors that prevent the analysis of the rest of your definitions (for example, when a component type cannot be found in the tree). **Analysis halted before completion** is displayed. This error might be due to one of the following:

- Undefined COMPONENTs found: ending analysis.
- Circular reference found in COMPONENT list: ending analysis.

When this occurs, correct the errors and analyze the tree again.

Distinguishable objects

Differences can be identified for objects in a data stream. This information is helpful when a schema analysis produced an error message concerning objects that are not distinguishable.

- [Objects in a data stream](#)
- [Schema analyzer and distinguishable objects](#)
- [Bound types](#)
- [Bound components](#)
- [Group starting set](#)
- [Group unbound set](#)
- [Initiator-distinguishable types](#)
- [Distinguishable objects of the same component](#)
- [Content-distinguishable components](#)
- [Content-distinguishable types](#)
- [Ending-distinguishable types](#)
- [Distinguishable data objects of an implicit group](#)
- [Distinguishable data objects of an explicit group](#)
- [Distinguishable syntax objects](#)

Objects in a data stream

A data stream is a byte-by-byte flow of data. Objects in a data stream include both data objects and syntax objects. Syntax objects indicate where a data object begins or ends. They include separators, initiators, terminators, delimiters, release characters, and pad characters. Sometimes, data values are used to identify where another data object begins or ends.

It might be necessary to distinguish between data objects in a component series, data objects of different types, or even syntax objects.

Schema analyzer and distinguishable objects

The schema analyzer indicates whether your data definitions are sufficient to distinguish the objects in your data stream. The following discussion explains how you can define your data so that the objects that need to be distinguishable are distinguishable. Chances are, if you analyzed your tree and received a message that involves distinguishable objects, you need to define the types differently or more specifically.

Bound types

A type is bound if its definition makes it clear where an instance of that type ends. If a type is bound, different objects of that type can be distinguished in a data stream. A bound type is easier to distinguish between an object of that type and an object of another type. The following tables describe how types may be bound.

An object of this type:

Is bound if any of the following is true:

Item

It is padded to a fixed size or its minimum and maximum content size are equal.
It has a terminator.

It has an include restriction list.

Partitioned item

Each non-partitioned item in the subtree is bound.

Sequence Group

It has an explicit fixed format.
It has a terminator.

Its last component is bound.

It is postfix delimited and its last component has a fixed range. For example, **Comment Field (3:3)** has a fixed range of **(3:3)**.

Partitioned Group

Each non-partitioned group in its subtree is bound.

Choice Group

It has a terminator.
The type of each selection component is bound.

Unordered Group

It has a terminator.

Bound components

A component is bound if it is possible to tell if a data object belongs to that component without comparing it to a component that follows it.

- [Component of a fixed group](#)
- [Component of an explicit delimited group](#)
- [Component of an implicit group](#)
- [Component of a choice group](#)
- [Component of an unordered group](#)

Component of a fixed group

Condition

Example

A range maximum is specified (it is not "s"), and its type is either a fixed group or an item whose length is fixed.

The component **InventorySection (0:3)** is bound because it is assumed there will be spaces in the data stream for 3 **InventorySections**.

Component of an explicit delimited group

Condition

Example

A range maximum is specified (it is not "s") if a component of the same group follows.

In the component list, **Security Sequence (0:10) Trailer** is bound in an explicit delimited group. It is assumed there will be a delimiter for all 10 **Security Sequences** in the data stream because **Trailer** is required.

The component is the last one and the explicit group has a terminator.

In the component list, **Security Sequence (s)** is bound in an explicit delimited group with a terminator. The system assumes the terminator will appear to indicate that there are no more **Security Sequences**.

Component of an implicit group

Condition

Example

Its range is fixed and its type is bound.

The component **Inventory Record (3:3)** is bound, if the type **Inventory Record** is bound according to any of the conditions listed under "[Bound Types](#)".

It has a component rule that binds it.

The component **PO Record (s)** is bound, if it has the following component rule which binds it:

PO# Field:PO Record = PO# Field:PO Record[LAST]

The schema analysis checks only that the component has a rule that refers to the component itself. The analysis does not check that the rule binds the component.
You must ensure that the rule is one that binds the component.

It is sized (using the Sized attribute) by the component that precedes it.

The component **Name Field (0:2)** is bound, if the previous component in the group has the Sized attribute.

Component of a choice group

Condition

Example

A component is bound if the type of a component is bound.

The component **Customer Record** is bound if the type **Customer Record** is bound.

Component of an unordered group

Condition

Example

A component is bound if its range is fixed and its type is bound.

The component **Address Field** is bound if its range is fixed, for example **(3:3)**, and the type **Address Field** is bound.

Group starting set

Data objects in a data stream need to be distinguishable when they could belong to two different groups. Specifically, the difference between the data that can come first in one group and the data that can come first in the other group. All of the possible types of data that may appear first in a group are referred to as the group's starting set.

The following describes the starting set of a group based on its group format:

Group Format	Starting Component Set
Explicit	Includes the type of its first component. <ul style="list-style-type: none">• Delimited• Fixed
Implicit	<ul style="list-style-type: none">• Includes the type of all components up to and including the first component that has a minimum range of at least one.• Includes the type of each component.
Choice	Includes the type of each component.

Group unbound set

Based on the nature of a bound group, the end of a bound group is determined without analyzing the data that follows it. However, if a group is not bound, the data that belongs to that group must be distinguishable from data that belongs to another type that might follow it in the data stream.

The unbound set of a choice or unordered group consists of the type of each component. The unbound set of a partitioned group consists of the unbound set of each unbound partition.

- [Unbound set of a sequence group](#)
-

Unbound set of a sequence group

The unbound set does not include a type that could come last, if that type is a required occurrence. For example, if the type **Comment Record** could appear last and the component is specified as **Comment Record (2:2)**, it is not in the unbound set, because the two occurrences of **Comment Record** are required. However, if **Comment Record** could appear last and the component is specified as **Comment Record (1:2)**, then **Comment Record** is in the unbound set, since its second occurrence is optional.

To determine the unbound set of a group, start with the last component of the group and proceed backward up the component list. If a component is unbound or has a minimum range of zero, continue up the list. Stop at the first component that is bound or that is unbound but has a minimum range greater than zero. Essentially, a group's unbound set is everything that is not clearly defined at the end of the group.

Initiator-distinguishable types

Initiator-distinguishable types are used during validation to determine existence of a type object based on the existence of its initiator.

- [Determining if a component is initiator-distinguishable from its following set](#)
- [Determining if a partition is initiator-distinguishable from its following set](#)
- [Determining if two types are initiator-distinguishable](#)

Determining if a component is initiator-distinguishable from its following set

A component is initiator-distinguishable from its following set if:

- The component is not a member of the identifier set and
- The type of the component has an initiator and
- The following set is empty or
- The type of the component is initiator-distinguishable from each type in its following set.

Schemas are analyzed to determine if components are initiator-distinguishable. Each implicit sequence group, choice group, and unordered group is analyzed to determine if their components are initiator-distinguishable. A component is marked as initiator-distinguishable when that component is initiator-distinguishable from its following set. The basis for this determination is found in ["Determining If Two Types are Initiator-Distinguishable"](#).

Determining if a partition is initiator-distinguishable from its following set

In a partitioned type, a partition is initiator-distinguishable from its following set if:

- The type of a partition has an initiator and
- The following set is empty or
- The type of the partition is initiator-distinguishable from each partition in its following set and the following set of a partition is the type of each partition that may follow.

Schemas are analyzed to determine if partitions are initiator-distinguishable. Each partitioned type is analyzed to determine if its partitions are initiator-distinguishable.

Determining if two types are initiator-distinguishable

The following table lists ways two types may be initiator-distinguishable. This is helpful if data validation errors indicate a type does not exist.

Type1	Type2	How to define them as initiator-distinguishable
item	item	If Type1 and Type2 have an initiator, and the initiators are different.
item	sequence group	Either: <ul style="list-style-type: none">• Type1 and Type2 both have an initiator and the initiators are different.or• Type1 has an initiator, Type2 does not, Type2 has no delimiter, and Type1 is initiator distinguishable from type of each component in the starting component set of Type2.
item	choice group or unordered group	Either: <ul style="list-style-type: none">• The Type1 and Type2 both have an initiator and the initiators are different.or• Type1 has an initiator, Type2 does not, Type2 has no delimiter, and Type1 is initiator distinguishable from type of each component of Type2.
item	partitioned item or partitioned group	Type1 has an initiator and is initiator distinguishable from each partition of Type2.
partitioned item	item or sequence group	Type1 has an initiator and each partition is initiator distinguishable from Type2.
partitioned item	partitioned item or partitioned group	Type1 has an initiator and each partition is initiator distinguishable from each partition of Type2.
sequence group	item	Type1 has no identifier, Type1 and Type2 both have initiators, and the initiators are different.

Type1	Type2	How to define them as initiator-distinguishable
sequence group	sequence group	Type1 has no identifier and or • Type1 and Type2 both have initiators and the initiators are different • Type1 has an initiator, Type2 does not have an initiator, Type2 has no delimiter, and Type1 is initiator distinguishable from the starting component set of Type2.
sequence group	choice group or unordered group	Type1 has no identifier and or • Type1 has an initiator, Type2 does not have an initiator, Type2 has no delimiter, and Type1 is initiator distinguishable from the type of each component of the Type2.
sequence group	partitioned item	Type1 has no identifier and Type1 must be initiator distinguishable from each partition of Type2.
sequence group	partitioned group	Type1 has no identifier and the Type1 must be initiator distinguishable from each partition of Type2.
partitioned group	partitioned item	Type1 has no identifier and each partition Type1 must be initiator distinguishable from Type2.
partitioned group	sequence group	Type1 has no identifier and each partition of the Type1 must be initiator distinguishable from Type2.
partitioned group	partitioned item	Type1 has no identifier and each partition of Type1 must be initiator distinguishable from each partition of Type2.
partitioned group	partitioned group	Type1 has no identifier and each partition of Type1 must be initiator distinguishable from each partition of Type2.
choice group or unordered group	item	Type1 and Type2 both have an initiator and the initiators are different.
choice group or unordered group	partitioned item	Type1 is initiator-distinguishable from each partition of Type2.
choice group or unordered group	choice group or unordered group	Either: • Type1 and Type2 both have an initiator and the initiators are different. or • Type1 has an initiator and Type2 does not, Type2 has no delimiter, and Type1 is initiator-distinguishable from the type of each component of Type2.
choice group or unordered group	sequence group	Either: • Type1 and Type2 both have an initiator and they are different. or • Type1 has an initiator and Type2 has no initiator and no delimiter, and Type1 is initiator-distinguishable from the type of each component in the starting component set of Type2.
choice group or unordered group	partitioned group	Type1 is initiator distinguishable from each partition of the Type2.

Distinguishable objects of the same component

When a component has a series range, for example, (1:10) or (s), one occurrence of that component must be distinguishable from the next occurrence of that same component.

Different data objects of a component are distinguishable if any of the following is true:

- The component type is bound.
- The component type is a non-partitioned group with an unbound set that is content distinguishable from the component type as a whole.
- The component type is a partitioned group and the unbound set of each non-partitioned group in its subtree is content-distinguishable from the component type as a whole.

Content-distinguishable components

A component is distinguishable from its following set when the component is:

- Initiator-distinguishable from its following set.

or

- Content- distinguishable from its following set.

A component is content-distinguishable from its following set when:

- The following set is empty.

or

- The type of the component is content-distinguishable from each component in its following set.

Content-distinguishable types

The following table lists the ways in which two types may be content-distinguishable. The table is helpful particularly if you analyzed your tree and received an error indicating that two types are not distinguishable. Look up the combination of types in the first two columns and read the list of ways to define them that would make them distinguishable.

For example, if you get an error indicating that **X** is not distinguishable from **Y**, where **X** is an item, and **Y** is a partitioned group, you would find the row in the table where **Type1** is an item and **Type2** is a partitioned group.

Remember that the order in which you compare two types matters. When you ask the question "Is type A content-distinguishable from type B?", this is not the same question as "Is type B content-distinguishable from type A?"

If you are trying to determine whether two types are distinguishable and you follow the guidelines in the following tables, you may encounter a situation in which you are comparing a type to itself. A type is never distinguishable from itself.

Another thing to keep in mind is that the context in which a type is used matters. When you ask the question, "Is type A, as a component of type C, content-distinguishable from type B?" this is not the same question as, "Is type A, as a component of type D, content-distinguishable from type B?"

The following table shows how to define types as content-distinguishable.

Type 1	Type 2	How to define them as content-distinguishable
item	item	<p>The first component or partition is either:</p> <ul style="list-style-type: none"> An item and marked as initiator distinguishable and <p>Both items are different partitions of the same partitioned subtree, or The second type has an initiator and those initiators are mutually exclusive, or Both types have the same initiator and the value of the first item is distinguishable from the value of the second item, or The second type has no initiator and the initiator of the first type is distinguishable from the value of the second type.</p> <ul style="list-style-type: none"> An item and not marked as initiator distinguishable and <p>Both items are different partitions of the same partitioned subtree, or The first item has an initiator, second has an initiator and initiator value of first is distinguishable from initiator value of the second. Both types have initiators and they are the same, or both types have no initiator and the value of the first item is distinguishable from the value of the second item. The first item has an initiator, second does not, and value of initiator distinguishable from the value of second item. The first item has no initiator, second does, and the first item's value is distinguishable from the value of the second item's initiator.</p>
item	sequence group, choice group, or unordered group	<p>The first component or partition is either:</p> <ul style="list-style-type: none"> An item marked as initiator distinguishable and <p>The second type has an initiator and those initiators are mutually exclusive, or The second type has no initiator and the item is initiator-distinguishable from each type in the starting component set of the group.</p> <ul style="list-style-type: none"> An item not marked as initiator distinguishable and <p>The item and group both have an initiator and the initiator of the item is distinguishable from the initiator of the group, or The item has no initiator, the group has an initiator, and the item's value is distinguishable from the value of group's initiator, or Both types have initiators and they are the same or both types have no initiator, and the item's value is distinguishable from the type of each component in the starting component set of the group, or The item has an initiator, the group has no initiator, and the item is content-distinguishable from the type of each component in the starting component set of the group, or The item has no initiator, the group has an initiator, and the item's value is distinguishable from each type in the starting component set of the group.</p>
item	partitioned item or partitioned group	The first component or partition is content-distinguishable from each partition of the second type.
partitioned item	item	Each partition of the first type is content-distinguishable from the second type.
partitioned item	sequence group	Each partition of the first type is content-distinguishable from the second type.
partitioned item	choice group	Each partition of the first type is content-distinguishable from second type.
partitioned item	unordered group	Each partition of the first type is content-distinguishable from second type.
partitioned item	item	Each partition of the first type is content-distinguishable from second type.
partitioned item	partitioned group	Each partition of the first type is content-distinguishable from second type.
group	item	Each partition of the partitioned group is content-distinguishable from the item.
partitioned group	partitioned item	Each partition of the partitioned group is content-distinguishable from each partition of the item.
partitioned group	sequence group	Each partition of the partitioned group is content-distinguishable from the second group.
partitioned group	partitioned group	Each partition of the first partitioned group is content-distinguishable from each partition of the second partitioned group.
partitioned group	choice group	Each partition of the partitioned group is content-distinguishable from the choice group.
partitioned group	unordered group	Each partition of the partitioned group is content-distinguishable from the unordered group.

Type 1	Type 2	How to define them as content-distinguishable
sequence group	item	<p>The first component is either:</p> <ul style="list-style-type: none"> • A group marked as initiator distinguishable and <p>The item has an initiator and the initiator value of the group is distinguishable from the initiator value of the item, or The second type has no initiator and the initiator value of the first type is distinguishable from the value of the second type.</p> <ul style="list-style-type: none"> • A group not marked as initiator distinguishable and <p>The group has an initiator, the item has an initiator and the initiator value of the group is distinguishable from initiator value of the item, or The group has an initiator, the item does not, and value of the initiator is distinguishable from value of the item, or Both types have initiators and they are the same, or both types have no initiator and each type in the starting component set of the group is content-distinguishable from the value of the second item.</p>
sequence group	sequence group	<p>The first component is either:</p> <ul style="list-style-type: none"> • A group marked as initiator distinguishable, and <p>Both groups are different partitions of the same partitioned subtree, or The second type has an initiator and those initiators are mutually exclusive, or The second type has no initiator and the first group is initiator-distinguishable from each type in the starting component set of the second group.</p> <ul style="list-style-type: none"> • A group not marked as initiator distinguishable, and <p>Both groups are different partitions of the same partitioned subtree, or Both groups have an initiator and the initiator value of the first group is distinguishable from the initiator value of the second group, or The first group has no initiator, the second group has an initiator and the type of each component in the starting component set of the first group is content-distinguishable from the second group, or The first group has an initiator, the second group has no initiator and the first group is content-distinguishable from the type of each component in the starting component set of the group. Both types have initiators and they are the same or both types have no initiator and The first group is content-distinguishable from the starting component set of the second group, or The second group is content-distinguishable from the starting component set of the first group, or The starting component set of the first group is content-distinguishable from the starting component set of the second group.</p>
sequence group	choice group or group	<p>The first component is either:</p> <ul style="list-style-type: none"> • A group marked as initiator distinguishable and <p>Both groups are different partitions of the same partitioned subtree, or The second type has an initiator and those initiators are mutually exclusive, or The second group has no initiator and the first group is initiator-distinguishable from the type of each component of the second group.</p> <ul style="list-style-type: none"> • A group not marked as initiator distinguishable and <p>Both groups are different partitions of the same partitioned subtree, or Both groups have an initiator and the initiator of the first group is distinguishable from the initiator of the second group, or The first group has an initiator, the second group has no initiator and the first group is content-distinguishable from the type of each component of the second group, or The first group has no initiator the second group has an initiator, and each component in the starting component set of the first group is distinguishable from the second group, or Both types have initiators, either the initiators are the same or both types have no initiator, and each component in the starting component set of the first group is content-distinguishable from the type of each component of the second group.</p>
sequence group	partitioned item or partitioned group	<p>The first component is either:</p> <ul style="list-style-type: none"> • A group marked as initiator-distinguishable and the group is initiator-distinguishable from each partition of the second type, or • A group not marked as initiator-distinguishable and the group is content-distinguishable from each partition of the second type.
choice group or unordered group	item	<p>The first component is either:</p> <ul style="list-style-type: none"> • A group marked as initiator distinguishable and The item has an initiator and the initiator value of the group is distinguishable from the initiator value of the item, or The second type has no initiator and the initiator value of the first type is distinguishable from the value of the second type. • A group not marked as initiator distinguishable and Both types have an initiator and the initiator value of the group is distinguishable from initiator value of the item, or The group has an initiator, the item does not, and the value of the initiator is distinguishable from value of the item, or The group has no initiator, the item has an initiator and the starting component set of the group is content-distinguishable from the item, or Both types have initiators and they are the same, both types have no initiator and the type of each component of the group is distinguishable from the value of the second item.

Type 1	Type 2	How to define them as content-distinguishable
choice group or group	sequence group	<p>The first component is either:</p> <ul style="list-style-type: none"> • A group marked as initiator distinguishable and <p>Both groups are different partitions of the same partitioned subtree, or The second type has an initiator and those initiators are mutually exclusive, or The second type has no initiator and the first group is initiator-distinguishable from each type in the starting set of the second group.</p> <ul style="list-style-type: none"> • A group not marked as initiator distinguishable, and <p>Both groups are different partitions of the same partitioned subtree, or Both groups have an initiator and the initiator value of the first group is distinguishable from the initiator value of the second group, or The first group has an initiator, the second group has no initiator and the first group is content-distinguishable from the type of each component in the starting component set of the second group. The first group has no initiator, the second group has an initiator and the type of each component of the first group is content-distinguishable from the second group, or. Both types have initiators and they are the same or both types have no initiator, and the type of each component of the first group is content-distinguishable from the type of each component in the starting component set of the second group.</p>
choice group or group	choice group or group	<p>The first component is either:</p> <ul style="list-style-type: none"> • A group marked as initiator distinguishable and <p>Both groups are different partitions of the same partitioned subtree, or The second type has an initiator and those initiators are mutually exclusive, or The second group has no initiator and the first group is initiator-distinguishable from the type of each component of the second group.</p> <ul style="list-style-type: none"> • A group not marked as initiator distinguishable and <p>Both groups are different partitions of the same partitioned subtree, or Both groups have an initiator and the initiator of the first group is distinguishable from the initiator of the second group, or The first group has an initiator, the second group has no initiator and the first group is content-distinguishable from the type of each component of the second group, or The first group has no initiator the second group has an initiator, and each component of the first group is distinguishable from the second group, or Both types have initiators and the initiators are the same or both types have no initiator, and the type of each component of the first group is content-distinguishable from the type of each component of the second group.</p>
choice group or group	partitioned item or partitioned group	<p>The first component is either:</p> <ul style="list-style-type: none"> • A group marked as initiator-distinguishable and the group is initiator-distinguishable from each partition of the second type, or • A group not marked as initiator-distinguishable and the group is content-distinguishable from each partition of the second type.

Ending-distinguishable types

Ending-distinguishability is used to determine if the end of a data object is distinguishable from the start of any other data object that could be next in the data.

The following table describes how two types may be ending-distinguishable. This is helpful if you are validating data and you receive a message that says a type exists, but it belongs to the wrong component.

Type1	Type2	How to define them as ending-distinguishable
item	item or sequence group or choice group or unordered group	Type1 is bound or the value of Type1 is content-distinguishable from Type2.
item	partitioned item or partitioned group	Type1 is bound or the value of Type1 is content-distinguishable from each partition of Type2.
sequence group	item or sequence group or choice group or unordered group	<p>Either:</p> <ul style="list-style-type: none"> • Type1 is bound. <p>or</p> <p>For each component in the unbound set of Type1</p> <ul style="list-style-type: none"> • If the component has a fixed range it must be ending-distinguishable from Type2. • If the component range is variable, it must be content-distinguishable from Type2 and ending-distinguishable from Type2.
sequence group	partitioned item or partitioned group	<p>Either:</p> <ul style="list-style-type: none"> • Type1 is bound. <p>or</p> <p>For each component in the unbound set of Type1</p> <ul style="list-style-type: none"> • If the component has a fixed range it must be ending-distinguishable from each partition of Type2. • If the component range is variable, it must be content-distinguishable from each partition of Type2 and ending-distinguishable from each partition of Type2.

Type1	Type2	How to define them as ending-distinguishable
choice group or unordered group	item or sequence group or choice group or unordered group	<p>Either:</p> <ul style="list-style-type: none"> • Type1 is bound. <p>or</p> <ul style="list-style-type: none"> • For each component of Type1, <p>The type of that component is bound, or The type of that component is unbound, and that type must be ending-distinguishable from Type2.</p>
choice group or unordered group	partitioned item or partitioned group	<p>Either:</p> <ul style="list-style-type: none"> • Type1 is bound. <p>or</p> <ul style="list-style-type: none"> • For each component of Type1, <p>The type of that component is bound, or The type of that component is unbound, and that type must be ending-distinguishable from each partition of Type2.</p>
partitioned item or partitioned group	item, sequence group, choice group, or unordered group	<p>For each partition of Type1, either:</p> <ul style="list-style-type: none"> • The partition is a partitioned type and ending-distinguishable from the second type, or • The partition of the first type is bound, or • The partition of the first type is unbound and ending-distinguishable from the second type.
partitioned item or partitioned group	partitioned item or partitioned group	<p>For each partition of the first type, either:</p> <ul style="list-style-type: none"> • The partition is a partitioned type and ending-distinguishable from the second type, or • The partition of the first type is bound, or • The partition of the first type is unbound and ending-distinguishable from the second type.

Distinguishable data objects of an implicit group

When a data object belongs to an implicit sequence, determining the component a data object belongs to depends on the format of the sequence, the type definition of a component, and the component properties.

- [Guidelines for defining an implicit delimited sequence](#)
- [Guidelines for defining an implicit sequence that has no delimiter](#)
- [Guidelines for defining an implicit unordered group that is delimited](#)
- [Guidelines for defining an implicit unordered group that has no delimiter](#)

Guidelines for defining an implicit delimited sequence

- If the type of a component has a terminator, that terminator must be different from the delimiter of the explicit group. If the delimiter and terminator are both defined as literal values, a type analysis confirms both have different values.
- The type of a component cannot be a binary item whose length is not fixed or sized.
- If the type of a component or a contained component is not bound, the type of the component cannot have a delimiter that is the same as the delimiter of the implicit sequence.
- If the range of a component is not bound, the type of that component must be content-distinguishable from the type of each component in the following set of that component.
- If the type of a component has both an initiator and terminator, the nested delimiters do not have restrictions. If this is not the case, all contained delimiters must be different.

Guidelines for defining an implicit sequence that has no delimiter

- The type of a component cannot be a binary item whose length is not fixed or sized.
- If the range of a component is bound, but the type of the component is unbound, the type must be ending-distinguishable from each type in the component's following set.
- If the range of a component is not bound, all the following rules apply:
 - The type of the component must be content-distinguishable from each type in the component's following set.
 - If the maximum range for that component is greater than one, and the type is unbound, the type must be ending-distinguishable from itself.
 - If the type of the component is unbound, the type must be end-distinguishable from each type in the component's following set.

Guidelines for defining an implicit unordered group that is delimited

- The type of a component cannot be a binary item whose length is not fixed.

- The type of a component must be content-distinguishable from the type of each other component.
- If the type of a component has both an initiator and terminator, the nested delimiters do not have restrictions. If this is not the case, all contained delimiters must be different.

Guidelines for defining an implicit unordered group that has no delimiter

- The type of a component cannot be a binary item whose length is not fixed.
- The type of a component must be content-distinguishable from the type of each other component.
- If the type of a component is unbound, the type must be ending-distinguishable from the type of each other component.
- If the maximum range is greater than one and the type is unbound, the type must be ending-distinguishable from itself.

Distinguishable data objects of an explicit group

When a data object belongs to an unordered group, distinguishing one data object from another depends on the format of the unordered group and the type definition of each component.

- [Guidelines for defining an explicit fixed group](#)
- [Guidelines for defining an explicit delimited group](#)
- [Objects of a choice group](#)
- [Objects of a partitioned type](#)

Guidelines for defining an explicit fixed group

- In a fixed group, each component must be either an item of fixed size, an item padded to a fixed length, or an explicit fixed group.
- The range maximum for each component must have a specific value; it cannot be **s**. A fixed amount of space is assumed in the data stream for each series member of a component in a fixed group.

Guidelines for defining an explicit delimited group

A component of an explicit delimited group can be a group or item, with the following restrictions:

- If the type of a component has a terminator, that terminator must be different from the delimiter of the explicit group. If the delimiter and terminator are both defined as a literal, analysis confirms if both have different values.
- If a component of an explicit delimited group is a binary item whose length is not fixed, that component must be sized by the component that precedes it.
- If the type of a component, or a contained component, does not have a terminator and the type of the component has a delimiter that is the same as the delimiter of the explicit group, the type of the component *must* be an explicit sequence. In this case, the delimiter appears as a placeholder for every data object if data of the same delimiter follows it.
- If an explicit group has a delimiter, the range maximum of any component other than the last one must have a specific value; it cannot be **s**. A delimiter is assumed in the data stream for each series member of a component of an explicit group with a delimiter.

Objects of a choice group

When a data object belongs to a choice group, which component the data object belongs to is based on the order the components appear in the type definition.

In a choice group, if a component is unbound it must be content-distinguishable from the type of each component that follows it in the component list.

Objects of a partitioned type

When data objects of different types appear in the same place in the data, the types must be distinguishable. The data must be distinguishable in the type definition. When a partition is part of a partitioned object, the process of cycling through the partitioned subtree begins to make the determination of which partition the data object belongs to.

In a partitioned type, each partition must be content-distinguishable from the type of each partition that follows it.

Distinguishable syntax objects

The system must be able to distinguish between different syntax objects contained within a group. To ensure that the syntax objects are distinguishable, use the following guidelines:

- If a component is unbound, make sure its delimiter and any delimiter contained in it is distinguishable from the delimiter of the group. If you get an analysis error, look for missing placeholders or components with a range maximum of **s**.

In a component or a contained component with the same delimiter as the type delimiter, the system assumes that the delimiter appears as a placeholder for every data object if data with the same delimiter follows it.

- If the delimited group has a terminator, make sure that the delimiter and any contained delimiter is distinguishable from the terminator.
The schema analysis verifies if the above requirement is met in a non-partitioned group.

Return codes and error messages

- [Type Tree Analyzer errors and warnings](#)
- [Schema analysis logic error messages](#)
- [Logic error and warning messages](#)
- [Schema analysis structure error messages](#)
- [Schema analysis structure warning messages](#)

Type Tree Analyzer errors and warnings

The Type Tree Analyzer checks the logic and internal consistency of your data definitions. Schema analyzer error and warnings messages are issued on the type of analysis that is performed: logic and structural.

- Logical analysis addresses the integrity of the relationships that you define in the schema.
- Structural analysis addresses the integrity of the underlying database.

Warnings indicate a successful analysis and are relatively insignificant. Warning messages provide information about inconsistencies that occurred, when the schema was changed, and were automatically resolved.

Errors are important. Error messages provide information about errors in your type definitions that you should correct. An error may result in unpredictable results in your mapping.

Words in italics represent information that varies, indicating information specific to the type for which the message is generated.

Schema analysis logic error messages

The following table lists the logic error messages that result from a logical analysis of a schema:

Return Code	Message
L100	COMPONENT neither inherited nor local: ` <i>type name</i> ` of TYPE: ` <i>type name</i> ` Hint: Look at the super-type's component list. The component is a valid type, but the supertype has a component list that restricts you from using this type as a component. You may have added subtype components before adding supertype components. Either remove all supertype components or add the components in error to the component list of the supertype.
L101	This GROUP must have at least one component - TYPE: ` <i>type name</i> ` Hint: If you want to map this group, add components. If you do not want to map it, make it a category.
L102	Circular reference found in COMPONENT # (` <i>type name</i> `) - TYPE: ` <i>type name</i> ' Hint: Look at the type of the component in error. It is probably missing an initiator or terminator.
L103	Circular reference found in Floating Component type - TYPE: ` <i>type name</i> ' Hint: Look at the floating component type in error. It is probably missing an initiator or terminator.
L104	DELIMITER for TYPE - ` <i>type name</i> ` must have a value Hint: All delimited groups need a delimiter. Edit delimited group properties to insert the missing delimiter.
L105	DELIMITER type neither inherited nor local - TYPE: ` <i>type name</i> ' Hint: The delimiter name has been entered incorrectly. It should be the name of a local type, or the name of an inherited delimiter, or the name of a type in the subtree of the inherited delimiter.
L106	Default DELIMITER not specified - TYPE: ` <i>type name</i> ' Hint: This Type was specified with a FIND option for its delimiter. Please add a default value to define what to use for building outputs.
L107	Default DELIMITER not in restriction list - TYPE: ` <i>type name</i> ' Hint: This delimiter was specified as a syntax item. Add the default value to the restriction list for that syntax item.
L108	DELIMITER type is not a SYNTAX ITEM - TYPE: ` <i>type name</i> ' Hint: Delimiters specified as an item must be specified to be interpreted as SYNTAX to set the value of the delimiter if it appears as a component in a data stream.

- L109
 DELIMITER type has no restriction list - TYPE: 'type name'
 Hint: All syntax items need a restriction list.
- L110
 RELEASE CHARACTER neither inherited nor local - TYPE: 'type name'
 Hint: The release character name has been entered incorrectly. It should be either the name of a local type, the name of an inherited release character, or the name of a type in the sub-tree of the inherited release character.
- L111
 Default RELEASE CHARACTER not specified - TYPE: 'type name'
 Hint: This Type was specified with a syntax item for its release character. Please add a default value to define a value for the release character that has not been encountered in the data.
- L112
 Default RELEASE CHARACTER not in restriction list - TYPE: 'type name'
 Hint: This Type was specified with a syntax item for its release character. Please add the default value to the restriction list of that syntax item.
- L113
 RELEASE CHARACTER type is not a SYNTAX ITEM - TYPE: 'type name'
 Hint: Release characters specified as an item must be specified to be interpreted as SYNTAX to set the value of the release character if it appears as a component in a data stream.
- L114
 RELEASE CHARACTER type has no restriction list - TYPE: 'type name'
 Hint: All syntax items need a restriction list.
- L115
 Floating Component TYPE neither inherited nor local - TYPE: 'type name'
 Hint: The floating component name has been entered incorrectly. It should be either the name of a local type, the name of an inherited floating component, or the name of a type in the sub-tree of the inherited floating component.
- L116
 INITIATOR type neither inherited nor local - TYPE: 'type name'
 Hint: The initiator name has been entered incorrectly. It should be either the name of a local type, the name of an inherited initiator, or the name of a type in the sub-tree of the inherited initiator.
- L117
 Default INITIATOR not specified - TYPE: 'type name'
 Hint: This Type was specified with a syntax item for its initiator. Add a default value to define a value for that initiator has not been encountered in the data.
- L118
 Default INITIATOR not in restriction list - TYPE: 'type name'
 Hint: This Type was specified with a syntax item for its initiator. Add the default value to the restriction list of that syntax item.
- L119
 INITIATOR type is not a SYNTAX ITEM - TYPE: 'type name'
 Hint: Initiators specified as an item must be specified to be interpreted as SYNTAX to set the value of the initiator if it appears as a component in a data stream.
- L120
 INITIATOR type has no restriction list - TYPE: 'type name'
 Hint: All syntax items need a restriction list.
- L121
 TERMINATOR type neither inherited nor local - TYPE: 'type name'
 Hint: The terminator name has been entered incorrectly. It should be either the name of a local type, the name of an inherited terminator, or the name of a type in the sub-tree of the inherited terminator.
- L122
 Default TERMINATOR not specified - TYPE: 'type name'
 Hint: This Type was specified with a syntax item for its terminator. Add a default value to define a value for that terminator has not been encountered in the data.
- L123
 Default TERMINATOR not in restriction list - TYPE: 'type name'
 Hint: This Type was specified with a syntax item for its terminator. Please add the default value to the restriction list of that syntax item.
- L124
 TERMINATOR type is not a SYNTAX ITEM - TYPE: 'type name'
 Hint: Terminators specified as an item must be specified to be interpreted as SYNTAX to set the value of the terminator if it appears as a component in a data stream.
- L125
 TERMINATOR type has no restriction list - TYPE: 'type name'
 Hint: All syntax items need a restriction list.
- L126
 COMPONENT range minimum (#) greater than range maximum (#) - COMPONENT 'type name' - TYPE: 'type name'
 Hint: The minimum range must be less than or equal to the maximum range.
- L127
 COMPONENT range minimum (#) less than inherited range minimum(#) - COMPONENT 'type name' - TYPE: 'type name'
 Hint: The component in error has been inherited. Look at the range of the component with the same name in the super-type's component list.
- L128

COMPONENT range maximum (#) greater than inherited range maximum(#)- COMPONENT `type name` - TYPE: `type name`
Hint: The component in error has been inherited. Look at the range of the component with the same name in the super-type's component list.

L129

COMPONENT RULE references a COMPONENT later in the component list - `type name` - TYPE: `type name`
Hint: Move the component rule to the component later in the list.

L130

COMPONENT RULE references undefined type - COMPONENT # of TYPE: `type name`
Hint: Verify the spelling of the data object name. The rule should reference a data object name of the component or a data object name of a component earlier in the component list.

L131

COMPONENT RULE references components of a partitioned group - COMPONENT # of TYPE type name

L132

Invalid partitioning: TYPE has no SUBTYPES - TYPE: `type name`
Hint: Remove the partitioned option from the class window or add sub-types to the Type in error.

L133

Type of COMPONENT exists, but its relative name is not valid: `type name` in TYPE: `type name`
Hint: To get the correct relative name, drag the type you want to use as a component and drop it in the component list of the Type. (Remember to delete the invalid component!)

L134

Reference to 'ANY' not allowed: COMPONENT number # of TYPE: `type name`
Hint: In this case, the Type in error is a group and it is not the root of a partitioned tree. ANY cannot be used if that component needs to be validated. So, if that group is partitioned, you cannot use ANY for a component up to and including the identifier (if there is one). If that group is not partitioned, you cannot use ANY at all.

L135

COMPONENT number # cannot reference a CATEGORY in TYPE: `type name` (because group is not partitioned)
Hint: In this case, the Type in error is a group and it's not the root of a partitioned tree. A category cannot be used if the component must be validated. So, if that group is partitioned, you cannot use a category for a component up to and including the identifier (if there is one). If that group is not partitioned, you cannot use a category as a component at all.

L136

COMPONENT `type name` occurs more than once in list - TYPE: `type name`
Hint: Each component in the same component list must have a unique type name. Try to make sub-types of the type name in error and replace each non-unique component with one of the new sub-types.

L137

COMPONENT `type name` and its super-type cannot be in same COMPONENT LIST (in TYPE: `type name`)
Hint: Try making another sub-type of the super-type and replace the super-type reference with the new sub-type.

L138

COMPONENT `type name` is same type as delimiter - TYPE: `type name`
Hint: A component and a delimiter cannot have the same name. You may need to add sub-types to the type name used in error to resolve this one.

L139

COMPONENT `type name` is sub-type of delimiter - TYPE: `type name`
Hint: This occurs when a syntax item is used to specify a delimiter. You can add another sub-type to the syntax item and replace the delimiter name with the new sub-type name.

L140

COMPONENT `type name` is super-type of delimiter - TYPE: `type name`
Hint: This occurs when a syntax item is used to specify a delimiter. You can add another sub-type to the syntax item and replace the component name with the new sub-type name.

L141

COMPONENT `type name` is same type as initiator - TYPE: `type name`
Hint: A component and an initiator cannot have the same name. You can add sub-types to the type name used in error and replace both the component name and the initiator name.

L142

COMPONENT `type name` is sub-type of initiator - TYPE: `type name`
Hint: This occurs when a syntax item is used to specify an initiator. You can add another sub-type to the syntax item and replace the initiator name with the new sub-type name.

L143

COMPONENT `type name` is super-type of initiator - TYPE: `type name`
Hint: This occurs when a syntax item is used to specify an initiator. You can add another sub-type to the syntax item and replace the component name with the new sub-type name.

L144

COMPONENT `type name` is same type as terminator - TYPE: `type name`
Hint: A component and a terminator cannot have the same name. Try adding sub-types to the type name used in error and replace both the component name and the terminator name with one of the new sub-types.

L145

COMPONENT `type name` is sub-type of terminator - TYPE: `type name`
Hint: This occurs when a syntax item is used to specify a terminator. You can add another sub-type to the syntax item and replace the terminator name with the new sub-type name.

L146

COMPONENT '*type name*' is super-type of terminator - TYPE: '*type name*'
Hint: This occurs when a syntax item is used to specify a terminator. You can add another sub-type to the syntax item and replace the component name with the new sub-type name.

L147
COMPONENT '*type name*' is same type as Floating Component - TYPE: '*type name*'
Hint: Make both the floating component name and the component name sub-types of the floating component.

L148
COMPONENT '*type name*' is sub-type of Floating Component - TYPE: '*type name*'
Hint: Make both the floating component name and the component name sub-types of the floating component.

L149
COMPONENT '*type name*' is super-type of Floating Component - TYPE: '*type name*'
Hint: Make both the floating component name and the component name sub-types of the floating component.

L150
COMPONENT '*type name*' is same type as release character - TYPE: '*type name*'
Hint: This occurs when a syntax item is used to specify a release character. You can add sub-types to the syntax item and replace both the component name and the release character name with the new sub-type names.

L151
COMPONENT '*type name*' is sub-type of release character - TYPE: '*type name*'
Hint: This occurs when a syntax item is used to specify a release character. You can add another sub-type to the syntax item and replace the release character name with the new sub-type name.

L152
COMPONENT '*type name*' is super-type of release character - TYPE: '*type name*'
Hint: This occurs when a syntax item is used to specify a release character. You can add sub-types to the syntax item and replace the component name with the new sub-type name.

L153
DELIMITER '*type name*' is same type as initiator - TYPE: '*type name*'
Hint: A delimiter and an initiator cannot have the same name. You may need to add sub-types to the type name used in error and replace both the delimiter and initiator names to refer to the new sub-types.

L154
DELIMITER '*type name*' is sub-type of initiator - TYPE: '*type name*'
Hint: This occurs when a syntax item is used to specify both an initiator and a delimiter. You can add another sub-type to the syntax item and replace the initiator name with the new sub-type name.

L155
DELIMITER '*type name*' is super-type of initiator - TYPE: '*type name*'
Hint: This occurs when a syntax item is used to specify both an initiator and a delimiter. You can add another sub-type to the syntax item and replace the delimiter name with the new sub-type name.

L156
DELIMITER '*type name*' is same type as terminator - TYPE: '*type name*'
Hint: A delimiter and a terminator cannot have the same name. You may need to add sub-types to the type name used in error and replace both the delimiter and terminator names to refer to the new sub-types.

L157
DELIMITER '*type name*' is sub-type of terminator - TYPE: '*type name*'
Hint: This occurs when a syntax item is used to specify both a delimiter and a terminator. You can add another sub-type to the syntax item and replace the terminator name with the new sub-type name.

L158
DELIMITER '*type name*' is super-type of terminator - TYPE: '*type name*'
Hint: This occurs when a syntax item is used to specify both a delimiter and a terminator. You can add another sub-type to the syntax item and replace the delimiter name with the new sub-type name.

L159
DELIMITER '*type name*' is same type as release character - TYPE: '*type name*'
Hint: A delimiter and a release character cannot have the same name. You may need to add sub-types to the type name used in error and replace both the delimiter and release character names to refer to the new sub-types.

L160
DELIMITER '*type name*' is sub-type of release character - TYPE: '*type name*'
Hint: This occurs when a syntax item is used to specify both a delimiter and a release character. You can add another sub-type to the syntax item and replace the release character name with the new sub-type name.

L161
DELIMITER '*type name*' is super-type of release character - TYPE: '*type name*'
Hint: This occurs when a syntax item is used to specify both a delimiter and a release character. You can add another sub-type to the syntax item and replace the delimiter name with the new sub-type name.

L162
DELIMITER '*type name*' is same type as Floating Component - TYPE: '*type name*'
Hint: A delimiter and a floating component cannot have the same name. Try adding sub-types to the type name used in error and replace both the delimiter and floating component names to refer to the new sub-types.

L163
DELIMITER '*type name*' is sub-type of Floating Component - TYPE: '*type name*'

Hint: This occurs when a syntax item is used to specify both a delimiter and a floating component. You can add another sub-type to the syntax item and replace the floating component name with the new sub-type name.

L164

DELIMITER '*type name*' is super-type of Floating Component - TYPE: '*type name*'

Hint: This occurs when a syntax item is used to specify both a delimiter and a floating component. You can add another sub-type to the syntax item and replace the delimiter name with the new sub-type name.

L165

INITIATOR '*type name*' is same type as terminator - TYPE: '*type name*'

Hint: An initiator and a terminator cannot have the same name. You may need to add sub-types to the type name used in error and replace both the initiator and terminator names to refer to the new sub-types.

L166

INITIATOR '*type name*' is sub-type of terminator - TYPE: '*type name*'

Hint: This occurs when a syntax item is used to specify both an initiator and a terminator. You can add another sub-type to the syntax item and replace the terminator name with the new sub-type name.

L167

INITIATOR '*type name*' is super-type of terminator - TYPE: '*type name*'

Hint: This occurs when a syntax item is used to specify both an initiator and a terminator. You can add another sub-type to the syntax item and replace the initiator name with the new sub-type name.

L168

INITIATOR '*type name*' is same type as release character - TYPE: '*type name*'

Hint: This occurs when a syntax item is used to specify both an initiator and a release character. You can add sub-types to the syntax item and replace both the initiator name and the release character name with a new sub-type name.

L169

INITIATOR '*type name*' is sub-type of release character - TYPE: '*type name*'

Hint: This occurs when a syntax item is used to specify both an initiator and a release character. You can add another sub-type to the syntax item and replace the release character name with the new sub-type name.

L170

INITIATOR '*type name*' is super-type of release character - TYPE: '*type name*'

Hint: This occurs when a syntax item is used to specify both an initiator and a release character. You can add another sub-type to the syntax item and replace the initiator name with the new sub-type name.

L171

INITIATOR '*type name*' is same type as Floating Component - TYPE: '*type name*'

Hint: An initiator and a floating component cannot have the same name. Try adding sub-types to the type name used in error and replace both the initiator and floating component names with the new sub-types.

L172

INITIATOR '*type name*' is sub-type of Floating Component - TYPE: '*type name*'

Hint: This occurs when a syntax item is used to specify both an initiator and a floating component. You can add another sub-type to the syntax item and replace the floating component name with the new sub-type name.

L173

INITIATOR '*type name*' is super-type of Floating Component - TYPE: '*type name*'

Hint: This occurs when a syntax item is used to specify both an initiator and a floating component. You can add another sub-type to the syntax item and replace the initiator name with the new sub-type name.

L174

TERMINATOR '*type name*' is same type as release character - TYPE: '*type name*'

Hint: A terminator and a release character cannot have the same name. You may need to add sub-types to the type name used in error and replace both the terminator and release character names with the new sub-types.

L175

TERMINATOR '*type name*' is sub-type of release character - TYPE: '*type name*'

Hint: This occurs when a syntax item is used to specify both a terminator and a release character. You can add another sub-type to the syntax item and replace the release character name with the new sub-type name.

L176

TERMINATOR '*type name*' is super-type of release character - TYPE: '*type name*'

Hint: This occurs when a syntax item is used to specify both a terminator and a release character. You can add another sub-type to the syntax item and replace the terminator name with the new sub-type name.

L177

TERMINATOR '*type name*' is same type as Floating Component - TYPE: '*type name*'

Hint: A terminator and a floating component cannot have the same name. You may need to add sub-types to the type name used in error and replace both the terminator and floating component names with the new sub-types.

L178

TERMINATOR '*type name*' is sub-type of Floating Component - TYPE: '*type name*'

Hint: This occurs when a syntax item is used to specify both a terminator and a floating component. You can add another sub-type to the syntax item and replace the floating component name with the new sub-type name.

L179

TERMINATOR '*type name*' is super-type of Floating Component - TYPE: '*type name*'

Hint: This occurs when a syntax item is used to specify both a terminator and a floating component. You can add another sub-type to the syntax item and replace the terminator name with the new sub-type name.

L180

RELEASE CHARACTER `type name' is same type as Floating Component - TYPE: `type name'

Hint: A release character and a floating component cannot have the same name. You may need to add sub-types to the type name used in error and replace both the release character and floating component names to refer to the new sub-types.

L181

RELEASE CHARACTER `type name' is sub-type of Floating Component - TYPE: `type name'

Hint: This occurs when a syntax item is used to specify both a release character and a floating component. You can add another sub-type to the syntax item and replace the floating component name with the new sub-type name.

L182

RELEASE CHARACTER `type name' is super-type of Floating Component - TYPE: `type name'

Hint: This occurs when a syntax item is used to specify both a release character and a floating component. You can add another sub-type to the syntax item and replace the release character name with the new sub-type name.

L183

COMPONENT NAME ambiguous: `type name' in TYPE: `type name'

Hint: This type has a component whose relative name can be associated with more than one type in the schema. Rename the conflicting types.

L184

RESTRICTION longer than max TYPE size - RESTRICTION # of TYPE: `type name'

Hint: The Type in error is an item. Either change the maximum size of the item or remove the restriction.

L185

RESTRICTION used in an earlier partition - RESTRICTION # of TYPE: `type name'

Hint: Item Partitions must have mutually exclusive restrictions. Remove the restriction from one of the partition restriction lists.

L186

Type of COMPONENT does not exist - `type name' in TYPE: `type name'

Hint: You probably entered an incorrect type name. Try the drag and drop approach to get the correct one.

L187

TYPE must be partitioned (since in a partitioned tree and has sub-types) - TYPE: `type name'

Hint: All types in a partitioned sub-type must have mutually exclusive data objects. Set the partitioned property for the type in error.

L188

TYPE is FIXED, COMPONENT # must have a maximum range value - TYPE type name

L189

TYPE is FIXED, but COMPONENT # is not fixed - TYPE: `type name'

Hint: If the component is not intended to be fixed in size, change the group format for the Type to implicit. If the group format is intended to be fixed, check the component: if that component is an item, make sure it has a Padded To length; if that component is a group, change its type to be of fixed syntax.

L190

BINARY text ITEM used as COMPONENT neither FIXED nor SIZED - COMPONENT # of TYPE: `type name'

Hint: The size of a binary text item must either have a Padded To length or it must be sized by the previous component.

L191

COMPONENT with SIZED attribute is not an UNSIGNED INTEGER ITEM TYPE - COMPONENT # of TYPE: `type name'

Hint: A component used to size the component that follows it must be defined as an unsigned integer item type.

L192

The last COMPONENT in the COMPONENT LIST may not have a SIZED attribute: TYPE: `type name'

Hint: Specify a component to follow the one with the sized attribute.

L193

Range of COMPONENT # must have a maximum value to indicate how many placeholders are needed for its series in TYPE: `type name'.

Hint: Change the range maximum to a specific value (not "s") if you may re-define the data this way.

L194

Cannot distinguish delimiter from terminator in TYPE: `type name'.

Hint: Make the range of the last component in the type fixed or make the delimiter of the type different from its terminator.

L195

Cannot distinguish delimiter contained in COMPONENT # from terminator of TYPE: `type name'.

Hint: Make that component bound or make that contained delimiter different from the type terminator.

L196

Cannot distinguish delimiter of COMPONENT # from delimiter of TYPE: type name because COMPONENT # has no placeholder.

Hint: Make that component bound or make that component's delimiter different from the type delimiter.

L197

Cannot distinguish delimiter of COMPONENT # from delimiter of TYPE: type name because COMPONENT # has no range maximum.

Hint: Make that component bound, or make that component's delimiter different from the type delimiter, or specify a range maximum that has a specific value (not "s") for the last component of COMPONENT #.

L198

Cannot distinguish delimiter contained in COMPONENT # from delimiter of TYPE: `type name'.

Hint: Make that contained component bound or make that contained component's delimiter different from the type delimiter.

L200

Cannot distinguish delimiter contained in COMPONENT # from delimiter of TYPE: `type name'.

Hint: Either make that contained component bound, make that contained component's delimiter different from the type delimiter, or specify a range maximum that has a specific value (not "s") for the last component of the contained component.

Logic error and warning messages

The tables in this section list the logic warning messages that result from a logic analysis of a schema.

The following table lists the warnings than can result when a map is compiled.

Warnings should be resolved because they may produce unpredictable results at mapping time.

Return Code	Message
L199	COMPONENT # is not distinguishable from COMPONENT # that may follow in TYPE: `type name'. Hint: Make the first COMPONENT bound, or look at the tables in " Distinguishable objects " to see how you can define the two component types as distinguishable.
L201	Different data objects of COMPONENT # are not distinguishable in TYPE: `type name'. Hint: See " Distinguishable objects " for more information about distinguishable objects.
L202	RESTRICTION list deleted: TYPE is not an ITEM - TYPE: `type name' Hint: Type class was changed from an item to a group or category, so the restriction list was deleted. If this was not your intent, change it back to the way it was.
L203	COMPONENT list deleted: TYPE is an ITEM - TYPE: `type name' Hint: Type class was changed from a group to a item or category, so the program deleted its component list. If this was not your intent, change it back to the way it was.
L204	DELIMITER deleted: TYPE is not a DELIMITED GROUP - TYPE: `type name' Hint: Group format was changed from delimited to something else, so the program deleted its delimiter. If this was not your intent, change it back to the way it was.
L205	COMPONENT RULE deleted: TYPE is a CATEGORY - TYPE: `type name' (warning) Hint: Type class was changed from a group to a category, so its component rule was deleted. If this was not your intent, change it back to the way it was.
L206	DELIMITER cannot be found (because first component is not required) - TYPE: `type name' (warning) Hint: If the delimiter is missing, a previously set initiator value or the default value is used.
L251	COMPONENT NAME could apply to more than one type:'type name' in TYPE: `type name' (warning).

Schema analysis structure error messages

The following table lists the structure error messages that result from a structural analysis of a schema:

Return Code	Message
S100	Invalid TYPE Name: SubTYPE # of TYPE: `type name'
S101	Invalid TYPE chain: SubTYPE # of TYPE: `type name'
S118	Invalid TYPE NAME WhereUsed chain - TYPE NAME: `type name' (error).
S133	Referenced COMPONENT not 'InUse' - COMPONENT # of TYPE: `type name' (error).
S134	COMPONENT previously referenced - COMPONENT # (COMP #) of TYPE: `type name' (error).
S149	Bad Parent COMPONENT Index - COMPONENT `type name' - TYPE: `type name' (error)

Schema analysis structure warning messages

The following table lists the structure warning messages that result from a structural analysis of a schema:

Return Code	Message
S102	Unused DELIMITER deleted: `type name' (at index #)
S103	Invalid DELIMITER pointer deleted - TYPE: `type name'
S104	Invalid default DELIMITER pointer deleted - TYPE: `type name'
S105	Invalid RELEASE Char pointer deleted - TYPE: `type name'

S106 Invalid default RELEASE Char pointer deleted - TYPE: `type name'
S107 Invalid INITIATOR pointer deleted - TYPE: `type name'
S108 Invalid default INITIATOR pointer deleted - TYPE: `type name'
S109 Invalid TERMINATOR pointer deleted - TYPE: `type name'
S110 Invalid default TERMINATOR pointer deleted - TYPE: `type name'
S111 Resetting DELIMITER Use Count (was # now #) - DELIMITER: `type name'
S112 Unused DESCRIPTION deleted: `type name' (at index #)
S113 Invalid DESCRIPTION pointer deleted - TYPE: `type name'
S114 Resetting DESCRIPTION Use Count (was # now #) - DESCRIPTION: `type name'
S115 Invalid Floating Component TYPE pointer deleted - TYPE: `type name'
S116 Invalid TYPE UsedInComp chain repaired - TYPE: `type name'
S117 Unused TYPE NAME deleted - TYPE NAME: `type name' (at index #)
S119 Resetting TYPE NAME use count (was # now #) - TYPE NAME: `type name'
S120 Repaired empty TYPE NAME WhereUsed chain - TYPE NAME: `type name'
S121 Unused RESTRICTION NAME deleted: `type name' (at index #)
S122 Invalid RESTRICTION NAME deleted no DESCRIPTION was available - TYPE: `type name'.
S123 Invalid RESTRICTION NAME deleted DESCRIPTION was `type name` - TYPE: `type name'
S124 Resetting RESTRICTION NAME Use Count (was # now #) - RESTRICTIONS: `type name'
S125 Unused RESTRICTION DESCRIPTION deleted: `type name' (at index #)
S126 Invalid RESTRICTION DESCRIPTION deleted - TYPE: `type name'
S127 Resetting RESTRICTION DESCRIPTION Use Count (was # now #) - RESTRICTIONS: `type name'
S128 Unused RULE deleted: `type name' (at index #)
S129 Invalid RULE pointer deleted - COMPONENT # of TYPE: `type name'
S130 Resetting RULE Use Count (was # now #) - RULE: `type name'
S131 Invalid COMPONENT TYPE Description pointer - COMPONENT #
S132 COMPONENT marked 'InUse' found in Free Chain- COMPONENT #
S135 COMPONENT in Free Chain referenced by a TYPE - COMPONENT #
S136 COMPONENT recovered and added to Free Chain - COMPONENT #
S137 TYPE in Free Chain referenced by another TYPE - TYPE #
S138 TYPE recovered and added to Free Chain - TYPE X'%04X'
S139 TYPE marked 'InUse' but not referenced - TYPE #
S140 Referenced TYPE not marked 'InUse' - TYPE #
S141 TYPE Free Chain not in order: sorting
S142 COMPONENT Free Chain not in order: sorting
S143 Overlap found in LIST Free Chain
S144 Free Chain extends into unallocated region
S145 Overlap found in COMPONENT LIST SPACE: list cleared COMPONENTS will be deleted
S146 Invalid COMPONENT LIST pointer: all COMPONENTS DELETED - TYPE: `type name'
S147 Resetting COUNT in COMPONENT LIST: some COMPONENTS may be lost
S148 RULE truncated (due to internal error): `type name' (at index #)

- S150 CATEGORY 'type name' was missing GROUP and/or ITEM attributes
S151 GROUP 'type name' was missing GROUP attributes
S152 ITEM 'type name' was missing ITEM attributes
-

Maps

A map node invokes a map within a flow.

Map design

Use the Map Designer to develop maps that define input and output specifications and mapping rules for data transformation.

- [Introduction to the Map Designer](#)

- [Auto Map preferences](#)

Use the Auto Map preferences to select the options for the Auto Map to use for mapping source and target objects.

- [Profiler preferences](#)

Use the Profiler preferences to configure the default Map Profiler utility settings.

- [Where Used view](#)

You can see all the locations in a map source (.mms) file where a particular map is called in rules, where a particular type is used in rules, or where the maps and types are used by a rule, in the Where Used view.

- [Search options](#)

The Design Studio supports standard search options.

- [Maps and map source files](#)

- [Input and output cards](#)

- [Card settings](#)

- [Map events](#)

When you run Design Server Maps using the Design Server runtime REST API, the related Map execution events are sent to the specified Elasticsearch server.

- [Map rules](#)

- [Search functionality](#)

The search functionality in the Map designer allows to search for a specific text within all rules in a current executable map and all its corresponding functional maps.

- [Formulating map rules](#)

- [Functional map basics](#)

- [Rename a functional map](#)

The functional map of an execution map can now be renamed. To rename a functional map follow the below procedure:

- [Map settings](#)

- [Configuring bursts](#)

- [Building maps](#)

- [Running a map](#)

- [Map Profiler](#)

The Map Profiler is a user-configurable utility that captures and reports map execution statistics.

- [Map performance](#)

- [Managing invalid data](#)

- [Map audit overview](#)

- [Troubleshooting](#)

- [Error and warning messages](#)

Introduction to the Map Designer

Use the Map Designer to develop maps that define input and output specifications and mapping rules for data transformation.

The Map Designer uses the definitions of data stored in the schemas (that are created using the Type Designer) to specify the transformation logic in the form of map rules. Map rules operate on input data objects and build output data objects. The map can be built for specific platforms, and then run on that platform to perform the transformation of the data.

During the design phase, you build and run maps from within the Map Designer. You can view the results of a map in the Map Designer. You can also use the Map Designer to build a map to be run on supported platforms.

If you have built a map to be run on a WebSphere DataPower SOA Appliance platform, to test it from the Map Designer, you must also install the test domain on the appliance.

If you are testing IBM Transformation Extender maps that you can run on a Sterling B2B Integrator server, or deploy to a Sterling B2B Integrator server map repository, you must first complete the installation and configuration requirements. Install the IBM Transformation Extender for Integration Servers on both the machine where the Design Studio is installed, and on the machine where the Sterling B2B Integrator server is installed. Configure the Design Studio connection properties for IBM Transformation Extender maps in the Sterling B2B Integrator preferences pane.

- [Using the Map Designer](#)

- [Define data objects and properties](#)

- [Create maps to specify sources and targets](#)

Using the Map Designer

To use the Map Designer, you must already have the schemas that define your data. The Map Designer uses the data object definitions that are stored in the schemas.

The Map Designer is used to:

- Create maps to specify the logic necessary to transform the input data to the desired output data.
- Identify the source and data objects of the input data.
- Validate and resolve the source data type properties defined in the Type Designer.
- Identify the target and data objects of the output data.
- Specify and build the output data according to the map rules.
- Provide information about data validation by generating trace files.
- View the run results of the map execution.

After defining data objects and their properties in the Type Designer, you define a map in the Map Designer where map cards specify the input source and the output target.

Define data objects and properties

The schemas are defined with the Type Designer. You can also import them from a project.

To define data:

1. Use the Type Designer or import operation to create or generate a schema.
2. Define the type properties of input and output data.
3. Specify restrictions of input data for item types.
4. Specify desired validation of input data in component rules.
5. Save and analyze the schema.

Create maps to specify sources and targets

After defining data objects and their properties in the Type Designer, you specify the source of the input and the target of the output in the Map Designer. Sources and targets are specified with map cards. Each map can have none or more input cards, and one or more output cards.

To create maps: in the Transformation Extender Development perspective:

1. Create a map source file.
2. Create required executable map(s) in the map source file.
3. Create an input card that represents your input data.
4. Create an output card that represents your desired output.
5. In the output card map rules, specify the logic necessary to transform the input data to the desired output data.
6. Build the map.
7. Run the map.

Auto Map preferences

Use the Auto Map preferences to select the options for the Auto Map to use for mapping source and target objects.

To access Auto Map preferences, in the Map Designer, click Window > Preferences > Transformation Extender > Map > Auto Map.

- **Mapping Scope**
To specify the level of map objects that the Auto Map operation maps, you must configure the mapping scope.
- **Rule Creation Option**
To specify that the Auto Map overwrite existing rules on objects of the map that the Auto Map creates, configure the option for rule creation.
- **Name Matching Options**
To specify the options that the Auto Map uses to match the object names on the source with the object names on the target, configure the Name Matching Options.
- **Mapping Criteria**
To specify the conditions that the Auto Map uses for creating rules, you must configure the mapping criteria.
- **Match percent examples**
Here are examples of the minimum percent amount that the source and target names must match each other for the Auto Map operation to create rules.

Mapping Scope

To specify the level of map objects that the Auto Map operation maps, you must configure the mapping scope.

Configure the mapping scope in the Auto Map preferences in the Map Designer.

Option

Definition

Map only immediate items

Automatically maps the child items in the target. The child items are the items directly under the group that you selected.

Map only immediate items and groups

Automatically maps both the child items and groups in the target. On the groups, though, the process creates only functional maps, and does not map the child items that are under the group.

Map all descendants

Maps all levels of the child items and groups in the target. Creates the functional maps. Maps the children of the functional maps and continues mapping lower levels until there is nothing else for the process to map. This mapping scope option is the default configuration.

Rule Creation Option

To specify that the Auto Map overwrite existing rules on objects of the map that the Auto Map creates, configure the option for rule creation.

Configure the rule creation option in the Auto Map preferences in the Map Designer.

Option

Definition

Overwrite existing rules

The Auto Map operation maps objects that have existing rules. When you check this option, the Auto Map operation overwrites those rules.

Name Matching Options

To specify the options that the Auto Map uses to match the object names on the source with the object names on the target, configure the Name Matching Options.

Configure the Name Matching Options in the Auto Map preferences in the Map Designer.

Option

Definition

Case sensitive

The Auto Map operation compares those source and target object names with the same case.

Alphanumeric characters (letters and digits only)

The Auto Map operation ignores the nonalphanumeric characters when it compares the source and target object names.

Example:

- Name Matching Options: The Alphanumeric characters (letters and digits only) option for matching name is selected.
- Mapping Criteria: The Create rules when the names of the sources and targets match mapping criterion is selected.
- Source: abc_dEf
- Target: abcdEf
- Operation: The Auto Map compare operation ignores the underscore symbol that is in the source object name. It compares the resulting abcdEf characters from the source object name with the abcdEf characters of the target object name.
- Result: There is a 100% match between the resulting abcdEf characters from the source object name and the abcdEf characters of the target object name.

Mapping Criteria

To specify the conditions that the Auto Map uses for creating rules, you must configure the mapping criteria.

Configure the mapping criteria in the Auto Map preferences in the Map Designer.

Option

Definition

Create rules when the names of the sources and targets match

When you click this option, the Auto Map creates rules when the source and target object names match. The match criteria is defined by the Name Matching Options that you selected.

Create rules when the names of the sources and targets match by the specified percent

When you click this option, you must also click the % match arrows to specify a percent. The percent is the degree of match between the source and target object names that the Auto Map operation uses to determine whether it creates rules. The Auto Map operation uses the match criteria as defined by the Name Matching Options that you selected. It creates the rules when the source and target object names match according to those selected matching options and by the percent you specified.

Use the Number of characters field to specify the number of characters to compare. For example, when you specify the number of characters as 5:

- The following is a match:

`Date_424_PO<=>Date_411_PO`

- The following is not a match:

`PO_424_LineItems<=>PO_411_LineItems`

Match percent examples

Here are examples of the minimum percent amount that the source and target names must match each other for the Auto Map operation to create rules.

If you select the option to create rules when the source and target names match by a specified percent, you must specify that percent amount. You can go back to the Select the options to map source to target window and select different percent amounts. Repeat this process until you get the matching results that you want.

The following table lists examples of source and target object names and the percent amount that is used for the match process. The Auto Map operation uses this percent amount as the matching criterion, which determines whether the operation creates the rules. In these examples, all of the names of the source and target objects match each other by at least the specified percent. In example 1, firstName matches fName by a minimum of 15 percent. Since the matching criterion is met, the Auto Map operation can create the rules.

Example	Source object name	Target object name	Match percent
1	firstName	fName	15
2	lastName	lName	25
3	address	addressLine	25
4	account	acct	20
5	country	ctry	20
6	zipCode	zcode	50
7	birthDate	bDate	15
8	event	evt	30
9	telephone	phone	15
10	proj	project	15
11	record	rcrd	15

Profiler preferences

Use the Profiler preferences to configure the default Map Profiler utility settings.

Before profiling a map, you must enable a **Function Times** option or a **Type Times** option to see statistics in the output, otherwise the report is empty.

To access user preferences, click Window > Preferences > Transformation Extender > Map > Profiler.

To run a map using the Map Profiler utility to capture and report map execution statistics, you must configure the profiler settings in the Map Designer for the following profiler options:

Function times

Enable the **All** or **Above** option to view function time statistics in the profiler report.

Option	Description
None	(Default setting) When None is enabled, function time statistics are not reported.
All	When you select All, each function and each function call is provided in the profiler output. Every occurrence of every function is displayed with the depth (level), number of iterations, elapsed time, map name, and type name.
Above	Use the Above option to obtain a smaller output file. When you specify a value in 1/10 milliseconds (10000 = 1 second), function times are only recorded when they exceed that value.
Summary	When you choose the Summary option, the report contains a summary for each function instead of listing every occurrence. A function is listed once with the total number of iterations and total elapsed time for all iterations. Sample report .

Type times

Enable the **All** or **Above** option to view type time statistics in the profiler report.

Option	Description
None	(Default setting) When None is enabled, type time statistics are not reported.
All	When you select All, each type per rule is provided in the profiler output. For each rule, the time, map name, and type name are reported.
Above	Use the Above option to obtain a smaller output file. When you specify a value in 1/10 milliseconds (10000 = 1 second), types times are only recorded when they exceed that value.
Summary	When you choose the Summary option, the report contains a summary of types per rule. A type name is displayed once with the total number of iterations and total elapsed time for those iterations. Sample report

File output

Option	Description
Fixed Width	This option produces a readable, flat file output.
Comma Delimited	This option produces a comma-delimited output which is useful for exporting to spreadsheet programs.

Where Used view

You can see all the locations in a map source (.mms) file where a particular map is called in rules, where a particular type is used in rules, or where the maps and types are used by a rule, in the Where Used view.

Open the Where Used view by right-clicking a map in the Outline view, or a type in the Map editor, and selecting Where Used.

From the Where Used view, you can go directly to the location where the map source file references the type or the map in a rule by double-clicking any of the locations returned in the view.

Search options

The Design Studio supports standard search options.

Search and replace

- Use the search operation to search through a single map source file or schema file, or multiple map source files or schema files, within search scope parameters that you specify, such as workspace, working set, and so forth.
- Use the replace operation to first do the search, and then replace text in the search results.

Find and replace

- Use the find operation to search for matches on text within a single map source file or schema file that you have open in the editor.
- Use the replace operation to replace text in the matched find results. This option is only available for searches within a schema.
- You can also do these operations using search and replace.

- [**Search and replace**](#)

This topic describes the features of search and replace.

- [**Find and replace**](#)

This topic describes the features of find and replace.

Search and replace

This topic describes the features of search and replace.

You have the option to do a search, which does a search operation only, or a replace, which does both a search operation and a replace operation. The search option searches the entire contents of a single map source file or multiple map source files, or a single schema file or multiple schema files based on the criteria you selected. The replace option does the search, and replaces the values in the search string with a replacement value for the search result components that you select to be included in the replace operation.

Features of the search and replace options:

- Run through the Transformation Extender tab in the Search window navigated from the Search menu on the menu bar.
- Operates on an entire map source or schema file or files. You can set the scope of the **Search** operation to do the search within the workspace, a specific working set, or a specific directory. For map source files, you can also set the scope to do the search in a current Map editor.
- Within a map source file or files, searches on:
 - All schema names
 - All card names
 - All output objects
 - All map names
 - All card type names
 - All map rules
 - All reference files

For example, you can search through map cards within a map source file, and replace file names ending with .txt with .dat. You can replace all card types named Record Set Data with Record Data. You can find card names that include the word Test.

- Within a schema file or files, searches on:
 - Type properties
 - Component rules
 - Restrictions
 - Type names

For example, you can search through all schemas in your workspace, and replace those type names named month with mm.

- [**Search For**](#)
Search For: is used to set the entity on which search and replace operates.
- [**Scope**](#)
Scope: is used to set the range within which search and replace operates.
- [**Using search and replace**](#)
There are different criteria that you can set when you use search and replace in the Design Studio.

Search For

Search For: is used to set the entity on which search and replace operates.

The following list describes the search entity options you can choose:

Type Tree Name

 Searches schema names

Card Name

 Searches card names

Output

 Searches output objects that have map rules

If you are doing a replace operation, it cannot replace text for this selection.

Map Name

Searches map names

Card Type Name

Searches card type names

Rule

Searches map rules

Reference Files

Searches reference files

Scope

Scope: is used to set the range within which search and replace operates.

The following list describes the search scope options you can choose:

Workspace

Searches for the search entity you selected, within the entire Eclipse workspace, which is where your projects are stored on your local drive

Working set:

Searches for the search entity you selected, within an entire Eclipse work set, which you create containing a subset of resources in the workspace

Directory:

Searches for the search entity you selected, within an entire folder

Current editor:

Option is only available when you select Map Source Files under Search In:

Searches for the search entity you selected, within the map source file that is open in the Map editor

Selection options:

- All Maps (default)
- Current Map
- Current and Reference Maps

Using search and replace

There are different criteria that you can set when you use search and replace in the Design Studio.

To use search and replace:

1. Click **Search > Search** on the menu bar, or press **Ctrl+H**.
The Search window opens.
2. Click the Transformation Extender Search tab.
This is the view through which you make your search and replace selections.
3. Enter text or wildcard characters in the Search string field.
Case sensitive and Whole word are optional fields that you can select.
4. Under **Search In:** click **Map Source Files**.
5. Under **Search For:**, click one of the search entities.
6. Under **Scope:**, click one of the scope entities.
7. Click **Search** to do a search operation only, or **Replace** to do both a search and a replace operation.
8. If you clicked **Replace**, the Map Designer window opens and displays the following message: Changes made using replace can not be undone. Do you want to continue with replace? Respond to the message by clicking **OK**.
9. While the operation is running, it displays the Search Query window containing the **Operation in progress** message. If it is searching through a large number of resources, the window remains open. Click **Run in Background** so that you can continue to access the Design Studio graphical user interface (GUI).
You can monitor the operation's progress by opening the Progress view, and viewing the progress indicator. You can also cancel the operation in this view.
The operation completes and displays any matches in the lower pane under the **Search** tab.
10. Click a map source file listed in the search results to see the details of the matches.

Double-click any map name in the list to locate it in the Map editor.

11. If you clicked **Replace** to do both a search and a replace operation, do the following additional steps:

- a. Select the check box next to each map name in the search results details to include in the replace operation.
There are two options in Search view that you can use if there are many search results to include (**Select All**), or exclude (**Deselect All**) from the replace operation.

- **Select All** - automatically selects all of the maps
- **Deselect All** - automatically deselects all of the maps, which were previously selected

- b. Enter the replacement text in the **With:** field.

The **Replace:** field contains the text you entered in the Search string field of the Search window.

- c. Click **Replace**.

The operation replaces the text in the **Replace:** field with the text in the **With:** field in every search result you selected to be included in the operation. It cannot replace text if you selected **Output** under the **Search For:** pane. When the operation completes, the Map Designer window displays a message about how many replacements were made.

- d. Click **OK**.

Find and replace

This topic describes the features of find and replace.

You have the option to do a find, which does a find or a find next operation only, a replace, which does a replace operation, a replace/find, which does both a replace operation and a find next operation, and a replace all, which does a replace operation for all found text. The available options are dependent on the objects you are searching.

Features of the find and replace options:

- Run through the Find/Replace window navigated from the Edit menu on the menu bar, or from the Find/Replace option on the context menu of an object.
- Operates on selected objects in a map source or schema file.
- Within a map source file, searches on:
 - Object names in a map card
 - Map rules in a card
 - Map elements in the Organizer folder, which is in the Outline view

Only the Find option is available for searches within a map source file.

- Within a schema file, searches on:
 - Text characters associated with types
 - Text characters associated with components
 - Text characters associated with restrictions

All options are available for searches on types within a schema file.

Only the Find and the ReplaceAll options are available for searches on components within a schema file.

- [Using find and replace](#)

There are different criteria that you can set when you use find and replace in the Design Studio.

Using find and replace

There are different criteria that you can set when you use find and replace in the Design Studio.

To use find and replace:

1. Select the area in which you want to search. For example, select the card to be searched.
2. Click **Edit > Find/Replace**.
3. In the Find field, enter the text you want to find.
4. To customize your search, select additional options in the window pertaining to direction, scope, case-sensitivity, whole words, and so forth.
5. Click Find to begin the search.
When search criteria is found, it is highlighted within the area that you selected in Step 1.
6. At this point, you can select from the following options:
 - To skip this instance and find the next, click Find.
 - To cancel the search, click Close.

Maps and map source files

The tasks involved in creating and managing map files include:

- Creating map source files
- Creating maps in the map source file
- Creating input and output cards for the map
- Building compiled maps
- Running maps
- Viewing run results

A map defines how to generate a data object of a certain type or multiple independent data objects, each of a certain type. A map contains input and output cards that transform data content from source formats to destination formats according to rules contained in the map.

A map source file stores related maps and functional maps in source format.

The Extender Navigator view lists the folder and file name of one or more map source files.

You must build (compile) a map source file before you can execute it.

- [Map name guidelines](#)
There are guidelines for you to follow when you are naming a map.
- [Map source file differences](#)
Map source files contain maps, and maps contain cards. Map source file differences are compared for each map source file at the map and card levels.
- [Troubleshooting tip](#)

Map name guidelines

There are guidelines for you to follow when you are naming a map.

Name your map according to the following guidelines:

- A map name can be up to 32 bytes in length in UTF-8 encoding.

- A map name cannot be a reserved word.
 - A map name cannot contain only digits or periods, or both.
 - A map name can contain combinations of the following types of characters:
 - Letters
 - Digits
 - ASCII characters 128–255
 - These special characters: ~ # % \ ? _ . ; `
 - Double-byte characters
- It is helpful to use an action-oriented map name that describes the purpose of the map. For example: **CreatePO**.
-

Map source file differences

Map source files contain maps, and maps contain cards. Map source file differences are compared for each map source file at the map and card levels.

Map source files are considered "different" when:

- A map exists in one map source file and does not exist in the other map source file.
- Any of the maps are different.

Maps are considered different if any of the following differences exist:

- Any card is different
- Any card exists in the first map and does not exist in the second map
- Any audit setting is different
- Any remark is different
- Any unresolved rule is different
- The order of the cards in the map is different

Cards are considered different if:

- The card settings are different
- The map rules on output cards are different

In addition to the map source file differences, the source file differences feature identifies differences for schemas, system definition files, and database/query files. You can view the results of the map source file differences.

Compare map source files in the Map Designer.

Troubleshooting tip

If map differences are indicated but not viewable, there is a possibility that if the map was exported, the adapter specified in the map source file was no longer supported or not installed at the time of export. When the map source is imported and then compared, a difference is detected but is not viewable.

Input and output cards

A map defines how to generate data objects of a certain type. A map contains input and output cards that transform data content from source formats to target formats. This data transformation occurs according to mapping rules on the data objects in the output card in the map.

Input and output cards represent data objects. Each card represents one data object, which is defined as a particular type. There are two types of map cards: input cards and output cards.

- An input card contains the complete definition of an input for the map including information such as source identification, retrieval specifics, and the behavior that should occur during processing.
- An output card contains the complete definition of an output for the map including information such as target identification, destination specifics, and the behavior that should occur during processing.

Each input of data and each output of data requires content definition settings. The content of each card is specified as a type in a schema.

- [Card overview](#)

Each map can contain none or more input cards and none or more output cards.

- [Native XML schema support](#)

- [Native JSON schema support](#)

Native JSON schema type tree is created automatically when a valid JSON schema is entered in the type tree field of a card.

- [Static input cards](#)

You can reduce the processing overhead of a map that runs repeatedly by including static input files in the map. A *static* input file doesn't change during map processing, such as a configuration file or a lookup table. A static file in a compiled map is parsed and validated only once, when the map loads. Other inputs are validated each time the map runs.

- [Automatically mapping source to target](#)

The Auto Map is available for you to use from the Design Studio.

Card overview

Each map can contain none or more input cards and none or more output cards.

The maximum number of input cards in a map is 100. The maximum number of output cards in a map is 100.

Each card represents one data object, which is identified by selecting a type from a schema. A card can represent an object in a row, a row in a record, the entire record, or the entire file.

Each card specifies a schema and a type as part of the card definition. This type, and the schema that contains it, is the link between the Type Designer and the Map Designer that defines the structure and properties of the data in the card.

- The input card represents input data. Input data validation occurs in the Type Designer.
- The output card represents output data and contains the map rules that transform that data. Therefore, only the output card has a rule column.
- [Card specifics](#)
- [Compositional hierarchy](#)

Card specifics

- More than one card in a map can reference the same type but the cards must have different card names.
- Each card is assigned a number when it is created. Cards are numbered sequentially, beginning at 1. The card numbers determine the order in which the cards and the data in those cards is evaluated. The card number is important when a map is used as a functional map. To reorder the cards, use the **Reorder** command on the Card menu.
- The structure of the card object appears in the card. An icon represents each component that the entire card data object is composed of. Each icon in the input and output card represents a type that has been defined in the Type Designer.
- The output column in the output card shows the structure and the names of the components of the output data object.
- In the case of partitioned groups, the structure shown is the partitioned group and its subtypes. An icon is shown for each component and each partition that compose the entire card object.
- Each output name appears in a separate cell in the output card.
- The rule column contains a rule cell for each output object. Enter map rules to define how to generate the data object of this card.

Compositional hierarchy

The Composition view of the cards shows the contents of the type definitions. Types are arranged in a compositional hierarchy, which shows the structure, layout or composition of the data. This hierarchy is different from the hierarchy in a schema, which is a classification hierarchy.

The data is defined in the Type Designer according to specific properties. The types are arranged on the input and output cards in the order they appear in the data. This compositional hierarchy shows the structure of the data.

Native XML schema support

IBM Transformation Extender provides native XML schema support. It supports generating specified global elements, specifying data types to be validated by using xsi:type, and processing data that contains complex types with mixed content. The associated NativeXMLSchema properties are added when you specify a native XML Schema (.xsd) file in the TypeTree property on an input or output card. The added properties are Identifier, GlobalElements, XSITypes, and MixedContent.

If you want to change the default values for the NativeXMLSchema properties, use the Native XML Schema Customization wizard.

An output card includes the Type>Metadata>xsi:schemaLocation property and the Type>Metadata>xsi:noNamespaceSchemaLocation property when you specify an XML schema in the card's TypeTree and Type properties.

- [Input-card type requirements for XML documents](#)
- [Parsing native XML schemas](#)
The initial map open process fully parses and validates the schema and creates a read-only, intermediary native schema schema (.mtx) file. Subsequent open processes retrieve the Transformation Extender-specific information that is stored in the cached .mtx file, similar to retrieving the information from a schema (.mtt) file. Retrieving the information from the cached .mtx file is designed to improve the response time for opening maps that reference an XML Schema.
- [Running the Native XML Schema Customization wizard task flow](#)
Follow this task flow to run the Native XML Schema Customization wizard from the input or output card of your map.
- [Support for XSDL hints in output cards](#)
When the TypeTree and Type properties of an output card specify an XML schema, Map Designer includes the xsi:schemaLocation and xsi:noNamespaceSchemaLocation fields in the card properties. The XSDL hint that you specify in the xsi:schemaLocation or the xsi:noNamespaceSchemaLocation field is included in the output XML schema when the map runs. If you specify XSDL hints in both the xsi:schemaLocation and the xsi:noNamespaceSchemaLocation fields, only the xsi:schemaLocation hint is included in the output XML schema.

Input-card type requirements for XML documents

When a native XML schema has a prolog, the Type setting of the main input card must be XSD (the root of the XML document). A parser error results when the XML schema has a prolog and the Type setting specifies a sub-element of the root.

As a general practice, use XSD as the Type setting in the main input card of all native XML schemas. The XSD setting parses correctly regardless of whether the schema includes a prolog.

For all Xerces schemas, use Doc XSD as the Type setting of the main input card, regardless of whether the XML document has a prolog. Parsing errors can result if the XML document includes a prolog and you specify a sub-element of Doc XSD as the Type setting of the main input card.

Parsing native XML schemas

The initial map open process fully parses and validates the schema and creates a read-only, intermediary native schema schema (.mtx) file. Subsequent open processes retrieve the Transformation Extender-specific information that is stored in the cached .mtx file, similar to retrieving the information from a schema (.mtt) file. Retrieving the information from the cached .mtx file is designed to improve the response time for opening maps that reference an XML Schema.

Running the Native XML Schema Customization wizard task flow

Follow this task flow to run the Native XML Schema Customization wizard from the input or output card of your map.

Before you can run the wizard, make sure that a native XML schema (.xsd) file is specified for the TypeTree property and that the NativeXMLSchemaCustomization properties were added.

Use the Native XML Schema Customization wizard to change the default values for the NativeXMLSchema based on your mapping requirements. When you start the Native XML Schema Customization wizard, it takes you step-by-step through the process that updates the NativeXMLSchemaCustomization properties in the card.

To run the Native XML Schema Customization wizard:

1. Click the button next to NativeXMLSchemaCustomization under the Value column.
The Native XML Schema Customization wizard opens. Follow the wizard prompts that guide you through the different windows.
2. Select the global elements that you want to be generated in the card. The card is generated with these specified global elements. Click Next to continue. You can click Finish to save the settings and return to the card that displays in the Map editor.
You must select at least one global element to continue to the next window in the wizard.
3. Select the global types for which you want to enable the use of derived types on base types by using the xsi:type attribute. Click Next to continue. You can click Finish to save the settings and return to the card that displays in the Map editor.
You do not have to make a selection in this window to continue to the next window in the wizard.
4. Select Support complex types with mixed content if the map must process data that contains complex types with mixed content.
You do not have to make a selection in this window to continue.
5. Click Finish to save the settings.
The Edit Card window redisplays in the Map editor.

The Identifier, GlobalElements, XSITypes, MixedContent properties that are listed in the card under NativeXMLSchemaCustomization contain updated values, depending on your selections.

If you updated all of the NativeXMLSchemaCustomization properties, you would see the following settings:

- Identifier: a unique numeric identifier for the native XML schema
- GlobalElements: a numeric designation and *global_element_name {namespace}* value for each of the global elements you selected that you want to be generated in the card
- XSITypes: a numeric designation and *global_type_name {namespace}* value for each of the global types for which you want to enable the use of derived types on base types by using the xsi:type attribute
- MixedContent: value that is set to True

Support for XSDL hints in output cards

When the TypeTree and Type properties of an output card specify an XML schema, Map Designer includes the xsi:schemaLocation and xsi:noNamespaceSchemaLocation fields in the card properties. The XSDL hint that you specify in the xsi:schemaLocation or the xsi:noNamespaceSchemaLocation field is included in the output XML schema when the map runs. If you specify XSDL hints in both the xsi:schemaLocation and the xsi:noNamespaceSchemaLocation fields, only the xsi:schemaLocation hint is included in the output XML schema.

The xsi:schemaLocation and xsi:noNamespaceSchemaLocation properties are not available in input cards.

Native JSON schema support

Native JSON schema type tree is created automatically when a valid JSON schema is entered in the type tree field of a card.

The JSON document is determined to be a JSON schema when a field called \$schema is found in outermost group. If this element is not found in the JSON document will be treated as a template.

When you use native JSON to define the main input or output card, always select the main Type as the first JSON type.

Note: The main input or output card is the card that has the file or other source that defines how to parse or create the document.
This restriction does not apply to functional maps.

A card property called Data Language is present for output cards which use native JSON. This card property allows the text encoding of the JSON output to be selected. The following encodings are supported: UTF-8, UTF-16BE, UTF-16LE, UTF-32BE, UTF-32LE. Previously, an item type in a native JSON type tree named encoding was used for this purpose. The encoding item remains valid for type trees and maps which use it. Conversion of a rule on an encoding item is necessary if the rule becomes invalid due to the absence of an encoding item in a native JSON type tree.

The encoding field on input always returns the value NONE. On output, this field can be set to any valid JSON encoding and the JSON document is produced that uses that encoding. If you do not configure the encoding, the default encoding used is UTF-8.

Restrictions with JSON Schema:

- Tuple validation is not supported.

- Elements will be loaded into a string array, but not validated.

The target of "if" statement must appear in the data stream before fields in the Then or the Else. Out of order data will cause the "if" to fail.

Dependencies that alter the initial type is not be supported. Data will be loaded into fields but not validated.

For example: Field B is defined as string.

A dependency where if field A is present then field B now must be a date.

ITX will handle this, but no validation will be done.

Static input cards

You can reduce the processing overhead of a map that runs repeatedly by including static input files in the map. A *static* input file doesn't change during map processing, such as a configuration file or a lookup table. A static file in a compiled map is parsed and validated only once, when the map loads. Other inputs are validated each time the map runs.

No new-line conversion or character-set conversion takes place in the file when the map runs on any platform.

To include a file in a map, specify the static file adapter and the file name on the input card before you compile the map. The file must exist at the time when the map compiles, and must be less than 10 MB in size. For static files, the OnSuccess property is always Keep and the OnFailure property is always Rollback.

You can override a static file input in a compiled map with another adapter. The static file input must pass validation in order for the map to load, but the map uses the adapter override when it runs.

You cannot use the static file adapter to override an input card during map execution. Static file input must be compiled into the map.

To change the contents of a static file, you must recompile and redeploy the map. When you enable map caching, you must restart the process before the map can use the updated static file.

Note: The `tx_install_dir/data_dir/config/config.yaml` configuration file controls map caching for Launcher and **RUN** maps. See the related information links for details about map caching in other environments.

- [Static file validation and tracing](#)

Transformation Extender (TX) validates a static input file in a compiled map only once, when the map loads, and produces a trace file only if the static file fails validation. TX validates other map inputs each time the map runs, and logs them in the map trace file as specified by the map settings.

Related information

- [TX map caching with IBM Integration Bus](#)
- [TX map caching with IBM Sterling B2B Integrator](#)
- [TX map caching with IBM Business Process Manager Advanced](#)
- [Map caching with TX programming APIs](#)

Static file validation and tracing

Transformation Extender (TX) validates a static input file in a compiled map only once, when the map loads, and produces a trace file only if the static file fails validation. TX validates other map inputs each time the map runs, and logs them in the map trace file as specified by the map settings.

When a static input file fails validation, the map does not load. TX logs the validation error in the appropriate log for the runtime environment for example, the Launcher's CompoundSystem log or the IBM Integration Bus log. The error includes the location of the static input trace file. TX creates the static input trace file as `map_name.mtr` in the map directory. If the map loads from memory, TX creates the `map_name.mtr` file in the map working directory.

When a static input file fails validation on a z/OS® system running in batch, CICS® or IMS, the trace is written to the DD name DTX#TRCE. Because the trace is written as a byte stream, it lacks formatting if the DTX#TRCE is directed to SYSOUT. If formatting is required, direct the DD card to a zFS file. For example:

```
//DTX#TRCE DD PATH='/tmp/dtx#trce.mtr',
//           PATHDISP=(KEEP),
//           PATHOPTS=(ORDWR,OCREAT),
//           PATHMODE=(SIRWXU,SIRWXG,SIROTH)
```

When you trace a static input file and the static file passes validation, the map trace file describes the static input as an existing work file. For example, you might trace two map inputs. Input 1 is not a static file and the input is invalid. Input 2 is a static file that passed validation during load. The trace file includes messages similar to the following:

```
.
.
(Level 2: Offset 389, len 6, comp 3 of 3, #1, DI 000000000053:)
Data at offset 389 ('00000<CR>') is INVALID data of TYPE
  X'0012' (total_hours element example).

(Level 1: Offset 297, len 98, comp 1 of 1, #4, DI 000000000054:)
Data at offset 297 ('Work A.      ...') is INVALID data of TYPE
  X'0003' (timesheet example).

(Level 0: Offset 0, len 396, comp 1 of 0, #1, DI 000000000055:)
Data at offset 0 ('FirstName      ...') is INVALID data of TYPE
```

```
X'0002' (all_employees example).  
INPUT 1 exists, but its type is in error.  
End of Validation messages for INPUT CARD 1.  
Reusing an earlier work file for INPUT CARD 2.  
End of Execution messages.
```

Automatically mapping source to target

The Auto Map is available for you to use from the Design Studio.

- [Auto Map overview](#)

The Auto Map automatically maps selected source and target groups.

- [Running the Auto Map task flow](#)

Follow this task flow to run the Auto Map in the Design Studio.

- [Setting the Auto Map options](#)

You can optionally change the default settings that originate from IBM Transformation Extender, or that you set up in the Preferences window.

- [Selecting the rules to create](#)

You can optionally select the rules that you want the Auto Map operation to create.

- [Editing rule proposals](#)

You can optionally change the rules that IBM Transformation Extender proposed for the target objects.

- [Renaming the map](#)

You can optionally change the functional map name.

Auto Map overview

The Auto Map automatically maps selected source and target groups.

Your source groups and item names must match your target group and item names according to the match properties set under Auto Map preferences.

Use the Auto Map wizard when you want to automatically map selected groups of your map. You can also run the Auto Map without using the wizard, by pressing Ctrl+A. The Auto Map operation maps the objects that you select in the following ways:

- Maps the objects that you selected from an input card in your map to the objects that you selected from an output card in your map.
 - Maps the objects that you selected from an output card in your map to the objects that you selected from another output card in your map.
-

Running the Auto Map task flow

Follow this task flow to run the Auto Map in the Design Studio.

When you start the Auto Map wizard, it takes you step-by-step through the process that automatically maps selected source and target objects.

Before you start the Auto Map wizard, you can select options in the Auto Map Preferences. These options are the default selections that the Auto Map uses. As the Auto Map wizard takes you through the process step-by-step, you can change the default selections. However, any changes to the selections that you make in the Auto Map wizard, are not saved. Subsequent runs of the Auto Map use the original default selections that you made in the Auto Map Preferences window. You can, again, change the default selections dynamically in the wizard. But, you can always update the settings in the Auto Map Preferences so that they are saved and used by the Auto Map in subsequent runs. Then, you do not have to reselect the options when you rerun the wizard. You are not required to select the Auto Map Preferences before you run the Auto Map. But, you can avoid continually reselecting the options every time you run the wizard. At any point, you can set the options in the Auto Map Preferences.

If required, you might also want to back up your map before you run the Auto Map because the Auto Map operation does not support an **Undo** function.

To run the Auto Map:

1. Open the map in the Map editor.
2. Select two groups that you want to map automatically.
One of the groups that you select must be a group component in an output card. You can select choice groups. To toggle your group selections on and off, press Ctrl, and click the selection.
Also, select source and target objects at the levels that you want to map. If you select the top level, the Auto Map operation maps all the groups and nested groups. The Auto Map icon in the Map editor toolbar is enabled.
3. Click the Auto Map icon.
The Map Designer window opens and displays an informational message. It warns you that the Auto Map operation does not support Undo so that you can back up your map before you complete running the Auto Map.
4. Click OK in the Map Designer window.
The Auto Map wizard starts. The Auto Map window opens and displays objects under the source and target groups that you selected.
5. Follow the wizard prompts. You can set the various options that the Auto Map operation uses to automatically map the source and target objects.
6. Click Finish.

The Auto Map operation automatically maps the selected source and target objects that are based on the matching properties and the proposed rules that you selected. The operation does not map the rule from the source to the target for the following objects:

- Objects that you did not select to create a rule.
- Objects for which the mapping process did not find a match.

Tip: You can run the Auto Map without using the wizard, by pressing Ctrl+A after you select the two groups that you want to map automatically. The Auto Map operation uses the settings that originate from IBM Transformation Extender, or that you set up in the Preferences window.

Setting the Auto Map options

You can optionally change the default settings that originate from IBM Transformation Extender, or that you set up in the Preferences window.

After you click Next in the Automatically map source to target page of the Auto Map wizard, the Select the options to map source to target page opens. You have the option in that page to configure settings that the Auto Map operation uses for the mapping process. The updated settings however, are not saved; they are not available to subsequent runs of the Auto Map. If you want the settings to be saved, configure them in the Preferences window.

To set the Auto Map options:

1. In the Select the options to map source to target page of the Auto Map wizard, you can set any of the mapping process options.
2. Click Next to open the next page in which you can select more options, or click Finish to run the Auto Map.

If you returned to that page from a subsequent page and changed settings, and then clicked Next, the Map Designer window opens. You are prompted to answer Yes or No to the **Do you want to run the match operation again?** message.

After you click Next, the Select rules to create page opens. It lists the source and target objects that meet the matching criteria that are specified on that page.

Selecting the rules to create

You can optionally select the rules that you want the Auto Map operation to create.

After you click Next in the Select the options to map source to target page of the Auto Map wizard, the Select rules to create page opens. It lists the source and target objects that meet the matching criteria that are specified on the Select the options to map source to target page.

You can run the Auto Map operation with the default settings or make the following optional changes from that page:

- Change the selection of the target objects for which you want the Auto Map operation to create rules.
- Edit the proposed rules.
- Rename the functional maps.

You click Finish on that page to run the Auto Map operation and exit the wizard.

To change the target object selections and the rules that you want the Auto Map operation to create for those objects, do the following optional steps:

1. Change the selection of the target objects for which you want the Auto Map operation to create rules.
 - a. Select the check box of the target objects for which you want the Auto Map operation to create rules.
The default is that the target objects listed in that page are all selected. Therefore, you can clear the check box of those target objects for which you do not want the Auto Map operation to create rules. Another option is to clear the check box of a target object that is a group and then make your selections.
The result is that those target objects that you did not select, are listed in the output with no rule mapped to it.
2. Edit the proposed rules.
 - a. Select one of these target objects for which there is an i indicator that is displayed between the target object and its rule.
The information (i) indicator signals to you that there are more rule proposals that you can optionally select for the object.
 - b. Click Edit rule proposals to open the Rule Proposals window in which you can change the rule selections that IBM Transformation Extender proposed as a result of the matching criteria.
3. Rename the functional maps.
 - a. Select one of the target objects for which there is a rule that contains a functional map.
 - b. Click Rename map to open the Rename Map window in which you can change the functional map name.
4. Click Finish to run the Auto Map operation and exit the Auto Map wizard.

The Auto Map operation automatically maps the selected source and target objects that are based on the matching properties and the proposed rules that you selected. The rule from the source is not mapped to the target for the following objects:

- Objects that you did not select to create a rule.
- Objects for which the mapping process did not find a match.

Editing rule proposals

You can optionally change the rules that IBM Transformation Extender proposed for the target objects.

The information (i) indicator that is displayed between the target object and its rule, signals to you that there are more rule proposals that you can optionally select for the object. After you select one of these target objects for which there is an i indicator that is displayed, and click Edit rule proposals in the Select rules to create page of the Auto Map wizard, the Rule Proposals window opens. The window lists all of the source objects that are possible matches. When you select one of the source objects, the window lists all of the various proposed rules for that selected source object. You can select a source object, and one of the rules to change the rule that IBM Transformation Extender proposed.

To edit the rule proposals:

1. In the Rule Proposals window of the Auto Map wizard, select a source object under Possible Matches.
When you select the source object, the proposed rules for that object are listed under Proposed Rules.
2. Select one of the rules.
3. Click Finish.

The Rule Proposals window closes. In the Select rules to create page, the proposed rule on the target object you selected is replaced by the rule that you selected.

The Auto Map operation automatically maps the selected source and target objects that are based on the matching properties and the proposed rules that you selected. The rule from the source is not mapped to the target for the following objects:

- Objects that you did not select to create a rule.
- Objects for which the mapping process did not find a match.

Renaming the map

You can optionally change the functional map name.

After you select one of the target objects for which there is a rule that contains a functional map, and click Rename map in the Select rules to create page of the Auto Map wizard, the Rename Map window opens. The window displays the default functional map name in the Map Name field.

To rename the functional map:

1. In the Map Name field that is displayed in the Rename Map window of the Auto Map wizard, rename the functional map.
The default is the selected functional map.
2. Click OK.
The Rename Map window closes. In the Select rules to create page, the functional map name in the rule for the targeted object you selected is replaced by the changed map name.

The Auto Map operation automatically maps the selected source and target objects that are based on the matching properties and the proposed rules that you selected. The rule from the source is not mapped to the target for the following objects:

- Objects that you did not select to create a rule.
- Objects for which the mapping process did not find a match.

Card settings

You configure how to run a map by selecting the Map menu, choosing Settings, and specifying the settings in the Properties view. Define **SourceRule** card settings on the input card, and define **TargetRule** card settings on the output card. All of the card settings are in the Properties view, the Data Audit Settings view, and the Remarks view.

Each input of data and each output of data require content definition settings. Card settings define the data object the card represents, and how the data is retrieved or routed.

Card settings can be exported. See "[Exporting a Map Source File](#)" for details about exporting map card settings.

- [Schema](#)
- [SourceRule input card settings](#)
- [TargetRule output card settings](#)
- [Input and output card settings](#)
- [Effects of modifying a schema](#)
- [Changing a type that is used for a card](#)
- [SyntaxCard](#)
The SyntaxCard setting defines the output card as a syntax card.
- [Editing a card](#)

Schema

The Schema setting for input and output cards defines the card name, schema, type, and file.

- [CardName](#)
- [Schema](#)
- [Type](#)

CardName

Each card in a map has a unique name to distinguish cards in the map. Precise naming of the card is practical and useful. A good practice is to name the card the same name as the data object it represents. Follow the card name guidelines.

Although cards can be named most anything you want, typically you would want to use a name that describes the data, such as **InputFile**, **ClaimData**, **ItemMasterRow**, and **CreditApprovalMsg**.

- [CardName guidelines](#)

CardName guidelines

Card names follow these guidelines:

- A card name can be up to 32 bytes in length in UTF-8 encoding.
- A card name cannot be a reserved word.
- A card name cannot contain only digits and or periods.
- A card name can contain *only* the following:
 - Letters
 - Digits
 - ASCII characters 128-255
 - These special characters: ~ # % \ ? _ ' . ; `

Schema

Cards represent a particular data object. Therefore, each input card and each output card is associated with a type in a schema that describes the content of that data. Therefore, each card includes a schema and type name as part of its definition. This schema is the link between the Type Designer and the Map Designer that communicates the structure and properties of the data to the Map Designer.

If you select a .xsd file, the NativeXMLSchema properties are added to the card. In the Edit Card window, expand NativeXMLSchema to see the added Identifier, GlobalElements, XSITypes, and MixedContent properties, along with their default values, as shown in the following list.

- Identifier: <Default>
- GlobalElements: All
- XSITypes: None
- MixedContent: False

Change the default values for the properties by running the Native XML Schema Customization wizard.

- [NativeXMLSchemaCustomization](#)

NativeXMLSchemaCustomization

NativeXMLSchemaCustomization includes several properties that support the Native XML schema.

- [NativeXMLSchemaCustomization > Identifier](#)
Native XML schema support includes the ability to generate the identifier of the file that the schema generates.
- [NativeXMLSchemaCustomization > GlobalElements](#)
Native XML schema support includes the ability to select global elements to be generated.
- [NativeXMLSchemaCustomization > XSITypes](#)
Native XML schema support includes the ability to specify the global types for which you want to enable the use of derived types on base types by using the xsi:type attribute.
- [NativeXMLSchemaCustomization > MixedContent](#)
Native XML schema support includes the ability to use complex types with mixed content in the card.

NativeXMLSchemaCustomization > Identifier

Native XML schema support includes the ability to generate the identifier of the file that the schema generates.

When you specify a .xsd file for TypeTree, the generated identifier displays under the Identifier property under NativeXMLSchemaCustomization in input and output cards. The default value is Default. When you run the Native XML Schema Customization wizard to update the other NativeXMLSchemaCustomization properties, the wizard generates a numeric value for the identifier.

NativeXMLSchemaCustomization > GlobalElements

Native XML schema support includes the ability to select global elements to be generated.

When you specify a .xsd file for TypeTree, the generated GlobalElements property displays under NativeXMLSchemaCustomization in the input or output cards. The default value is All. You can run the Native XML Schema Customization wizard to select the specific global elements that you want to be generated in the card. When the global elements are specified, the NativeXMLSchemaCustomization.>.GlobalElements properties contain the numeric designation and *global_element_name {namespace}* value for each of the global elements you selected. As a result, instead of generating all of the global elements, IBM Transformation Extender generates only the global elements that you selected.

NativeXMLSchemaCustomization > XSITypes

Native XML schema support includes the ability to specify the global types for which you want to enable the use of derived types on base types by using the xsi:type attribute.

When you specify a .xsd file for TypeTree, the generated XSITypes property displays under NativeXMLSchemaCustomization in the input or output cards. The default value is None. You can run the Native XML Schema Customization wizard to enable the use of derived types on base types by using the xsi:type attribute. When the global types are specified, the NativeXMLSchemaCustomization.>.XSITypes properties contain the numeric designation and *global_type_name {namespace}* value for each of the global types you selected.

NativeXMLSchemaCustomization > MixedContent

Native XML schema support includes the ability to use complex types with mixed content in the card.

When you specify a .xsd file for TypeTree, the generated MixedContent property displays under NativeXMLSchemaCustomization in input or output cards. The default value is False. You can run the Native XML Schema Customization wizard to set support of mixed content for the card by selecting Support mixed content types. When the support is set, the Native XML Schema Customization > Mixed Content property in the card is set to True. As a result, the data that the card represents can contain complex types with mixed content.

Type

The Type Name setting specifies the name of the type in the schema that describes the data represented by the card. This type must exist in the schema specified with the Schema setting.

- [Metadata](#)

The Schema > Type > Metadata card setting is applicable to Xerces XML schemas.

Metadata

The Schema > Type > Metadata card setting is applicable to Xerces XML schemas.

The default location for metadata, XML Schema, or DTD, is the location that is specified in the Doc group of the XML schema (type property: Intent > Validate as > Location).

Use the Schema > Type > Metadata card setting to specify the XML Schema or DTD location in the map at the card level.

The Metadata card setting takes precedence over the location that is specified in the schema.

You can override the Metadata setting by using the command line (-M). However, it cannot be overridden at the system level by using the Integration Flow Designer.

Change the metadata location by using this card setting in the map instead of changing it in the schema. When the Schema > Type > Metadata card setting is not set, the XML Schema or DTD location is derived from the schema by default.

- [Metadata > NameSpaces](#)

The NameSpaces property lists all of the namespaces that are referenced in the schema that is used in the Metadata location in an input or output card.

Metadata > NameSpaces

The NameSpaces property lists all of the namespaces that are referenced in the schema that is used in the Metadata location in an input or output card.

On an input card, you can do the following processes:

- Delete
 - You can delete a namespace by right-clicking the specific namespace and clicking Delete.
- Reload
 - You can reload all of the namespaces that you deleted by right-clicking the NameSpaces property and clicking Reload.

On an output card, you can do the following processes:

- Add
 - You can add a namespace by right-clicking the NameSpaces property and clicking Add.
 - The Namespace Dialog window opens.
 - You define custom namespaces by specifying the URI in this window.
- Delete
 - You can delete a namespace by right-clicking the specific namespace that you want to delete, and clicking Delete.
- Reload
 - You can reload the namespaces that you deleted by right-clicking the NameSpaces property and clicking Reload.

If there are namespaces that are defined on the output cards, when the map runs, those namespaces are written to the beginning of the XML file.

For more information about namespaces, see the topics about XML schema datatypes in the Type Designer documentation.

SourceRule input card settings

SourceRule settings specify what data to retrieve, where to get it, how much to get, and what to do when an error occurs.

The **SourceRule** settings define how the input is retrieved. The data object specified in the input card defines what data is retrieved. The **SourceRule** settings define how the data is retrieved.

To define card-specific data retrieval behavior, specify the **SourceRule** settings for each input card. An input card represents an input source. The content of an input is a *data object* such as a table of patient records in a relational database, a message that contains a mortgage application, or a file of item master update records that is

transferred from another location using the FTP adapter.

The content of an input card is defined with the card **Name**, **Schema**, **Type Name** and Filesettings. These settings define what the data is, but do not specify how to retrieve the data. The **Source**, **FetchAs**, **WorkArea** and **FetchUnit** card settings are some of the **SourceRule** settings that define how the data is retrieved for each input card.

- [SourceRule Setting overrides](#)
- [Input data retrieval](#)
- [FetchAs](#)
- [FetchAs > WorkArea](#)
- [FetchAs > FetchUnit](#)
- [GET](#)
- [Source](#)
- [Source > FilePath](#)
- [Source > Command](#)
- [DatabaseQueryFile](#)
- [DatabaseQueryFile > Database](#)
- [DatabaseQueryFile > Query](#)
- [DatabaseQueryFile > File](#)
- [Transaction](#)
- [OnSuccess](#)
- [OnFailure](#)
- [Scope](#)

SourceRule Setting overrides

Input card settings are stored in the compiled map. You can override the setting for each option from other applications, including the Command Server, Launcher, and Integration Flow Designer.

Execution commands can be used to override the map settings or card settings compiled into the compiled map when a map is run. See the Execution Commands documentation for a complete description of the override execution commands.

The **SourceRule** card settings and their override execution commands are listed in the table below:

SourceRule Card Setting	Override Execution Command
FetchAs	Not available
Backup	Not available
WorkArea	Available (-W)
Source	Available (-IA), (-ID), (-IE), (-IF), (-IM)
FilePath	Available (-IA), (-ID), (-IE), (-IF), (-IM)
OnSuccess	Available (-I...X)
OnFailure	Available (-I...B)
Retry	Available (-I...XRcount:interval)
Warnings	Not available
Scope	Not available
FetchUnit	Not available

Input data retrieval

The **FetchAs** setting applies only to input cards. It indicates how the adapter retrieves the input data. By default, this setting is **Integral**, meaning that all of the input data is retrieved once from the source.

Setting this parameter to **Burst** means that the adapter will retrieve the input data in increments specified in the **FetchUnit** parameter. The units of data correspond to the highest-level repeating object in the schema. For example, a payroll file contains 105 records. Using **Burst** mode with the **FetchUnit** parameter set to 10, data will be sent in 11 "chunks". The first ten chunks will contain ten records each and the eleventh chunk will contain the remaining five records.

Despite the fact that bursts process some of the map data and that multiple sets of outputs are built, transactional integrity is not sacrificed. If desired, the **Scope** card setting can be set to **Map** to prevent any changes from being committed until *all* bursts have successfully completed.

There are several potential benefits to using **Burst** mode:

- A single map consumes a large quantity of input data.
However, only a small portion of that data is required to build an individual output. **Burst** mode can be used to minimize the size of the workspace (an internal index of types present in the data stream). In some cases, a smaller workspace can increase map performance.

- Coordinate multiple inputs together.
For example, the first unit of data for Input Card 1 must be coordinated with the first unit of data for Input Card 2, and so on. If **Burst** mode is not enabled, one input must then act as a "driver" that must search for matching outputs.
- Process a single logical unit of work at one time.
For example, you may want to process one row from a database table at one time or one message from a message queue at one time. This would equate to the **Burst** mode being enabled with the **FetchUnit** parameter set to 1.

FetchAs

The **FetchAs** setting on an input card specifies how the data will be retrieved when the map is executed.

Value

Description

Integral

Data is retrieved once (in its totality) from the data source. If another input is set to **Burst** mode, the same integral data for this card is used for each burst. When an input **FetchAs** is **Integral**, the **Source** is requested to retrieve data one time for the entire mapping.

Burst

Data is retrieved in incremental units from the data source. The unit of data retrieved for each burst is adapter-specific, and is specified as the **Source FetchUnit**. When an input **FetchAs** is **Burst**, the data is retrieved once per burst.

When an input **FetchAs** is **Integral**, any failure action causes the entire map to fail.

- When **FetchAs** is **Integral**, the maximum size of a file that a map can process is 2 GB.
- When **FetchAs** is **Burst**, the maximum size of a file that a map can process is much larger, but a maximum of 2 GB per burst.

FetchAs > WorkArea

Each time a map is executed, information about the data and map is kept in a workspace and is used as the input data is validated and the output data is built. This workspace information can either be stored in files or written to memory.

A card's **WorkArea** setting specifies whether to reuse that work area on subsequent executions of the same map. If you are repeatedly going to run the *same* map using the *same* input data more than once, setting the **WorkArea** card setting value to **Reuse** increases the speed of map execution because data validation for this input is not repeated. There are guidelines for reusing work areas.

Value

Description

Reuse

Reuse the work area. After a map runs for the first time, the work area that is created for the input card is not deleted. Then, on subsequent executions of the same map, the data for this input card is *not* validated, and the work area information for the card is retrieved from the existing work area.

!Reuse

Do not reuse the work area. The input card's **WorkArea** is created for each map execution.

- [Guidelines for reusing work areas](#)

Guidelines for reusing work areas

Use the following guidelines when defining whether to reuse workareas. The **WorkArea** card setting depends upon the map setting for **WorkSpace Location** and **PageSize** and **PageCount** settings.

- WorkSpace Location setting. If the map setting for **WorkSpace** is **File**, the input card's work area can be reused. If the map setting for **WorkSpace** is **Memory**, the input card's work area can be reused only when the map is initiated using the **RUN** function. It cannot be reused from an API.
- WorkSpace **PageSize** and **PageCount** settings. When reusing work files, the paging configuration (page size and number of pages) for the first execution and subsequent executions of the map *must* be the same.
For example, if the first execution (when the work files are created) uses a paging configuration of eight pages of 64K each, the subsequent executions (where the work files are reused) must use the same paging settings, in this case eight pages of 64K each.
- For One Map Only.** The **WorkArea** card setting applies to a given map only. For example, it does *not* apply to two different maps that use the same input.
- FetchAs = Integral.** When a map is executed and the input card **FetchAs = Integral**, the work area is automatically reused for each burst. If **WorkArea = Reuse**, that work area is saved on completion of the map execution.
- FetchAs = Burst.** When the input card has a **FetchAs of Burst**, a new work area is created for each burst. If **WorkArea = Reuse**, the last work area is saved on completion of the map execution.

FetchAs > FetchUnit

The **FetchUnit** setting only applies to input cards. The **FetchUnit** setting defines the number of units of data to retrieve each time a request for data is made to the adapter. **FetchUnit** is primarily used when the **FetchAs** setting is set to **Burst** although it may also be used in other situations. The default value for **FetchUnit** is **S** (unspecified all).

A specific **FetchUnit** value allows you to aggregate, or "batch", a set of physical units (such as messages on a queue) into a logical unit of work. The unit of data is adapter-specific.

Value

Description	
S	Get all objects, database rows, messages on a messaging queue, or files.
1 - nnnn	<p>Positive integer representing the logical or physical unit of data.</p> <p>For messaging adapters, FetchUnit defines the number of messages to retrieve.</p>
	<p>For File or Echo, FetchUnit defines the logical unit of data.</p> <p>For database adapters, FetchUnit defines the number of rows.</p> <ul style="list-style-type: none"> • FetchUnit details • FetchUnit with multiple input cards

FetchUnit details

The unit of data is adapter-specific.

- If the adapter is **File**, the **FetchUnit** refers to a logical unit of data and divides map execution into valid units of work based on a logical unit of data.

For other adapters, the **FetchUnit** refers to a physical unit of data.

- For messaging adapters, the **FetchUnit** setting determines the number of messages per burst.
- For database adapters, the **FetchUnit** setting determines the number of database rows.

When an input **FetchAs = Integral**, the **Source** is requested to retrieve data once for the entire mapping.

When an input **FetchAs = Burst**, the data is retrieved once per burst.

For example, a database has the rows sorted in reverse chronological order by date. You want to map only the first row because that row contains the most recent data. Set the input **FetchAs = Integral** and **FetchUnit = 1**. Only the first row in the database is accessed.

Using different **FetchUnit** values for the same data object on multiple input cards has limitations.

FetchUnit with multiple input cards

In a case where more than one input card in a map references the same data object, the values of the **FetchAs** and **FetchUnit** card settings must be the same on all cards that reference that data object. Only one instance of the schema component structure is stored with the compiled map, therefore the card settings that control that component structure must be the same on all cards using that data object. The schema component structure stored with the map has one number that represents the maximum instances of that component. The **FetchUnit** setting changes this number.

For example, a map contains two input cards that both reference the **ContactFile** data object. This data object may be the card object (as specified with the **TypeTree** (**Contact.MTT**) and **Type** (**ContactFile Data** settings, as shown below) or may be a component of the card object.

To achieve desired data retrieval results, you must set the **FetchAs** and **FetchUnit** values the same on both cards that use the same type (**ContactFile**).

If different **FetchAs** and **FetchUnit** values are specified for the same data object on two different input cards in the same map, the **FetchAs** and **FetchUnit** values specified in the first card are used.

GET

The **GET** settings define all of the input data settings.

Source

The **Source** card setting identifies the source of the input data. The data may be in a file, from a specific application, in a database, from a message queue, or from other possible sources. Defining the adapter source specifies where to go to get the data and specifics on which data to get.

The meaning of the various adapter source settings is specific to the adapter being used to get the input data.

The maximum size of a file that a map can process in integral mode is 2 GB. In burst mode, the maximum size is much larger, but can only be 2 GB in one burst.

Source > FilePath

When **Source = File**, the **FilePath** setting specifies the name and path of the input data file. Enter the filename or browse your file system to select the input file.

Source > Command

When **Source** = *various resource adapters*, the **Command** card setting provides the adapter-specific commands to connect to the data source and retrieve the input data. For example, if the input data is coming from a file, the **Command** specifies the name of the input file. If the input data is coming from a database, the **Command** might include the database name, user ID, password, and so on.

The command syntax for the source and target adapter commands is detailed in the adapter-specific documentation. The maximum length of a source or target adapter command is 2000 bytes. Limitations to the maximum adapter command length may apply to some adapters.

DatabaseQueryFile

When **Source** = **Database**, the **DatabaseQueryFile** settings are used to specify the database specific settings: the name of the desired database/query file, the database name, and the query.

DatabaseQueryFile > Database

Specify the database that contains the desired query definition.

DatabaseQueryFile > Query

Specify the name of the desired query.

DatabaseQueryFile > File

Specify the database/query file (.mdq) that contains the desired source database and query definition.

Transaction

The **Transaction** settings include the input data **OnSuccess**, **OnFailure**, and **Scope** settings.

OnSuccess

The **OnSuccess** setting specifies the location of the source data when a map, a burst, or a card successfully completes. The availability of the **OnSuccess** setting is adapter-specific.

The **OnSuccess** setting works with the **Scope** setting, which defines when to apply the action specified with the **OnSuccess** setting.

Value

Description

Keep

The input data should *not* be removed from its source.

For example, if the input is a messaging adapter, and the map successfully completes, do *not* remove the messages from the message queue.

KeepOnContent

The input data should not be removed unless it has no content.

For example, if the input is a file, and the map completes successfully, but the input file had no data (it was empty), delete the input file. Otherwise, keep the input file.

Delete

Delete the input data or remove it from its source.

For example, if the map completes successfully, remove the messages from the message queue.

OnFailure

The **OnFailure** setting specifies the location of the source data if the map, burst, or card does not successfully complete.

Note: The exact meaning and availability of the options for this setting is adapter-specific. See your adapter documentation for more information.

The **OnFailure** setting works with the **Scope** setting, which defines when to apply the action specified with the **OnFailure** setting.

Value

Description

Rollback

Any changes made during map processing are rolled back and the original state of the data is restored.

If the map, burst, or card does not complete successfully, rollback any changes that were made to the source data during map execution.

For example, if the source of the input data was a stored procedure in a database that changed the contents of one or more tables, such as with an INSERT or UPDATE statement, if the map fails, the **Rollback** option would restore the affected tables to their original state.

Commit

If the map, burst, or card does not complete successfully, commit any changes that were made to the source data during map execution.
If the map fails, send the message produced for the output card to the specified message queue.

For example, if the source of the data was a message on a message queue and the message is removed from the source message queue, regardless of whether the map succeeds or not.

Scope

The **Scope** card setting tells the source or target adapter at what point during map execution to apply **OnSuccess** and **OnFailure** actions. The default setting is **Map**.

Value

Description

Map

Apply the **OnSuccess** and **OnFailure** settings based on the success or failure of the entire map execution (which may be one or more bursts).
In a RUN map, the **Map** scope applies the **OnSuccess** and **OnFailure** actions based on the success or failure of the top-level map.

Burst

Apply the **OnSuccess** and **OnFailure** settings based on the success or failure of each burst.
In a RUN map, the **Burst** scope applies the **OnSuccess** and **OnFailure** actions based on the success or failure of the map that is specified on the **RUN** function.

Card

Apply the **OnSuccess** and **OnFailure** settings based on the success or failure of this input. In this case, when a rule fails or validation fails, the source adapter applies the **OnFailure** action. If validation completes successfully for the input card, the source adapter applies the **OnSuccess** action.

- [FetchAs and Scope actions](#)

FetchAs and Scope actions

The FetchAs setting and the Scope setting affect when an OnSuccess or OnFailure action occurs.

If an input **FetchAs** setting is Integral, any failure action causes the entire map to fail.

Table 1. Effect of Scope setting when FetchAs setting is Integral

Scope	Action specified by OnSuccess or OnFailure setting occurs:
Map	After all bursts
Burst	After each burst
Card	After input validation for that input for each burst

Table 2. Effect of Scope setting when FetchAs setting is Burst

Scope	Action specified by OnSuccess or OnFailure setting occurs:
Map	After all bursts
Burst	After the first burst
Card	After input validation for that input for the first burst

TargetRule output card settings

The **TargetRule** settings define how the output data is routed. The data object specified in the output card defines what data is output.

To define card-specific output data routing, define the **TargetRule** settings for each output card. **TargetRule** settings specify where to send the data, and what to do when an error occurs. An output card represents an output target. The content of an output is a *data object* such as a table of customer records in a database, or a message that contains a business transaction, or a file of records.

The content of an output card is defined with the **CardName**, **Schema**, **Type**, and **File** settings. These settings define what the data is, but does not specify how to route the data. The **Target**, **Retry**, and **Scope** card settings are some of the **TargetRule** settings that define how the output data is routed for each output card.

- [TargetRule Setting overrides](#)
- [PUT](#)
- [FilePath](#)
- [Target > Command](#)
- [DatabaseQueryFile](#)
- [DatabaseQueryFile > File](#)
- [DatabaseQueryFile > Database](#)
- [DatabaseQueryFile > Table](#)
- [Transaction](#)
- [OnSuccess](#)
- [OnFailure](#)
- [Scope](#)

TargetRule Setting overrides

Output card settings are stored in the compiled map. It is possible to override the setting for each option from both the Command Server and the Integration Flow Designer, which are supplied with the Design Studio.

Execution commands can be used to override the map settings or card settings compiled into the compiled map when a map is run. See the execution commands list for a complete description of the override commands.

The **TargetRule** card settings and their override execution commands are listed in the table below:

TargetRule Card Setting

Override Execution Command

Backup

Not available

Target

Available (-OA), (-OD), (-OE), (-OF), (-OM)

OnSuccess

Available (-O...X)

OnFailure

Available (-O...B)

Retry

Available (-O...Rcount:interval)

Warnings

Not available

Scope

Not available

PUT

The **PUT** settings define all of the output data settings.

FilePath

When **Target = File**, the **FilePath** setting specifies the name and path of the output data file. Enter the filename or browse your file structure to select the output file.

Target > Command

The **Command** setting provides the adapter-specific commands to connect to the data target and process the output data.

For instance, if the output data is being put into a file, the **Command** specifies the name of the output file. If the output data is being sent to a message queue, the **Command** might include the name of the message queue and any other setting required to connect to the message queue and put the message there.

The command syntax for the source and target adapter commands is detailed in the adapter-specific documentation.

DatabaseQueryFile

When **Target = Database**, the **DatabaseQueryFile** settings are used to specify the database specific settings: the name of the desired database/query file, the database name, and the query.

DatabaseQueryFile > File

Specify the database/query file (.mdq) which contains the desired target database and query definition.

DatabaseQueryFile > Database

Specify the database that contains the desired query definition.

DatabaseQueryFile > Table

Specify the database/query file (.mdq) which contains the desired target database and query definition.

Transaction

The **Transaction** settings include the output data **OnSuccess**, **OnFailure**, and **Scope** settings.

OnSuccess

The **OnSuccess** setting specifies the location of the target data when a map, a burst, or a card completes successfully. The availability of the **OnSuccess** setting is adapter specific. The default value is **Create**.

The **OnSuccess** setting works in conjunction with the **Scope** setting, which defines when to apply the action specified with the **OnSuccess** setting.

Value

Value	Description
Create	Output data should be sent by the target adapter to its target. For example, if the map completes successfully, the message adapter should put the message on the specified queue.
CreateOnContent	Output data should be sent only if content exists. For example, a message adapter should put the message produced on the specified message queue. If no data was produced, no message should be sent.
!Create	Regardless of whether or not data content is produced, output data is not to be sent to its target. This option is available for temporary data storage as a map runs.
Append	The output data should be appended to an existing data file. If the data file does not exist, it should be created.

OnFailure

The **OnFailure** setting specifies the location of the target data if the map, burst, or card does not complete successfully.

The exact meaning and availability of the options for this setting are adapter specific. The default value is **Rollback**.

The **OnFailure** setting works in conjunction with the **Scope** setting, which defines when to apply the action specified with the **OnFailure** setting.

Value

Value	Description
Rollback	Any changes made during map processing are rolled-back and the original state of the data is restored. If the map, burst, or card does not complete successfully, reverses any changes made to the target data during map execution.
Commit	If the map, burst, or card does not complete successfully, commit any changes that were made to the target data during map execution. If the map fails, send the message produced for the output card to the specified message queue.

Scope

The **Scope** card setting tells the source or target adapter at what point during map execution to apply **OnSuccess** and **OnFailure** actions. The default setting is **Map**.

Value

Value	Description
Map	Apply the OnSuccess and OnFailure settings based on the success or failure of the entire map execution (which may be one or more bursts).
Burst	Apply the OnSuccess and OnFailure settings based on the success or failure of each burst.
Card	Apply the OnSuccess and OnFailure settings based on the success or failure of this output. In this case, when a rule fails or validation fails, the target adapter applies the OnFailure action. If validation completes successfully for the output card, the target adapter applies the OnSuccess action.

Input and output card settings

The **Retry** and the **Backup** settings appear on both input and output cards.

- [Retry](#)
- [Retry > Switch](#)
- [Retry > MaxAttempts](#)
- [Retry > Interval](#)
- [DocumentVerification](#)
- [Backup settings](#)
- [Backup > Switch](#)
- [Backup > When](#)
- [Backup > BackupLocation](#)

- [BackupLocation > Directory](#)
 - [Directory > Value](#)
 - [BackupLocation > Filename](#)
 - [BackupLocation > FileName > Action](#)
 - [BackupLocation > FileName > Value](#)
-

Retry

The **Retry** settings are defined for each input and output card. The Retry setting is compiled into the map, but may be overwritten by the execution settings defined in the Integration Flow Designer and the **AdapterRetry** (-Y) execution command.

If adapter errors occur (for example, failure to connect) and Retry Switch = On, the adapter process is retried at the interval specified with the **Interval** value up to as many times as defined with the **MaxAttempts** setting.

The **Retry** settings specify the number of times and the interval at which to attempt access an unavailable source or target.

Define the **Retry** settings for each input (or output) card as required.

Retry > Switch

The **Switch** setting determines whether **Retry** for the specific adapter is **ON** or **OFF**. The default value of the **Switch** setting is **OFF**.

Value	Description
OFF	Retry is inactive. No attempt is made to access an unavailable source or target.
ON	Retry is active. Attempts are made to access an unavailable source or target for the number of attempts specified with MaxAttempts and at the time interval specified with Interval .

Retry > MaxAttempts

The **MaxAttempts** setting specifies the maximum number of times to try access to a source or target for a map. The default value for the **MaxAttempts** setting is 0.

Value	Description
nnnnn	A number that represents the maximum number of times an attempt is made to access the source or target for a map. After the maximum number of attempts has been attempted without access, the map returns a return code and message. Valid entries are integers from 0 to 65535.

Retry > Interval

The **Interval** setting specifies the interval (in seconds) at which each attempt to access the source or target is made. The default value is 0.

Value	Description
n	A number that represents the number of seconds at which each attempt to access the source or target is made.

DocumentVerification

For non-Xerces input schemas, use the **DocumentVerification** setting for the supplemental parsing (validation) of XML documents during the standard data validation process. In addition to the standard data validation process that takes place when a map runs, enabling this setting instigates an external library to validate XML data according to the Schema or DTD selected.

At the output card level, you can choose the specific situations in which the verification process takes place.

- [Use of DocumentVerification](#)
 - [Use of DocumentVerification for output data](#)
-

Use of DocumentVerification

To use DocumentVerification for input or output, you must enable the **MapAudit > SettingsAudit > Data** setting and add the data types to the Data Audit Settings view.

Except for the Well-Formed option, the DocumentVerification card setting must not be used with Xerces schemas on input because it initiates an extra and unnecessary validation process.

When you use DocumentVerification, the schema object that is associated with the Document Type must have the following properties set:

- The Document Type property must be XML.
- The Document Type > Metadata value must be set to either DTD or Schema.
- A DTD or Schema file location must be specified (Document Type > Location).

In the map input card, the following properties must be set:

- The Schema > Type is set to the object you associated with the Document Type as described earlier.
- The SourceRule > GET > Source > DocumentVerification setting is set to Always, OnSuccess, OnFailure, or Well-Formed.

Based on these settings, when the map runs, an external parser (document verification program) validates the XML document. If there are no errors, the map completes successfully. If there are errors, the map fails and the error information is written to an audit or trace file (when enabled).

The supported formats are XML with a DTD and XML with an XML Schema.

DocumentVerification settings

Because this setting is at the card level, you can set different options for different input and output cards.

Schema Validation

(Default setting) The option is turned off, meaning that only the standard data validation process takes place.

Always

Validates an XML document whenever an instance of an object with XML as the Document Type property is found at the card level. This XML document validation takes place in addition to the standard data validation process by using an external validation program.

OnSuccess

Validates an XML document when an instance of an object has XML as the Document Type property at the card level and is considered valid during the standard data validation process. This supplemental validation takes place in addition to the standard data validation process by using an external validation program.

OnFailure

Validates an XML document only when an instance of an object has XML as the Document Type property at the card level and fails validation during the standard data validation process. This supplemental validation takes place in addition to the standard data validation process by using an external validation program.

Well Formed

For Xerces schemas, disables validation of the XML document against the Schema and instead parses it to verify that it is well-formed.

Ignore Undefined Constructs

Ignores any elements that are not defined in the Xerces schema or native XML schema. Undefined elements would otherwise fail well-formed parsing with an Element not found error. Ignored elements are not accessible for any type of mapping and are not valid input for the PACKAGE and CPACKAGE functions.

Use of DocumentVerification for output data

For output data, the **DocumentVerification** option only works if the MapAudit Settings Audit > Data setting is enabled for the output. This is because output data is only validated (by means of the standard validation process) if the MapAudit Settings Audit > Data map setting for the output is enabled and the output item is specified under **Data Audit Settings** in the **Organizer** as shown:

If the MapAudit Settings Audit > Data map setting is enabled for an output card, standard data validation occurs during the audit process, and the external verification program can run when specified.

Backup settings

The **Backup** settings, when enabled, control when, where, and how the data for input and output cards should be copied to a specified backup file.

You can define the **Backup** settings for each input or output card as needed.

Backup > Switch

Use the **Switch** setting to activate the **Backup** feature.

Value

Description

OFF

Data for the card is not copied to a backup file.

ON

Data for the card is copied to a backup file based on the subsetting values selected.

Backup > When

The **When** setting specifies when a backup file is created. The default setting is **Always**.

Value

Description

Always

Always create a backup of the data.

OnError

Create a backup only if a burst fails to complete successfully.

Backup > BackupLocation

The **BackupLocation** value indicates that backup data is copied to a file. There are subsettings that enable you to select information about the file, such as where and when it is created.

BackupLocation > Directory

BackupLocation Directory enables you to specify the directory into which the backup file is written. The file is written to the **Map** directory by default.

Value

Description

Map

Creates the backup file in the map directory.

Custom

Specifies the directory location for the backup file.

When you select Custom, a **Value** field appears where you can enter the file path.

Directory > Value

If you selected the **Custom** value for the BackupLocation Directory setting, type the file path in this field or browse for a file.

BackupLocation > Filename

The BackupLocation Filename setting is used to specify the type of naming scheme used for the backup file. **Custom** is the default setting.

Value

Description

Custom

Specifies a custom name for the backup file. When you select Custom, the **Value** field appears into which you can enter the custom file name.

Unique

Generates a unique audit log name in the following format:

Mer_mapname_processkey_mapcounter_hostname.extension

Run_mapname_processkey_mapcounter_hostname.extension

The prefix "Mer" is used with top-level maps while "Run" is used for run maps.

mapname

The executable map name.

processkey

Unique value for each process.

mapcounter

Unique value for each map instance (within the same process).

hostname

The host name of the computer where Transformation Extender is running.

extension

is the backup filename extension, which is *.Inn.bak* (for input cards) or *.Onn.bak* (for output cards), where *nn* represents the card number.

BackupLocation > FileName > Action

The **BackupFileAction** setting specifies how data should be written to the backup file. **Create** is the default setting.

Value

Description

Create

Creates the backup file at the start of map execution. If a file exists with the same file path, it is overwritten. If more than one burst is used, data will be appended to the same backup file.

Append

If a file exists with the same file path, the data is appended to the existing file. Otherwise, a new file is created. If more than one burst occurs, data for each burst is appended to that same backup file.

BackupLocation > FileName > Value

If the BackupLocation Filename setting is **Custom**, enter the file name in this field.

Effects of modifying a schema

When you modify a schema or any type definitions in that schema, the next time you open a map that uses that schema, the map is updated.

If you save the schema without making changes to it, you are still prompted to update the map.

Changing a type that is used for a card

If you delete, move, or rename a type, and that type is used as the type of a card, the type will appear as having no components. Viewing the **Properties** of the type shows that the type is an **Invalid type**.

SyntaxCard

The SyntaxCard setting defines the output card as a syntax card.

Configure an output card as a syntax card by using the card settings and selecting Yes for the SyntaxCard setting. The default value for the SyntaxCard setting is NO.

When you specify a card as a syntax card, the map passes any syntax items it created to the next card. A map uses a syntax card so that the delimiter structure defined for a variable syntax object persists as the data flows from one output card to the next output card in the map. The card that follows a syntax card inherits the delimiter structure that is set by a syntax card.

A syntax card works with other output cards in the following way:

1. Output card 1 is defined as a syntax card. The syntax card maps most elements to =NONE and maps only the syntax items. For example, an EDI X12 element delimiter value can be represented as ="*".
2. Output card 2 maps only the data. Output card 2 does not map the syntax items that are defined in output card 1. But the mapping process maps the data according to the output card 2 structure as if the syntax items have been set.

For example, when producing EDI such as X12 and EDIFACT documents, syntax elements define the syntax of the components in the EDI schema. The syntax can vary for each input. Without a syntax card, you must define the syntax elements in the envelope to use the syntax structure later on. When bursting, a map can produce many EDI transactions. Without a syntax card, the only way to initialize the syntax is to include an envelope in each burst. The service that reads the output from the map must strip the envelopes from the output.

When your map includes a syntax card, the syntax need not be initialized with each burst. Also, it is not necessary to strip the envelopes from the output.

There are additional requirements and considerations for using the SyntaxCard setting.

You might have a requirement to pass the syntax value on additional output cards that do not follow directly after the card that is configured as the syntax card. In this scenario, you must also set the SyntaxCard setting to Yes on all of the output cards in between the syntax card and the card that uses the syntax values defined on the syntax card. For example, output card 1 is the SyntaxCard. You need to use the syntax values on output card 3. Therefore, you must also set the SyntaxCard setting to Yes on output card 2.

On the syntax item in the schema where the Delimiter property value is Variable, you must set the FIND setting to NO. When the FIND setting is set to NO, the predefined or default value is mapped.

You must use the same schema for the output card defined as the syntax card and the output card that uses the syntax values defined on the syntax card.

Although you are not restricted from doing so, you should use only the output card that is defined as the syntax card to define the syntax items.

Editing a card

When you edit input and output cards, the map rules that were not affected by the change are retained. You can change anything specified in a map card: the card name, the schema, the type of the card, and the data source or target.

- [Changing a card name](#)
- [Changing the type of an input card](#)
- [Changing the type of an output card](#)
- [Changing the schema of a card](#)

Changing a card name

When a card name is changed, any reference to that card in any map rule is automatically changed to the new name.

In an example that has an input card named **Input**, which is referenced in two places in the map rule, if you change the name of the input card from **Input** to **GrowthReport**, the new card name is automatically changed.

The input card is edited and the name of the card is changed from **Input** to **GrowthReport**.

The instances of **Input** in the map rule are automatically changed to **GrowthReport**. All instances of the card name are changed in all map rules where it appears.

Changing the type of an input card

You can change the type (**TypeName**) of an input card. Note that when you change the type of an input card, if an object within that card is referenced in a map rule, the object name might now be invalid.

If a map rule contains the name of an object on an input card:

```
Flower:ProgressReading:Input
```

The type of the input card is **GrowthReport** and it is changed to **ProgressReading**, which is a component of **GrowthReport**. The object names in the map rule are now incorrect. **Flower** is a component of **ProgressReading**, which is now the type of the card. After the type of the card has changed, the correct name for **Flower** is **Flower:Input**.

To correct the object name in the map rule, drag the object name to the map rule again. If you change the type of a card and you have many map rules in that card, you might want to use search and replace by selecting the Replace option under the Transformation Extender Search tab in the Search window navigated from the Search menu on the menu bar instead.

Not all modifications to an input type will affect map rules. For example, if you add a component to an input type, map rules are not affected. Also, if you delete an input component that is not referenced in a map rule, the map rules are not affected.

Changing the type of an output card

You can change the type (**TypeName**) of an output card. Changing the type of an output card retains the map rules for the objects that still exist within the card object, even if they are in a different order.

- If there was a map rule on an object that no longer exists in the card object, that map rule becomes unresolved.
- If there is a map rule on a data object that still exists in the card object, the map rules are retained.

Even though the order of the output components changed, the map rules stay intact.

Changing the schema of a card

The schema of a card can be changed. Optionally, you can change the schema of all the cards in that map and in referenced maps that use the previous schema. When the schema for a card is changed, you can choose to replace all references to that schema with the new schema.

In some situations, you would not want to replace all references to the schema. For example, if only the input of the selected schema had changed, you would not want the output to change.

Map events

When you run Design Server Maps using the Design Server runtime REST API, the related Map execution events are sent to the specified Elasticsearch server.

Map execution events include these events:

- Execution details
- Map names
- Elapsed execution times
- Map start and stop information
- Map and its adapters return codes
- Error details

Map rules

Map rules are used in output cards of a map to define how output data is built. Map rules are required for output data generation. If there is no map rule, there is no output data. When no output is desired for a given data object, a map rule of =NONE is required.

A map is not complete until each rule cell has a map rule, or appears dimmed (the map rule is unavailable). Empty rule cells generate a build error when the map is built.

A map specifies how to generate a particular output data object, or multiple independent objects, each of a particular type. Each output card has a rule column in which you enter map rules. Color-coding of map rules provides visual ease of use in formulating and understanding rules. When color-coding is enabled, syntax errors in map rules are underlined.

Map rules are validated only when that map rule is changed.

- [Map rules overview](#)
- [Rule editor](#)
- [Color coding of map rules](#)
- [Map rules as expressions](#)

Map rules overview

Map rules are used to:

- Convert an object in the input data from one value to another value in the output data
- Perform conditional logic
- Execute another map to produce file with specific contents

A map rule is an expression that evaluates to data. The maximum length of a map rule is 32K characters (including non-printable characters).

Non-printable characters are not displayed in the Rule editor, but display in the rule cell on the output card in the Map editor as `o`.

Rule editor

Use the Rule editor display and enter map rules for the selected object on the output card in the Map Designer..

When you select a rule icon on the output card in the Map Designer, the rule displays in the Rule editor. Changes to map rules are made and committed in the Rule editor.

Non-printable characters do not display in the Rule editor, but display in the rule cell on the output card in the Map editor as `□`.

- The default Commit key in the map rule is `Enter`. To enter a new line hold down the `Ctrl` key and press `Enter`. If `Tab` is the commit key, simply press `Enter`.
- To enter the object name of any data object, drag it from the map card into the Rule editor.

You can type in extra spaces in a map rule to make it easier to read. Line feeds are not required, but may improve the way the rules are displayed. In addition, you can create a new line where you want one.

Text literals enclosed in quotation marks may not contain new lines.

Color coding of map rules

To aid in readability, map rules can be colored with user-configured colors. When color-coding is enabled, each element of a map rule can be configured with a different color.

The elements of map rules are categorized to allow color specification by category. Colors are assigned to:

- Unknown
- Syntax
- Number
- Type Name
- Map Name
- Reserved
- Literal
- Comment
- Background
- [Configuring colors for map rules](#)

Configuring colors for map rules

Enable color-coding by selecting a color for each category of map rule elements.

To configure map rule colors

1. Click Window > Preferences > Transformation Extender > Type Tree > Group Window.
The Group window appears.
2. From the Rule Color Specifications list, select one of the categories.
3. Click the color button.
The Color window appears.
4. Select a color from the color palette and click OK in the Color window and OK in the Group window.

To disable color coding

Remove the color selections you have made through the Group window.

Map rules as expressions

A map rule is an expression that evaluates to data. An expression is any valid combination of literals, data object names, operators, functions and map names. The expression in a map rule generates the desired data. For example:

- The expression Name Field:Input evaluates to the value of the data object Name, which is a subtype of Field, which is a component of the data object Input.
- The expression 500 evaluates to a data object with a value of 500.
- The expression COUNT (Record:OrderFile) evaluates to a data object that has a numeric value equal to the count of the Record objects in the data object OrderFile. A map rule has a maximum length of 32K characters. Rules in older maps might start with an equals (=) sign.

Some examples of simple expressions are shown in the following table.

Expression

This expression uses:

"ABC Company"
a literal
City Field:Record
a data object name
City Field:Record + State Field:Record + ZipCode Field:Record
operators (plus signs) and data object names
UPPERCASE (City Field:Record)
the UPPERCASE function and a data object name
RecordMap (Record:InputFile)
a map name (RecordMap) and data object names

An expression can be complex and have simpler expressions nested within it. Examples of more complex expressions using nested expressions are shown below:

```
IF (Qualifier = "ST", Address, NONE)
LOOKUP (ProductCode:LookupList, ItemCode:Data =
ItemCode:LookupList)
EXTRACT (Record:Input, Company Field:Record:Input = "My
Company")
COUNT (EXTRACT (Row:Input, Company Column:Row:Input = "My
Company"))
```

These examples use the functions **IF**, **LOOKUP**, **COUNT**, and **EXTRACT**. These functions are specific to the IBM Transformation Extender suite of products. Functions have their own syntax. For more information on functions, see the Functions and Expressions documentation.

- [Map rules define how to generate data objects](#)
- [Map rules are evaluated independently in sequential order](#)
- [Generating no output](#)
- [Text literals as NONE](#)
- [Special characters in map rules](#)
- [Syntax errors in map rules](#)
- [Data object names in map rules](#)
- [Card names in map rules](#)
- [Component names in map rules](#)
- [Index in map rules](#)
- [Partition in map rules](#)
- [Comments in map rules](#)
- [Unavailable rule cells](#)
- [Use ellipses](#)

Map rules define how to generate data objects

The object of an output card may be composed of components and partitions. Each object on the output card has a corresponding rule cell, where map rules that transform that data are entered.

A map rule evaluates to a particular data object. The data object is either a group type or an item type. The following rule evaluates to the data object **Record**, which is a group:

```
= LOOKUP (Record:Input, Company:Record:Input = "ABC Printing")
```

A map rule for a data object must evaluate to a data object with the same subclass. Objects with an Item Subclass of **Date & Time** must evaluate to an object with an Item Subclass of **Date & Time**.

For example, the output item ShipDate has the Item Subclass of **Date & Time**. The following map rule is valid when the data object ShipToDate also has an Item Subclass of **Date & Time**:

```
ShipDate = ShipToDate:PO:Input
```

The output CompanyName is defined with an Item Subclass of Text. The following map rule:

```
CompanyName = "Pop's Pizza Co."
```

assigns the value of a text literal to the output.

A text literal is text enclosed in double quotes and is interpreted as a text item. A rule that evaluates to something other than text, for example, CompanyName = 24 would be invalid.

To view the type properties of a data object in a card, right-click the type in the input or output card and click Show in Properties on the context menu. If the Properties view is shown, select the map rule to update the Properties view to show the properties of the object.

A map rule on a group type must evaluate to a data object of the same group. For example, the following map rule on the output group **Record** must evaluate to the group type **Record**:

```
Record = LOOKUP (Record:Input, Name:Record:Input = "Kathy")
```

The map rule is for the group type **Record**, so it must evaluate to the type **Record**. The rule shown above does evaluate to **Record** because the LOOKUP function returns a **Record**. The rule would be invalid, for example, if it evaluated to the group **Invoice**.

Map rules are evaluated independently in sequential order

Each map rule is evaluated independently from all other rules. Each output object with a rule is created independently from the others.

Map rules are evaluated in sequential order. When a map is executed, the first rule of the first output card is evaluated. Next, the second rule of the first output card is evaluated, and so on, until the last rule of the first output card. Then, the first rule of the second output card is evaluated, and so on, to the last rule of the last output card.

Generating no output

Each map rule cell requires a map rule. If an output is optional, you may not want to generate the output for that data object. For map rules for data objects for which you do not want to generate output, enter the following:

= **NONE**

The reserved word **NONE** specifies to generate none of the given output. The word **NONE** is case-sensitive.

You don't have to type in the rule = **NONE**. The command **Insert NONE if Empty** puts this rule into the empty rule cells on the active output card, including those empty cells that cannot be seen by the current view of the card.

Text literals as **NONE**

The use of "" as text literals in a map rule evaluates to "none". For example, the following two rules are equivalent:

```
= EXTRACT (Record:Input, Store:Record:Input = "")  
= EXTRACT (Record:Input, Store:Record:Input = NONE)
```

Special characters in map rules

To enter a Hexadecimal value of a special character in a map rule, enter the Hex value for the characters in double brackets (open and close carets).

For example, to enter the text value <WSP>, enter the Hex value for the open caret <<3C>>, and then the literal values for the rest: WSP

```
<<3C>>WSP>
```

Syntax errors in map rules

Syntax errors are reported when the map rule is committed. The default "commit" key is the **Enter** key. Pressing **Enter** on a map rule commits that rule and reports any syntax errors.

The cursor is automatically positioned at the first syntax error. When color coding is enabled, syntax errors can be configured to display with underline.

Data object names in map rules

The following examples show different kinds of data object names that can appear in a map rule. The following is a partial list of the possible data object names. For a detailed list of object names, see the Functions and Expressions documentation.

Data Object	Appears in map rules as:	Description
Card names	OrderFile	User-defined name of input or output card A unique name for each card in the map provides the ability to distinguish between card objects that may be used in more than one card.
Component	Record:OrderFil e Quantity:Record: OrderFile	Separated by colon (:) Interpret as "of": " Quantity of Record of OrderFile " Quantity is a component of Record . Record is a component of OrderFile .
Index	Note[3]:Report	Specific occurrence appears in square brackets [n] immediately after the type A particular occurrence of a data object indicated by the index
Partition	Customer<>Rec ord	Separated by <> When a type is partitioned, its subtypes, or partitions, appear in the card with <>.

Data Object	Appears in map rules as:	Description
Shortened data object name	Company Field::Input	A period (.) is used as an abbreviation of ellipses (...). The shortest object name that is different from all other object names in the same map is used. Some object names may not include ellipses, because the shortest unique name is the entire name.

Card names in map rules

A map rule can see the entire card data object. For example, if an input card name is **OrderFile**, the name for the entire card object is **OrderFile**.

The name of the card appears as the first data object of the card. In the example shown below, the card name is **OrderFile**. This name appears last in each object name:

Record:OrderFile

Quantity:Record:OrderFile

A unique name for each card in the map provides the ability to distinguish between card objects.

For example, if two input cards are defined by the same type, these two input cards may exist with the same type, but their card names are different. If the name of the first card is **Input1** and the second input card is **Input2**. If a component **Record** exists as data objects in both these cards, the difference is distinguishable **Record** in one input and **Record** in the other because the **Record** types in the different cards have different names **Record:Input1** and **Record:Input2**.

Component names in map rules

A component in a data object name is indicated by a colon (:). The colon is the separator between a data object and the group or category type it is a part of (or the card name it is contained in).

For example, **Record** is a component of the data object **OrderFile**. The object name for **Record** is **Record:OrderFile**.

When you see a colon (:), you can read it as "of". For example, the name of the component **Quantity** in the input card is **Quantity:Record:OrderFile**. You can read the data object name for **Quantity** as "The **Quantity** of **Record** of **OrderFile**".

Index in map rules

An object name can refer to a particular occurrence of a data object indicated by the index.

For example, the third **Note** in a series of **Note(s)** has an index of 3. In a data object name, an index appears in square brackets immediately after the type being indexed. The indexed third **Note** of **Report** is **Note[3]:Report**.

Partition in map rules

When a type is partitioned, its subtypes or partitions appear in the card.

In a map rule, a partition is referenced by the symbol <>.

For example, the partition **CallRecord** of the group type **SetofCallRecords** is **CallRecord<>SetofCallRecords**.

Think of a partition as a particular kind of the partitioned type. For example, in the name **Detail<>Record**, you can think of it as a **Detail** kind of **Record**.

An object name includes component references and partition references. For example, the partition **Back** of **OrderRecord**, on the input card **File**, appears in a map rule as **Store Field:CallRecord<>OrderRecord:File**

Comments in map rules

You can add inline comments to map rules. Comments do not affect how map rules are evaluated. A comment may appear anywhere in a rule, as long as it does not separate object names.

A comment is identified with the start characters /*. While the end characters */ can be present, they are not required. If the end characters are not present, everything up to the end of that rule is the comment.

The following map rule has two comments, which are indicated by blue text:

```
= EXTRACT (Record:Customers /*Extract the Records */,
State Field::Customers = "FL" /* Get only the customers in
Florida */) 
```

Using comments for map rules, or portions of map rules, is a convenient way to retain complex and lengthy map rules for future use without deleting them.

Unavailable rule cells

The output of a map produces each output card object. To produce a card object, the required objects that compose the card object must be produced. There is a rule cell for each object within the card object.

If an output object is an item, you must supply a rule for it. If an output object is a group, you have two options. Either you can enter a rule for the object as a whole, or you can expand the view of that object in the output card and enter rules for each data object.

Defining how each component of the output object as a whole must be produced is the equivalent of defining how to create the entire output object.

If map rules are entered for each component of the output object, map rules may not be entered for the entire output object. When you expand the entire output object, notice that the rule cell next to the output object is automatically unavailable (grayed out). Entering a rule for the entire output object would be redundant, because the individual components of the output object already have map rules.

When you expand a group that has a component range maximum greater than 1, the rule cells on the components of that group are unavailable.

Use ellipses

Enables the **Use ellipses** display option for subsequent object names dragged to the Rule editor. The **Use ellipses** options causes lengthy object names to display the shortest possible object name.

Rather than having the entire name appear, unique portions of the name are replaced with a period (.) that is used as an abbreviation of ellipses (...).

Complete object name: **Company Field:Record:Input**

Shortened name: **Company Field:.:Input**

If the **Use ellipses** option is enabled, the shortest object name that is different from all other object names in the same map is used. Some object names may not include ellipses, because the shortest unique name is the entire name.

The **Use ellipses** option does not affect object names that already exist in map rules; it only affects object names that are entered after the option is enabled.

Search functionality

The search functionality in the Map designer allows to search for a specific text within all rules in a current executable map and all its corresponding functional maps.

- [To search in a Map using Search icon](#)

To search in a Map using Search icon

1. On the Map toolbar, click on the Search icon.
The Map search panel opens.
2. In the Search panel, enter the text you want to search.
3. In the Search in section, select component to search within the input and output cards or select Map Rules to search in the Rule Editor.
4. Check the Search Submaps check box to search in the child maps.
5. Click the Search button.
The relevant search results display.

Formulating map rules

Implement map rules to build specific output results.

- [When input data is missing](#)
- [Mapping an input item to an output item](#)
- [Mapping a group to a group](#)
- [Outputs with maximum range greater than 1](#)
- [Indexing an output](#)
When you know the number of objects to generate for each map execution, index the number of occurrences of that output object. To *index an output* is to generate a specific number of occurrences of that output.
- [Mapping to multiple occurrences of a group](#)
- [Indexing an Input](#)
- [Mapping from a floating component type](#)

When input data is missing

If you map an input to an output, but the input is missing, either the output will have no content or it will not exist. The existence of the output depends on how the output is defined.

- [When portions of an input series are missing](#)
- [When an output evaluates to NONE](#)

When portions of an input series are missing

When portions of an input series are missing, the output to which it is mapped built according to the **Track** property of the type in the Type Designer. The **Track** property indicates whether to only track the components that have content (**Track = Content**) or all components, including those that do not have content (**Track = Places**).

If both the input data object and the output data object have the **Track = Places** property, the place holder is generated on output, even if there is no data. Example output data with a missing third item would be:

YES,YES,,YES

If the input data object has the **Track = Content** property, the empty occurrences of that data object are not mapped and the empty occurrences are not built. Example output data with a missing third item would be:

YES,YES,YES

When an output evaluates to NONE

If an output object evaluates to "none", the information appearing in the output depends on how the output has been defined. The following table defines how output is build when the output occurrence evaluates to "none":

Table 1. When a required item evaluates to "None"

Context	Output is built in this manner:
Always	The item's initiator and terminator are built.
Occurs within a fixed explicit group	If a special value for NONE has been assigned, NONE is built as the special value. Otherwise, NONE is built as the pad characters for the Padded to Length specified.
Occurs within a delimited explicit group	The group's delimiter appears in the output. If a special value for NONE has been assigned, NONE is built as the special value. Or, if a pad property is specified as applied in any context, NONE is built. Otherwise, NONE is not built.
Occurs within an implicit group	If the group has a delimiter, the group's delimiter appears in the output. If a special value for NONE has been assigned, NONE is built as the special value. Or, if a pad property is specified as applied in any context, NONE is built.

Table 2. When an optional item evaluates to "None"

Context	Output is built in this manner:
Always	The item's initiator and terminator are not built.
Occurs within a fixed explicit group	If a special value for NONE has been assigned, NONE is built as the special value. Otherwise, NONE is built as all pad characters for the specified length.
Occurs within a delimited explicit group	The group's delimiter appears in the output if data with the same delimiter follows.
Occurs within an implicit group	No output is built.

Table 3. When a required group evaluates to "None"

Context	Output is built in this manner:
Always	The group's initiator and terminator are built. Required components are built.

Table 4. When an optional group evaluates to "None"

Context	Output is built in this manner:
Always	The group's initiator and terminator are not built.
Occurs within a fixed explicit group	Each component is built, down to its items, building NONE for each item according to the special value for NONE and pad property specifications.
Occurs within a delimited explicit group	The group's delimiter appears if data with the same delimiter follows.
Occurs within an implicit group	No output is built.

An item within a fixed group must have the Padded to = Fixed Size property specified, or the size minimum must equal the size maximum. If an item is required, and content is specified as required, an output invalid message appears in the summary trace file.

Mapping an input item to an output item

You might want an output item to be equal to some input item. In that case, drag the input item into the rule cell of the output item.

- [Automatic item conversions](#)
- [Intermediate item conversions](#)
- [Concatenating text strings](#)

Automatic item conversions

When mapping an item to an item and the input items have the same subclass as the output items, the item type properties of the input item are automatically used as the item type properties of the output item.

For example, an item type is defined according to the following:

Item Subclass = Number

Presentation = Decimal with a required decimal separator

Pad = Yes

Padded to = Fixed Size

Padded to > Length = 7 bytes

Pad > Value = 0

Pad > Justify = Right

If an input type is defined as a decimal number, with no pad character, no separator, and two decimal places, the following table shows an example of input and output types of an item type definition.

Property	Input Type	Output Type
Presentation	Decimal	Decimal
Separators	No	Yes
Places > Decimal Min	2	2
Pad > Value	(none)	0
Padded to	(none)	7
Justify	(none)	Right

The input data is automatically converted to the format of the output data object. The output data will be in decimal format, right justified, padded to 7 bytes with a pad character of 0.

An input item mapped to an output item must have the same **Item Subclass**. For example, both items must have an **Item Subclass = Number** or both must have an **Item Subclass = Text**. To convert the **Item Subclass** of an item, use the conversion functions, such as **NUMBERTOTEXT**, **TEXTTODATE**, and so forth.

Intermediate item conversions

When using an item with an **Item Subclass** of **Number** or **Date & Time** as a text argument, use the **TEXT** function to convert that item to a text form.

Concatenating text strings

To concatenate text, use the plus symbol (+) between the objects to be concatenated.

For example, the output **Name Field** is defined in the map rule as the input **FirstName Field**, a space, and then the **LastName Field**.

```
=FirstName Field:EmployeeInfo + " " + LastName  
Field:EmployeeInfo
```

Mapping a group to a group

Desired output of an input group may be the same as an input group. If the output group is the same type and is defined in the same schema as the input group, drag the input group into the map rule cell of the output group. If the input group is a partition of the output group, you can drag the partition into the rule cell of the output group.

If the input group and output group are different types, you cannot drag the input group to the output group. Either expand the output group down to its item components, or use a functional map.

- [Automatic conversion of syntax items](#)
- [Mapping to a group with a maximum range of 1](#)

Automatic conversion of syntax items

Syntax items are automatically converted from an input group to the syntax items of the output group. If the delimiters, terminators, and release characters of a data object are variable, these values are defined in the input data stream. To convert these values to other values, you may drag the input group to the output group map rule. The output group is built using the default values or data values for the syntax items.

Mapping to a group with a maximum range of 1

Generally, do not drag a group from the input to the output. This is only possible if the input group and the output group are the exact same group, which is not very common. When the range of a group has a maximum of 1, expand that output group and map to its components.

Outputs with maximum range greater than 1

For outputs with a component range maximum greater than 1 - (for example, (1:10) or (s)), define how many of that particular output to generate. There are two possible situations:

- the number of objects to create is known, or
- the number of objects to create is based on the number of some other object in the input or output data.

When the number of objects to create is known, index that number of output occurrences.

When the number of output objects is based on the number of some other object, create a map rule that maps the output according to the number of occurrences of an input.

Indexing an output

When you know the number of objects to generate for each map execution, index the number of occurrences of that output object. To *index an output* is to generate a specific number of occurrences of that output.

Indexing an output adds an output object to the output card. This output object has the index number in square brackets. For example, [1].

The index number identifies the occurrence of the object. The index number [1] indicates the first occurrence. The index number [2] indicates the second occurrence, and so on. After you have indexed an output, you can enter a map rule for that indexed output. If you have indexed a group, you can then expand the group and add map rules at the component level.

Map to the first occurrence of an indexed object by dragging that object from the input card to the map rule of the item on the output card. Define the second occurrence of the output by dragging the desired object from the input card to the map rule of the item on the output card. If you do not want a third indexed data object, enter the map rule = NONE.

Mapping to multiple occurrences of a group

When an output group has a range with a maximum greater than 1 and you want to map a certain input group to that output group, there are two possibilities:

- If the input group is the *same* as the output group, drag the input object from the input card to the map rule of the object on the output card. The output is created up to its maximum range.
- If the input group is *different* from the output group, use a functional map.
To generate occurrences of an output group based on occurrences of input data, use a functional map.

Indexing an Input

When a map rule refers to a particular occurrence of an input, the input must be indexed. In the map rule of the output object, enter the index number in square brackets [] immediately after the input object you are indexing.

Mapping from a floating component type

If an input card data object is a group that has a floating component type, you can map from the floating component type. A floating component must be distinguishable from every component of the group it is assigned to. On input, a floating component can appear after the initiator and after each component. If a floating component is associated with an input data object, it can appear as the first component.

If a type object has content, a floating component is built as the specified literal after the initiator and after each component. If this feature is used, it is up to the user to include a literal that will validate as an object of the type of the floating component.

You cannot map to a floating component in the output. If you have a floating component in the output data to which you want to map, you must define them as components of the output.

Functional map basics

A functional map is like a subroutine; it maps a portion of data at a time. It takes one or more input objects and generates one output object. For example, you might have a functional map that maps one Message to one row in a database table. Or you might have a functional map that maps one Header and one Detail to one ItemRecord.

The result of the functional map is sent directly to the output card when the type of the output card is:

- A group

- A partitioned item, when more than one of the partitioned type's subtypes has a rule other than the =NONE rule. The functional map evaluates the rules in the order that they are specified, and when the result of a rule is not NONE, the functional map builds the output of that rule as the partition that the rule is on.

A functional map result that returns directly to the output card is not available to the calling map for use in subsequent operations.

When the output of a functional map is an item, you can use the **ASFUNCTION** general function to return the output of the functional map to the calling map instead of to the output card. The calling map can use the returned result in subsequent operations in the same map rule. With **ASFUNCTION**, the result of the functional map returns to the calling map rule when the type of the output card is:

- A non-partitioned item type
- A partitioned item, in one of the following conditions:
 - There is a rule on the partitioned item.
 - All of the subtypes for the partitioned type, except one, have the =NONE rule.

When you do not use **ASFUNCTION** to run a functional map, the item is sent directly to the output card.

- [When to use a functional map](#)
- [Similarities of input and output data](#)
- [Using a functional map](#)
- [Comparing executable and functional maps](#)
- [Syntax of a functional map expression](#)
- [Entering the map rule that references the functional map](#)
- [Input card types of a functional map](#)
- [Input arguments as data objects](#)
- [Output card type of a functional map](#)
- [Using multiple inputs in a functional map](#)

When to use a functional map

The use of functional maps is very common. Almost every executable map created will use at least one functional map.

To map a group in the input to a different group in the output, use a functional map.

For example, use a functional map to map an input row to an output row when the rows are defined differently. Or, use a functional map to map from a file containing many input rows, to generate a file of many output rows with one output row per input row. The first output row would correspond to the first input row, the second output row corresponds to the second input row, and so on.

A map defines how to generate the output data. One important factor to consider in determining when to use a functional map is the presence of an output component with a range of more than one. For example, ranges of (s) or (1:10). The number of this output object to be created is based on the number of some input object.

Another important factor in determining when to use a functional map is when you want to transform the data - mapping from one or more types to a *different* type. In the preceding example of the functional map that maps one **Message** to one row in a database table, the input row and the output row are two different types.

Use a functional map when the number of a certain output group that you want to create is based on the occurrences of some input or output data - and the types are different types.

Similarities of input and output data

Frequently, your input and output data contain very similar information, but the data is arranged differently.

For example, you have a file of invoices from your company's internal application and you want to map it to a file of invoices to send to another company. The input and output invoices are really reflections of one another. The layout of an invoice in the application file may be different from the layout of an invoice in the output file. The important point is that you want to make an output invoice for each input invoice.

Or you may receive health claims in a standard format, and you want to map them to a different standard format. The input claims are embedded in a hierarchical layout where data is not repeated. The output claims may be arranged so that each claim has the same key data repeated: the provider, the patient, and so on. To make an output claim for each input claim, use a functional map. There is a one-to-one relationship between an input claim and an output claim.

A good indication of when to use a functional map is when there is a one-to-one relationship between a group in the input and a different group in the output.

As an example, you have a data source with three input records. A functional map provides a way to map one input record to one output record. The functional map defines the first input record that generates the first output record. Next, the second input record generates the second output record. The third input record generates the third output record. The result is three output records.

Using a functional map

The basic steps to follow for using a functional map are as follows:

1. Determine the need for using a functional map.
2. Determine the input argument(s) of the functional map.
3. Enter the map rule that references the functional map.
4. Create the functional map.
5. Create the input card(s).
6. Create the output card(s).
7. Enter map rules in the functional map.

The Functional Map Wizard eliminates Steps 5 and 6.

- [Example scenario for using a functional map](#)

Example scenario for using a functional map

An example of a functional map is shown below.

A data file contains statistics for different kinds of cars. Each record in the file represents a car. For each car, there are fields for mileage, head room, rear seat, trunk room, and weight. Each field is padded to a specific length. The record is fixed. For example:

Marlin	22	2	27	11	2930
Pacer ZSX	17	3	26	11	3350
Wildcat	22	3	18	12	2640
Fantasia DX	17	3	27	15	2830
La Vella	23	2	28	11	2070
Blue Baron	25	2	26	12	2650
Cavalino	20	4	29	16	3250

The objective is to map this data to a file containing delimited records. The delimited record has only the car name and the weight. In the delimited record, the weight is in decimal format. The desired output data result is:

Marlin,29.30
Pacer ZSX,33.50
Wildcat,26.40
Fantasia DX,28.30
La Vella,20.70
Blue Baron,26.50
Cavalino,32.50

You want to generate one delimited record for each fixed record in the input file.

Input data					Output data
Marlin	22	2	27	11	2930
Pacer ZSX	17	3	26	11	3350
Wildcat	22	3	18	12	2640
Fantasia DX	17	3	27	15	2830
La Vella	23	2	28	11	2070
Blue Baron	25	2	26	12	2650
Cavalino	20	4	29	16	3250

Define the map rule for the output **CarRecord(s)**, with a range of (s). There is an indefinite number of **CarRecord(s)**.

Ask yourself "How many **CarRecord(s)** do I want to generate?" If you want a specific number, the answer would be that number. To generate two **CarRecord(s)**, index two occurrences of **CarRecord**, expand each occurrence, and map to its components. For more information on indexing, see "[Indexing an Output](#)".

If the number of **CarRecord(s)** is based on the number of **FixedRecord(s)** in the input, generate one **CarRecord** for each **FixedRecord**. So, the answer to your question of how many **CarRecord(s)** to generate is, "As many as there are **FixedRecord(s)** in the input."

Note: When you see an output group with a range, it is always a good idea to stop and ask yourself, "How many of these do I want to generate?" The answer will either be:

- A specific number - index the output.
or
- The number is based on the number of some input - use a functional map.

You need a mechanism that takes a single **FixedRecord** and creates a single **CarRecord**. To do this, create a functional map, which is similar to a user-defined function. This functional map maps a single **FixedRecord** to a single **CarRecord**. The map is called for every occurrence of a **FixedRecord** in the input. This generates exactly one **CarRecord** for each **FixedRecord**.

For example, if there are 23 **FixedRecord(s)**, 23 **CarRecord(s)** are generated. The first **CarRecord** corresponds to the first **FixedRecord**. The second **CarRecord** corresponds to the second **FixedRecord**, and so on.

Comparing executable and functional maps

There are two kinds of maps: executable maps and functional maps.

Executable Map

An *executable map* is a map responsible for the totality of your inputs and outputs. The sources and targets of an executable map are entire files, database tables, messages, applications, and so on. Think of an executable map as the main map, the top-level map. Executable maps are compiled and run.

Functional Map

A *functional map* is referenced by another map through a map rule. A functional map maps just a portion of the entire data. Data sources and targets are not specified in the cards of functional maps. Functional maps are not compiled and run.

Note: A given map can be used both as an executable and as a functional map. You may want to test small portions of your data and temporarily use a map you created as a functional map as an executable map on that portion of the data.

If a map is used as a functional map, the input data and output data settings specified in that map are ignored.

Syntax of a functional map expression

Map rules on executable map output cards specify the functional map name. The syntax of a functional map expression is similar to the syntax of a function.

Similar to a function, a functional map has one or more inputs, and one output.

The inputs and the output are data objects.

The syntax of a functional map expression is the following:

```
FunctionalMapName (argument1, argument2,...argumentn)
```

Where **n** is the argument number. The input arguments of a functional map appear between the parentheses in the functional map expression separated by commas.

An input argument of a functional map is any valid expression that evaluates to data, including data objects, functions, and literals. For example, an input argument may be any of the following:

Input argument

Example

data objects

```
FixedRecord:Input
```

functions that evaluate to data

```
EXTRACT (StoreInfo:Inventory, StoreName:Inventory = "Lee  
Furniture")  
INDEX (Record:File)
```

literals

```
"payroll"
```

Each of these expressions evaluates to data. For example, the following expression evaluates to the data object **FixedRecord**:

```
FixedRecord:Input
```

The following expression evaluates to **StoreInfo** data objects:

```
EXTRACT (StoreInfo:Inventory, StoreName:Inventory = "Lee Furniture")
```

The **INDEX (Record:File)** expression evaluates to an integer; the "**payroll**" expression evaluates to text.

For a detailed discussion of functions and expressions, see the Functions and Expressions documentation.

- [Determining the arguments of a functional map](#)

Determining the arguments of a functional map

The input arguments of a functional map are the objects necessary to create *one* occurrence of the output object where the map rule is.

To determine what the input argument(s) of a functional map should be, ask yourself, "What do I need to create this output?" Whatever is necessary in a functional map must be passed to it as an argument. There may be one or more input arguments. The arguments of a functional map can be thought of as triggers. The number of times a functional map is triggered depends on the number of occurrences of the input arguments in the data and the expressions used for each argument.

To determine the input arguments of a functional map, ask yourself what objects are necessary to create *one* occurrence of the output object.

- [Input argument to a functional map evaluates to NONE](#)

Input argument to a functional map evaluates to NONE

If an occurrence of input argument is missing, the functional map is not evaluated.

A file that has no content will still allow the functional map to execute.

For example, in the following functional map expression the **GroupInfo** object is optional with a component range of (0:3):

```
MakeForm (EntryForm:Input, GroupInfo:Input)
```

If there are no occurrences of **GroupInfo** in the source data object, the specified **MakeForm** functional map is not evaluated.

In the following map rule, the **OrderMap** functional map is specified with the (DetailRecord:Order:Input, SummaryRecord:Order:Input) arguments:

```
OrderMap (DetailRecord:Order:Input, SummaryRecord:Order:Input)
```

The data object **SummaryRecord** has an optional component range of (0:1). For the first **Order**, **SummaryRecord** is present. The functional map **OrderMap** is evaluated. In the second **Order**, **SummaryRecord** is missing. The map **OrderMap** is not evaluated for that second **Order**.

To evaluate the entire **OrderMap** if **SummaryRecord** may be absent from the source data, specify the entire **Order** as an input argument, rather than **SummaryRecord**. Specifying the entire **Order** ensures that **OrderMap** is evaluated, even when **SummaryRecord** is missing. Then, in the functional map **OrderMap**, you can still map from **SummaryRecord**.

Entering the map rule that references the functional map

Map rules that reference functional maps are entered in the rule cells on the output card of the executable map. Map rules belong in the rule cell of the data object that requires additional data processing. The number of times a functional map is triggered depends on the number of occurrences of the input arguments in the data and the expressions used for each argument.

This section explains how to manually create a functional map. You can also use the "[Functional Map Wizard](#)".

For the executable map for mapping the car data (see [Example Scenario for Using a Functional Map](#)), the executable map is **CarData**. The source data contains fixed records and the goal is to generate an output file containing one delimited record for each fixed record in the source data.

A functional map on the **CarRecord(s)** data object can generate one **CarRecord** for each **FixedRecord**. A suitable name for the functional map can be **RecordMap**, because the functional map maps one record at a time. In the **CarData** executable map, the map rule for the **CarRecord(s)** data object in the output card would contain an equal sign, the name of the functional map, and a set of parentheses:

```
=ListCars ( )
```

To map the fixed record input object to the map rule, drag the **FixedRecord(s)** data object from the input card into the rule cell inside the parentheses. Commit the map rule by pressing the **Enter** key.

A map rule specifies how to generate that certain output. The map rule on **CarRecord(s)** specifies how to create each **CarRecord**. Basically, the map rule specifies "to generate a **CarRecord** data object, use a functional map named **ListCars** on a **FixedRecord** data object." The functional map **RecordMap** uses the **FixedRecord** to create a **CarRecord**.

The functional map **ListCars** can be created before the executable map. The creation order for the map rule or functional map is not relevant. However, an error occurs at build time when a rule includes a nonexistent map name.

Input card types of a functional map

The input and output cards in a functional map must be in the same order as the data objects are specified in the map rule argument.

The input card type(s) in a functional map must be coordinated with the input argument(s).

For example, if the input arguments evaluate to **Record**, **Claim**, and **LookupFile**, in that order, then the input card types in the functional map must be **Record**, **Claim**, and **LookupFile**, in the same order.

Input arguments as data objects

Input arguments of a functional map can evaluate to **Item** or **Group** type data objects.

Item types

When an input argument of a functional map evaluates to an item type, the type of the corresponding input card must be an item type with the same **Item Subclass** (**Number**, **Text**, **Date & Time**, or **Syntax**).

An input argument might evaluate to a number. If the following input argument evaluates to a number, the corresponding input card type must be an item with an **Item Subclass of Number**:

```
INDEX (Record:File)
```

If the input argument is a text literal (for example, "`payroll`"), the input card type that corresponds to this argument must be an item with an **Item Subclass of Text**.

Group types

When an input argument of a functional map evaluates to a group type, the type of the corresponding input card must be the exact same group. For example, if the second input argument evaluates to **InvoiceSet**, the **TypeName** of input card #2 must be **InvoiceSet**.

If there is one argument of the functional map, there will be one input card. The **TypeName** on the input card must correspond to the data object specified in the argument for the functional map.

Functional maps do not require **InputData** and **OutputData** settings.

When **InputData** and **OutputData** settings exist in input and output cards in a functional map, they are ignored when the executable map is compiled.

Output card type of a functional map

The type of the output card in a functional map is the type whose map rule includes the functional map name. For example, if the functional map name is entered in the rule cell for **CarRecord(s)**, the output of that functional map must be **CarRecord**.

The output of a functional map must match the output data object (where the map rule that calls the functional map is), therefore a functional map has only *one* output card.

A functional map may have only *one* output card.

- [Entering map rules](#)

Entering map rules

Map rules of a functional map are entered to generate the desired output of the functional map. Similar to an executable map, output data is generated in the specified format. This specified format is determined by the type properties of the type on the output card.

Essentially, a function maps a single data object for every data object received from the input.

Using multiple inputs in a functional map

A functional map may require more than one input. For example, you are mapping an input purchase order to an output purchase order. The input **PO** consists of a header and multiple details. The output **PO** is composed of a series of item records. Each **ItemRecord** contains the **PO#** and the company name, which must be mapped from the **Header** of the input **PO**. Each **ItemRecord** also contains item, quantity and price information, which is mapped from a **Detail** in the input.

You need a functional map to generate an **ItemRecord**. The inputs of the functional map would be everything that is necessary to generate one **ItemRecord**: both the **Detail** and **Header** from the input. A suitable name for the functional map may be **MakeItemRecord**.

The input **PO** is composed of one **Header** and multiple **Details**. You want to generate one **ItemRecord** for each **Detail**.

The functional map **MakeItemRecord** takes one **Detail** at a time, along with the **Header**, and creates one **ItemRecord**.

The two input arguments of **MakeItemRecord** are **Detail** and **Header**. You would enter the following map rule for the output **ItemRecord(s)**:

```
= MakeItemRecord (Detail:Input, Header:Input)
```

In the map **MakeItemRecord**, you would create two input cards. The first input card corresponds to the first input argument, **Detail**. The second input card corresponds to the second input argument, **Header**.

- The input arguments must match the input card types, in the same order.
- The order of the arguments does not affect the number of times a rule is evaluated. However, it may affect the order in which the output objects appear and the performance at run time. For more information on how rules are evaluated, see the Functions and Expressions documentation.

In **MakeItemRecord**, map the **PO#** and **Company** from the **Header** into the components of **ItemRecord**. Map the **Item**, **Qty**, and **Price** from the **Detail**.

- [Input objects are triggers](#)

Input objects are triggers

A functional map is triggered by the presence of the input objects that are specified as the arguments of the functional map. Each time a new combination of input objects is evaluated, the functional map is executed.

The functional map **MakeItemRecord** has two inputs: **Detail** and **Header**. One output **ItemRecord** is produced for each combination of **Detail** and **Header** in the input. In this example, there is one **Header** for many **Details**. The same **Header** object is sent to the functional map again and again with each new **Detail**.

For a complete explanation of how map rules are evaluated, see the Functions and Expressions documentation.

Rename a functional map

The functional map of an execution map can now be renamed. To rename a functional map follow the below procedure:

1. In the Map workspace list, right click on the functional map you want to rename and select Edit option.
Edit Map window opens.
2. In the Edit Map window, enter the new name in the Name field.
3. Click OK.
The Edit Map window closes.

After renaming the functional map, the renamed map name should be reflected in below places:

1. Rule editor
2. Connection Rule
3. Map tab name
4. Map Navigator list
5. Map workspace list

Map settings

Map settings specify properties for map executions that are not specific to a particular input or output. These include settings for auditing a map, tracing data content, performance settings, validation settings, and map-specific warnings and errors.

Map settings are defined for a map during the design phase and are compiled with the map. Except for BurstRestart, the map settings compiled with a map can be overwritten at run time.

The default map settings can be defined or changed in the Map Designer options for Run Options by clicking Window > Preferences > Transformation Extender > Map > Run Options.

- [MapAudit settings](#)
 - [MapTrace settings](#)
 - [WorkSpace settings](#)
 - [Century settings](#)
 - [Validation settings](#)
 - [Retry settings](#)
 - [Warnings settings](#)
 - [CodePageFallback](#)
 - [MapRuntime setting](#)
-

MapAudit settings

MapAudit settings specify options for creating and storing map audit logs.

If the **MapAudit Switch** is set to **ON**, the audit log is created and can be appended. In addition, there are flexible sub-switches that allow for the audit log to be created or appended based on the following conditions:

- When a map fails.
- If the map has an error.
- If the map has a warning or error.

- [MapAudit > Switch](#)
 - [MapAudit > BurstAudit](#)
 - [BurstAudit > Data](#)
 - [BurstAudit > Execution](#)
 - [MapAudit > SummaryAudit](#)
 - [SummaryAudit > Execution](#)
 - [MapAudit > SettingsAudit](#)
 - [SettingsAudit > Data](#)
 - [SettingsAudit > Map](#)
 - [MapAudit > AuditLocation](#)
 - [AuditLocation > Directory](#)
 - [AuditLocation > FileName](#)
 - [AuditLocation > Sized](#)
-

MapAudit > Switch

The **Switch** setting enables or disables the map audit feature. The default value for the **Switch** setting is **OFF**.

Value	Description
OFF	Audit data for the map will not be created.
ON	Audit data for the map will be created based on the values of the related settings.

MapAudit > BurstAudit

Expand this setting to define the audit log BurstAudit Data and BurstAudit Execution settings.

BurstAudit > Data

The **Data** setting is used to include or exclude data information in the audit log. The burst audit data information includes a burst count, the card number, the object index, the audit log status code, the object level, the size of the object, and the object name. You can specify that you want to audit a particular object by including that object on the Data Audit Settings view.

The BurstAudit Data settings are listed in the following table. The default setting is **Never**.

Value	Description
Always	Write data to the audit log, regardless of the map execution outcome.
Never	Do not log data information.
OnBurstError	If the outcome of the execution of the burst meets a failure condition, write the burst data to the audit log.
OnBurstWarningOrError	If the outcome of the execution of the burst is a warning or meets a failure condition, write the burst data to the audit log.

- [Data > SizeValidation](#)

Data > SizeValidation

When BurstAudit Data is set to **Always**, **OnBurstError**, or **OnBurstWarningOrError**, the **SizeValidation** field becomes available. The input trace file will report when data fails to meet size requirements for the type.

TooLong TooShort

When you choose this option and the minimum or maximum size requirements are not met during data validation, one of the following size-specific messages will be indicated in the input trace file:

```
"fails minimum size requirements"  
"fails maximum size requirements"
```

Wrong Size

This is the default option for this setting. If the minimum or maximum size requirements are not met during data validation, it will be indicated in the input trace file with a generic message similar to the following example:

```
"data at offset xx ('DATA...') is the wrong size for the TYPE X' (02)  
(typename)"
```

When using the **TooLong TooShort** value for the BurstAdit Data **SizeValidation** setting and the data fails the minimum size requirement, the input trace file entry will look similar to the following example:

```
(Level 1: Offset 41, len 6, comp 5 of 7, #1, DI 00000005:  
Data at offset 41 ('SYNTAX...') fails minimum size requirements  
TYPE X'0007' (LongSyntaxItem ROOT).
```

BurstAudit > Execution

The **Execution** value determines whether to include audit execution information in the audit log.

The execution audit log includes the command used to run the map from a command line, the number of input and output objects, and information about data sources, data targets, work area size, and the time it took to run the map. The execution log does not include information about restarts.

The BurstAudit Execution settings are listed in the following table. The default setting is **Never**.

Value	Description
Always	Include burst audit execution information in the audit log, regardless of map execution outcome.
Never	Do not include burst audit execution information in the audit log, regardless of map execution outcome.
OnBurstError	If the burst execution outcome meets a failure condition, write burst audit execution information to the audit log.
OnBurstWarningOrError	If the burst execution outcome is a warning or meets a failure condition, write burst audit execution information to the audit log.

MapAudit > SummaryAudit

Expand this setting to define the summary audit **Execution** settings. Summary audit provides audit information on map execution without the detailed burst information.

SummaryAudit > Execution

The **Execution** value determines whether to include summary audit execution information in the audit log. Summary audit provides audit information on map execution without the detailed burst information.

The SummaryAudit Execution settings are listed in the following table. The default setting is **Always**.

Value	Description
Always	Include summary burst audit execution information in the audit log, regardless of the map execution outcome.
Never	Do not include summary burst audit execution information in the audit log, regardless of the map execution outcome.
OnError	If the map execution outcome meets a failure condition, write the execution information to the audit log.
OnWarningOrError	If the map execution outcome is a warning or meets a failure condition, write the execution information to the audit log.

MapAudit > SettingsAudit

Expand this setting to define which settings are included in the audit log.

SettingsAudit > Data

The **Data** value determines whether to include data settings in the audit log.

The SettingsAudit Data settings are listed in the following table. The default setting is **Never**.

Value	Description
Always	Include data settings in the audit log, regardless of the map execution outcome.
Never	Do not include data settings in the audit log, regardless of the map execution outcome.
OnError	If the map execution outcome meets a failure condition, include data settings in the audit log.
OnWarningOrError	If the map execution outcome is a warning or meets a failure condition, include data settings in the audit log.

SettingsAudit > Map

The **Map** setting is used to include or exclude the settings from the map in the audit log.

The SettingsAudit Map settings are listed in the following table. The default setting is **Never**.

Value	Description
Always	Include map settings in the audit log, regardless of map execution outcome.
Never	Do not include map settings in the audit log, regardless of map execution outcome.
OnError	If the map execution outcome meets a failure condition, write map setting information to the audit log.
OnWarningOrError	If map execution outcome is a warning or meets a failure condition, write map setting information to the audit log.

MapAudit > AuditLocation

The **AuditLocation** setting is used to specify where the audit information is stored.

The availability of additional settings (such as Directory, FileName, Action, or Sized) varies according to the AuditLocation setting.

The MapAudit AuditLocation values are listed in the following table. The default value is **File**.

Value	Description
File	Audit information is stored in a file. There is a maximum of 259 characters in the path and filename.
Memory	Audit information is stored in memory, and appended to the last set of data returned during map execution. The Memory value is used when the map is run by the RUN function or by an API.

When a map is run by a Command Server or Launcher, the **Memory** setting is ignored. The audit log is created in a file in the map directory, as specified with the AuditLocation **FileName** setting.

AuditLocation > Directory

The **Directory** setting is used to specify the same directory as the map or a user-defined directory. The default value for the **Directory** setting is **Map**.

Value	Description
Map	The same directory as the map.
Custom	The directory you specify using the Value setting.

- [Directory > Value](#)

Directory > Value

If you selected the **Custom** value for the AuditLocation > Directory setting of the MapAudit setting, enter the full path of the directory in the Value field, or browse to select the directory.

AuditLocation > FileName

Use the FileName setting to specify a default or custom name for the audit log file.

The default value is Unique.

Value

Description

Default

The file is automatically assigned an audit log file name based on the compiled map name with the .log file name extension.

Unique

Generates a unique audit log name in the following format:

Mer_mapname_processkey_mapcounter_hostname.log

Run_mapname_processkey_mapcounter_hostname.log

The "Mer" prefix is used with top-level maps while "Run" is used for run maps.

mapname

The executable map name.

processkey

Unique value for each process.

mapcounter

Unique value for each map instance (within the same process)

hostname

The host name of the computer where Transformation Extender is running.

You cannot view an audit log file that has a unique name from the Data Audit Settings view.

Custom

Use the FileName > Value setting to specify the custom name of the audit log file.

Use the Custom value if the log files produced for one map trigger another map, so that each log file triggers another instance of that other map. Be sure to specify different directories for the log files.

The number of characters allowed for the log file name is contingent on the number of characters in the directory path. The total number of characters in the path name and the file name is limited to a maximum of 255 characters.

- [FileName > Action](#)
- [FileName > Value](#)

FileName > Action

The **Action** setting for the AuditLocation > FileName setting of the MapAudit setting is used to create a new audit log file or to append the audit log to an existing audit log file. The default value for the **Action** setting is **Create**.

Value

Description

Create

The audit log file is created each time the map is run.

Append

The audit log is appended to an existing audit log file each time the map is run. If no audit log file exists at the start of map execution, an audit log file is created.

FileName > Value

If you selected the **Custom** value for the AuditLocation > FileName setting of the MapAudit setting, enter the file name in the Value field.

The amount of characters allowed for the audit log filename is contingent on the amount of characters in the directory path. The total number of characters in the path name and the file name is limited to a maximum of 255 characters.

AuditLocation > Sized

When **AuditLocation = Memory**, audit information is stored in memory and appended to the last set of data returned during map execution. The memory setting can be used only when the map is run by the **RUN** function or an API. Audit data is prefixed with a text number indicating the size of the audit data, followed by a <SP>.

Value	Description
ON	The audit data is sized.
OFF	Not sized: the audit data is returned with no size.

MapTrace settings

MapTrace settings specify properties for a trace file used to diagnose invalid data or incorrect type definitions.

- [MapTrace > Switch](#)
- [MapTrace > TraceLocation](#)
- [TraceLocation > Directory](#)
- [TraceLocation > Filename](#)

The FileName setting (Map Settings.>.TraceLocation.>.FileName) specifies how a map trace file is named. A map trace file with a default name is overwritten the next time the map is traced. Specify a custom name or a unique name to compare map trace files across multiple runs of the same map.

- [MapTrace > InputContentTrace](#)
- [InputContentTrace > CardNumber](#)
- [InputContentTrace > StartObject](#)
- [InputContentTrace > EndObject](#)
- [MapTrace > RulesTrace](#)
- [RulesTrace > CardNumber](#)
- [MapTrace > SummaryContentTrace](#)
- [MapTrace > Format](#)

The MapTrace.>.Format setting specifies whether the contents of the trace file are binary or text. Binary files are smaller than text files and are likely to have less impact on performance (however, never use tracing in a production environment).

MapTrace > Switch

Define whether map tracing is enabled. A trace file can be used for diagnosing invalid data or incorrect type definitions.

Value	Description
OFF	Trace data for the map will not be created.
ON	Trace data for the map will be created.

MapTrace > TraceLocation

Use the **TraceLocation** settings to specify where the trace file will reside. The **TraceLocation** is **File**, which specifies that trace information is written to a file as specified with the **Directory** and **FileName** settings.

TraceLocation > Directory

The **Directory** setting is used to specify the same directory as the map or a user-defined directory. The default value for the **Directory** setting is **Map**.

Value	Description
Map	The same directory as the map.
Custom	The directory you specify using the Value setting.

- [Directory > Value](#)

Directory > Value

If you selected the **Custom** value for the TraceLocation.>.Directory setting of the MapTrace setting, enter the full path of the directory in the Value field, or browse to select the directory.

TraceLocation > Filename

The **FileName** setting (Map Settings > TraceLocation > FileName) specifies how a map trace file is named. A map trace file with a default name is overwritten the next time the map is traced. Specify a custom name or a unique name to compare map trace files across multiple runs of the same map.

Default

The default map trace file name is *mapname.mtr*.

Custom

Specify a name for the map trace file in the **Custom > Value** field.

Unique

Automatically creates a unique map trace file name in the format:

`Mer_mapname_processKey_mapCounter_hostname.mtr`

MapTrace > InputContentTrace

Specify options of which input data is traced. The default value for the **InputContentTrace** setting is **All**.

Value

Description

OFF

Trace input data for the map will not be created

All

Trace all inputs.

Card

Trace a single input. Specify the input map card by typing the number of the map card into the value field of the **CardNumber** setting.

Range

Trace a range of input objects. Specify the beginning object by typing the number of the object into the value field of the **StartObject** setting. Specify the ending object by typing the number of the object into the value field of the **EndObject** setting.

InputContentTrace > CardNumber

To trace a single input trace, enter the card number. For example, to include only messages about card 2 in the trace file, enter 2.

InputContentTrace > StartObject

Enter the beginning object number to be traced. For example, to trace data objects 1000 to 2000, enter 1000 as the **StartObject** value.

InputContentTrace > EndObject

Enter the ending object number to be traced. For example, to trace data objects 1000 to 2000, enter 2000 as the **EndObject** value.

- [Using ranges for InputContentTrace](#)

Using ranges for InputContentTrace

The range specified for objects is not precise. The intention is to reduce the number of objects in the trace file, not to specify precise objects. To trace the first object, specify a **StartObject** range of 0, not 1.

The object number is not provided in the trace file. Providing a range is an attempt to trace less than the whole, not a precise range of objects.

MapTrace > RulesTrace

The **Detail** setting specifies options of which output data is traced. The default value for the **Detail** setting is **OFF**.

Value

Description

OFF

Trace output data for the map will not be created.

All

Trace all objects built.

Card

Trace a single output. Specify the output map card by typing the number of the map card into the value field of the **CardNumber** setting.

RulesTrace > CardNumber

To trace a single output trace, enter the card number. For example, to include only messages about card 2 in the trace file, enter 2.

MapTrace > SummaryContentTrace

The **SummaryContentTrace** setting is used to place a summary of the validation status of each input and the build status of each output after the map is built into the trace file. The default value for the **SummaryContentTrace** setting is **OFF**.

Value	Description
OFF	Do not include a summary in the trace file.
ON	Include a summary in the trace file.

MapTrace > Format

The MapTrace > Format setting specifies whether the contents of the trace file are binary or text. Binary files are smaller than text files and are likely to have less impact on performance (however, never use tracing in a production environment).

Option	Description
Text	Generates a text trace file. This is the default format.
Binary	Generates a binary trace file. Use the dtxmaptrace utility to convert a binary trace file to text. ASSERT messages in the binary trace file indicate which data type (from the schema) Transformation Extender is validating.

Related reference

- [dtxmaptrace utility](#)

WorkSpace settings

WorkSpace settings specify the location (file or memory) of the workspace and **PageSize** and **PageCount** settings for configuring the page size. **WorkSpace** settings allow you to optimize the speed of map processing.

When a map is run, a paging scheme to access information about the data as efficiently as possible is used. Portions of both working data and actual data are paged, as needed. The workspace can be a combination of files and memory, or simply memory-based. When file-based workspace is used, the amount of memory accessed at a given time depends on the number of pages and the count of pages configured for a map to run.

The **WorkSpace** settings specify the location (file or memory) of the workspace and **PageSize** and **PageCount** settings for configuring the page size.

The default value for the **WorkSpace** setting is **File**.

Value	Description
File	A work file is created for each input card. Use the Directory setting to specify the directory.
Memory	A memory-based workspace is created.

The **WorkSpace** settings allow you to optimize the speed of map processing.

- When the size of the input data a map is processing is large, file-based workspace using default paging is recommended (**Workspace = File**).
- When small amounts of data are processed, smaller page sizes and memory-based workspace is recommended (**Workspace = Memory**).
- When using burst processing, adjust the workspace configuration for the size of burst data.
If a particular map card is not referenced by any map rules, no work area for that card will be created, and the audit log will indicate that the work area is not found.
- [WorkSpace > PageSize](#)
- [WorkSpace > PageCount](#)
- [WorkSpace > Directory](#)
- [WorkSpace > WorkFilePrefix](#)

WorkSpace > PageSize

The **PageSize** setting specifies the size of a memory page to be used for work files. For maps that include a large number of type references, you may want to increase the page size. The default value for the **PageSize** setting is 64.

Value	Description
n	

Enter a number that represents the size of a memory page in kilobytes. The page size should be a multiple of 2, between 2 and 1024.

WorkSpace > PageCount

The **PageCount** setting specifies the number of memory pages to use for work files. For large data files, you may want to increase the number of pages used. The default value for the **PageCount** setting is 8.

Value

Description

n

Enter a number that represents the number of memory pages to use for workspace. Valid numbers are 1 to 9999 (an integer greater than 0).

WorkSpace > Directory

The **Directory** setting is used to specify the same directory as the map or a user-defined directory. The default value for the **Directory** setting is **Map**.

Value

Description

Map

The same directory as the map.

Custom

The directory you specify using the **Value** setting.

- [Directory > Value](#)

Directory > Value

If you selected the **Custom** value for the **Directory** setting of the **WorkSpace** setting, enter the full path of the directory in the **Value** field, or browse to select the directory.

WorkSpace > WorkFilePrefix

Use the **WorkFilePrefix** setting to select a convention for naming work files.

The default value is **Unique**.

Value

Description

MapName

The name of a work file is the base name of the compiled map file. The extension is **.Inn**, for an input, and **.Onn** for an output, where *nn* is the card number. For example, the work file for input card number 3 has the extension **.I03**.

Unique

Generates unique work files with no conflict between work files each time a given map is run. If this option is turned on, work files are automatically deleted.

- [WorkFilePrefix > Action](#)

WorkFilePrefix > Action

Use the **WorkFileAction** setting to save or delete the work files. The default value for the **WorkFileAction** setting is **Delete**.

Value

Description

Delete

Delete the work files after a map is executed. All work files are deleted, except those being saved for reuse, as specified on the input cards for each map.

Create

Save the work files based on the **Value** and **FilePrefix** settings.

Century settings

Century settings specify the current or sliding century.

- [Century](#)
- [Century > CCLookup](#)

Century

The **Century** setting specifies the current or sliding century. The default value for the **Century** setting is **Current**.

Value	Description
Current	Missing century numbers are derived based on the current system date.
Sliding	Enables SlidingCentury options. Missing century numbers are derived based on the value of the CCLookup setting.

Century > CCLookup

The **CCLookup** setting specifies the first year in a 100-year range used to derive a missing century for a date item, using CCYY format. The default value for the **CCLookup** setting is **1950**.

For example, if the value of the **CCLookup** setting is **1960**, a YYMMDD formatted date between 1960 and 1999 is interpreted as having a century date of **19**, while dates between 2000 and 2059 are interpreted as having a century date of **20**. In this case,

- 600101 is assigned a century of 19
- 580101 is assigned a century of 20

Value	Description
n	Enter a number in CCYY format that represents the start of a 100-year range. Valid numbers are 1 to 9999.

Validation settings

Validation settings specify how strictly the data is validated by specifying response behavior for types of validation errors.

- [Validation](#)
- [Validation > OnValidationError](#)
- [Validation > RestrictionError](#)
- [Validation > SizeError](#)
- [Validation > PresentationError](#)
- [Validation > Ignore Component Rules](#)

Use the Ignore Component Rules map setting to run a map without validating component rules, for example, when your data does not conform exactly to a particular schema.

Validation

During map execution, the input data is validated to assure it matches the definition of that input. Specify how strictly the data is validated by specifying response behavior for types of validation errors.

Value	Description
Standard	Standard validation occurs when the map is run.
Custom	Enables custom validation options. Define these options in the OnValidationError , RestrictionError , SizeError , and PresentationError settings.

Validation > OnValidationError

Under certain circumstances, multiple errors are detected that occur during validation. The **OnValidationError** setting defines where to halt map execution. With either setting, if an error is found in the input, no output objects are built.

Value	Description
Stop	Only validate through the input card where the first error occurs. Stop executing the map after validating that input card, even if there are multiple input cards.
Continue	Continue validating data until all errors are found. Even when validation finds data errors in a given input card, map execution continues with the next input card.

Validation > RestrictionError

Define whether to validate the data with restrictions at run time. For example, if you are running some test data and you do not need the item values to match the restrictions, you might wish to ignore the restrictions.

Value

Description
Ignore
Ignore restrictions during validation.
Validate
Validate with restrictions.
Ignore (no warnings)
Allows restrictions to be fully ignored, in all situations other than when ignoring is required for object identification, such as when an item is partitioned by a restriction list.
Restrictions of syntax items are not ignored, because these are used in determining the value of the syntax object.

Validation > SizeError

Define whether to validate the data with size at run time.

Value	Description
Ignore	Ignore minimum and maximum size of items that appear in delimited data during validation.
Validate	Validate with size.
For example, if an item is defined as having a minimum of 3 bytes, but it only has 2 bytes in the data, the mapping process ignores the fact that it only has 2 bytes and accepts the data as valid.	

Validation > PresentationError

Define whether to validate the data with presentation settings at run time. For example, if a number is defined as an integer, but it appears in the data as a decimal number, the mapping process ignores the fact that the number appears as a decimal and accepts the data as valid anyway.

Value	Description
Ignore	If the item has an invalid presentation, the system cannot interpret it as an object of the defined type. Even if the data fails the item presentation check, the mapping process handles the data as if it were valid. Although the mapping process attempts to map the item, it might not produce the correct output.
Validate	Validate with presentation settings.

Validation > Ignore Component Rules

Use the Ignore Component Rules map setting to run a map without validating component rules, for example, when your data does not conform exactly to a particular schema.

The Validation map setting Ignore Component Rules prevents a component rule from being evaluated when either:

- The group does not include a component that has a group identifier attribute.
- The component rule occurs on a component that is after the component that has the group identifier attribute.

Retry settings

The **Retry** map settings define whether to try again if compiled map files, work files, audit log files, or trace files that are needed during map execution cannot be opened.

Specify the **Interval** and **MaxAttempts** to define how many times and at what interval attempts are made to access an unavailable file.

If adapter errors occur (for example, failure to connect) and Retry Switch = On, the adapter process is retried at the interval specified with the Interval value up to as many times as defined with the **MaxAttempts** setting.

- [Retry > Switch](#)
- [Retry > MaxAttempts](#)
- [Retry > Interval](#)

Retry > Switch

The **Switch** setting enables or disables **Retry** options. The default value for the **Switch** setting is **OFF**.

Value	Description
OFF	Disables Retry options. Only one attempt to access an unavailable file. If this fails, a return code and message (Could not open file) are returned by the map.

ON

Enables **Retry** options. Retry is enabled for the number of attempts specified by the value of the **MaxAttempts** setting at the interval specified by the value of the **Interval** setting.

Retry > MaxAttempts

Use the **MaxAttempts** setting to specify the maximum number of attempts to access an unavailable file referenced by the map. The default value is **10**.

Value

Description

n

Enter a number that represents the maximum number of attempts to access an unavailable file referenced by the map. After the maximum number of attempts has been exhausted unsuccessfully, a return code and message are returned by the map. Valid entries are integers 0 to 65535.

Retry > Interval

Use the **Interval** setting to specify the interval to wait between attempts to open a system file referenced by the map. The default value for the setting is 0.

Value

Description

n

Enter a number that represents the number of seconds to wait between attempts to access input data referenced by the map. Valid entries are integers 0 to 9999.

Warnings settings

Warnings settings specify that the map should fail when any of the selected warning codes are returned. The default value for the **Warnings** setting is **Every**.

Enabling the **Warnings** options impacts the transaction behavior specified for each card in the map.

If a map execution warning code occurs, the map fails after processing of that map is finished.

Maps called (from the failed map) with a **RUN** function, **PUT** function, or **GET** function will still process, and the behavior defined in that map's card settings for Transaction OnSuccess and Transaction OnFailure occurs.

If any warning is set to **Fail** the map, and the map fails due to the cause associated with the selected warning, the execution audit includes the map-failed message with the appropriate warning code preceded by a minus sign, for example: **Map failed on warning (-27)**.

Define your audit settings for each map, as the audit log of the main map may not reflect all failed map information if the failed maps are called with a RUN, PUT, or GET function.

Value

Description

Every

Map fails when any of the warning codes are returned. This setting treats every warning the same way, depending on the value of the **Return** setting. For **Return**, you can select **Warn**, **Ignore**, or **Fail**.

Custom

Map fails only when the user-selected warning codes are returned. The specific warnings are available only if the value of the **Warnings** setting is **Custom**.

- [Other Warnings settings](#)
- [Warnings > Return](#)

Other Warnings settings

If you selected the **Custom** value for the Warnings setting, specific warnings settings appear. For each warnings setting, you can select the specific map response to the warning. The default value for the each of these additional settings is **Warn**.

Value

Description

Warn

Issue a warning if any warning is returned.

Ignore

Ignore all warnings if any warning is returned.

Fail

Fail the map if any warning is returned.

Warnings > Return

The **Return** setting specifies the response for warnings. The default value for the **Return** setting is **Warn**.

Value	Description
Warn	Issue a warning if any warning is returned.
Ignore	Ignore all warnings if any warning is returned.
Fail	Fail the map if any warning is returned.

CodePageFallback

Use the CodePageFallback map setting to specify the code page conversion fallback behavior when mapping between objects with a specific data language set and there is a character found that is not defined in the target character set during the conversion process. The default value is Skip.

Value	Description
Skip	Skip a character that cannot be converted to the target code page. When a character is skipped, the result is that the content of the target data is a character shorter than the content of the source data.
Substitute	If you select Substitute, the Substitute Value property opens. Specify any value for the Substitute Value property. At run time, if a string contains a character that is not defined in the target character set, the mapping process replaces the character with the substitute character or string that you specified in the Substitute Value property. If you do not specify a value for the Substitute Value property, the mapping process replaces the character that is not defined in the target character set, with the International Components for Unicode (ICU) substitution character that is specified in the conversion tables.

MapRuntime setting

The MapRuntime setting determines the runtime platform of an executable map. The default value is Transformation Extender. The alternate option, DataPower, pertains to an WebSphere DataPower SOA Appliance.

To design and create DataPower maps in the Map Designer, which is during the pre-production phase, use the DataPower runtime option to build and test them. After you have successfully tested the maps, you can deploy the maps to a DataPower appliance during the production phase. Maps that are built specifically for a DataPower appliance do not support all of the features of the Transformation Extender. The limitations on the appliance are described in the [Limitations of appliances](#) topic. The DataPower run time requires a compiled map (.dpa) and the input files for the map to run.

Configuring bursts

For input card settings that have a **FetchAs** mode of **Burst**, the type of input object (as defined in the **Source** setting) and the **FetchUnit** work together to achieve the desired results.

For example, if an input type is one message, the **FetchUnit** should be 1. If the input type is a series of messages, the **FetchUnit** can be any integer.

- When the **Source** is a File of Messages
When the map is run, the first message in the file is found, and converted as specified. Consecutive messages in the file are encountered, and converted. And so on until all of the messages in the file are processed. A burst is identified by the definition of the input type - a message in this case.

If the target is a file, each message produced by a burst is appended to the same file. If you define the target as a message queue, each message produced is placed on the output queue after each burst.

- When the **Source** is a Queue of Messages
Change the source of the input to be a message queue. In this case, fetch the first message on the queue, convert it to the message, and route it to its target - which can be a file or another message queue, for example. Then, fetch the next message on the input queue and process it, and continue to fetch messages one at a time until the input message queue has no more messages.

Retrieval of data prior to the first burst is specific to the adapter.

- [Batching logical messages](#)
- [Using bursts for large files](#)
- [Configuring bursts for error recovery](#)
- [Configuring the RUN function and bursts](#)
- [Logical or adapter-specified FetchUnits](#)

Batching logical messages

You may process bursts of multiple messages. Under some conditions, this may improve performance. To do this, the logical input (that is, the type of the input card) is a series of messages. Then, adjust the **FetchUnit** to the quantity of messages to process for a given burst.

Automatic Logical Unit Adjustment: If the input type has one component, the range maximum is automatically adjusted for that component to match the FetchUnit. For example, the input type is a group with one component (could be a series of messages with a range maximum of 10) and the FetchUnit = 20. The series range maximum is automatically adjusted to 20 when the map runs.

If the input type has no components, or has more than one component, the type of that input is not automatically adjusted. In this case, if the amount of data retrieved for a burst does not match the input type definition, the input is either invalid or a burst warning is issued "Input valid, but unknown data found".

Using bursts for large files

To merge three inputs into one, the application takes the first row from the first input, the first row from the second input, and the first row from the third input, and constructs a new row from data of all three input rows. Then, take the second row of the first input, the second row of the second input, and the second row from the third input to construct a new second output row. Then, do the same with each matching input row, constructing a matching output row until all the input is consumed.

To configure the map to run coordinated bursts, set up each input's **CardMode** to **Burst** - and use the same **FetchUnit** for all three inputs.

When more than one input **CardMode** is set to **Burst**, and there is data for one burst and not another, the type definitions for the inputs determine the response. If the card object is required, and that card is set to Burst, data must be present if data for any other card set to **Burst** mode is present. In this case, the burst (and subsequently the map) will fail and the error message "One or more inputs is invalid" is displayed".

Configuring bursts for error recovery

Errors that occur during burst processing might be recoverable, or they can cause only a burst to fail, or they can be severe enough to cause the entire map to fail. Your card and map settings control many of these failure points.

If the following error occurs...	Card fails	Burst fails	Map fails
Any source connection fails	✓	✓	✓
The FAIL function causes a component rule to fail	✓	✓	✓
The type of an input is invalid	✓	If stop on first error is set, input validation fails, causing the burst to fail. If stop on first error is not set, input validation continues until all inputs are validated. Input validation as a whole causes a burst failure if any input type is invalid.	If the error limit is reached and a burst fails because one or more inputs are invalid, the map fails.
The FAIL function causes a transformation rule to fail	✓	✓	✓
If an output audit fails If output audit is used, it is recommended that Scope is set to burst or map.		✓	✓
If fail is set for an input or output Warnings.	✓	✓	✓
Any target connection fails.	✓	✓	✓

If an adapter is referenced as a function argument and no FAIL function is used, all operations rollback when a burst fails. When a burst is successful, any successful operations are committed.

Configuring the RUN function and bursts

When using the RUN function, a source or target used for the map that is run can be configured to be part of the transaction of the burst using the RUN function. To do this, set the **Scope** of the card in the map referenced in the RUN function to **Map**. In this case, if the map that contains the RUN rule fails, any adapter used in the map referenced in the RUN function whose scope is **Map** will exercise the **OnFailure** action.

If the **Scope** of a source or target is set to burst or card, that resource will act based on the appropriate completion status of the card or burst configured in the referenced RUN function.

When using nested RUN functions, if a source or target contained in a nested RUN has a **Scope** set to **Map**, any higher failure will cause that source or target to exercise its **OnFailure** action.

The rollback transaction behavior on maps called with the RUN function is achieved by using an output card, not the PUT function.

Logical or adapter-specified FetchUnits

All adapters have a **FetchUnit**. Some adapters have the capability to aggregate transactional data units. A transactional data unit refers to some unit of data that can be acted upon without affecting the state of another such data unit - for example, removing a message from a queue. Some adapters have the capability to divide one transactional unit into smaller discrete units of data. Some adapters cannot do either of the above. How the **FetchUnit** is interpreted depends on what the adapter's capabilities are.

When the data retrieved from a source adapter represents one transactional unit that cannot be subdivided into smaller discrete units of data without the adapter knowing the content of the data, the **FetchUnit** is interpreted as a logical unit. For example, if an input is a file, it is not a good idea to delete that file until a map completes its work

- a stream file is one transactional data unit that cannot be divided into physical units by the adapter. In this case, the **FetchUnit** is interpreted as logical by looking at the type definition of the input data to decide when a burst is complete.

When an adapter can aggregate transactional data units as one source, the **FetchUnit** is interpreted in terms of the transactional data unit. For example, if the data source is MQSeries, multiple messages can be aggregated, and the **FetchUnit** is the maximum number of messages to retrieve for a given burst. In this case, it is up to you to make sure the input type definition matches the amount of data retrieved for a single burst.

For database adapters, an input query represents one transactional unit that can be divided into rows by the adapter. In this case, the **FetchUnit** is the maximum number of rows to retrieve from the database for a given burst.

When the source adapter can only retrieve one transactional unit of data, the source **Scope** is limited to **Map** - this prevents you from trying to delete half a file, or half a buffer, for example. If the source adapter can retrieve multiple transactional units of data, the source **Scope** can be **Map**, **Burst** or **Card**.

Building maps

After you have entered map rules in an executable map, created necessary functional maps, and entered map rules for the desired output data objects, the next step is to build the executable map. A compiled map file is created when no build errors occur when the executable map, and all referenced maps, are analyzed.

An executable map has a data source or target specified for each of its cards. Executable maps are built. Functional maps are not built.

Data sources and targets specified for an executable map are defaults built into the compiled map file. Data sources and targets, and other map settings, can be overridden from the Integration Flow Designer or when a map is run using the Command Server.

- [Referenced maps](#)
- [Multi-platform maps](#)

You can compile a map to run on multiple platforms, for example, Microsoft Windows, AIX®, and Linux®. At run time, Transformation Extender runs the appropriate version of the map for the platform.

- [Including a static input file in a compiled map](#)

You can include a static input file in a compiled map to reduce processing overhead in maps that run multiple times.

- [Building a map](#)

Building a map compiles it into an executable map.

- [Rebuilding a map that has changed](#)

- [Building maps for a single platform](#)

To build a map that runs on a single platform, use Map>Build for Specific Platform and choose the platform you want the map to support. To build all maps in the map source file for a single platform, use Map>Build All for Specific Platform and choose the platform.

- [Sample JCL](#)

When you build a map for the IBM z/OS platform, a job control language (JCL) template is generated in the map directory.

- [Schema errors](#)

- [Map build analysis errors](#)

- [Map build warning messages](#)

- [Map build compile errors](#)

- [Compiled map files](#)

Referenced maps

When a map is built, the map and all of the functional maps that are referenced within that map are analyzed. For example, when the map called **MyMap**, which references **MakeRecord**, which references **ItemMap**, is built, all three maps are analyzed. If there are no build errors, a compiled map file is created. The referenced maps are compiled within this file. Referenced maps are not created as separate compiled maps.

Multi-platform maps

You can compile a map to run on multiple platforms, for example, Microsoft Windows, AIX®, and Linux®. At run time, Transformation Extender runs the appropriate version of the map for the platform.

In Design Studio, click Window > Preferences > Transformation Extender > Map > Build Options to select the platforms that you want your maps to support. Apply saves your choices. OK saves your choices and closes the window.

When you build maps with Map>Build or Map>Build All, the compiled maps support your selected platforms.

Including a static input file in a compiled map

You can include a static input file in a compiled map to reduce processing overhead in maps that run multiple times.

To include a file in a map, specify the static file adapter and the file name on the connection before you compile the map. The file must exist at the time when the map compiles, and must be less than 10 MB in size.

Note: No new-line conversion or character-set conversion takes place in the file when the map runs on any platform.

Building a map

Building a map compiles it into an executable map.

The build analyzes the logical interfaces within map rules, and if the analysis is successful, compiles the map. Analysis includes checking for missing rules, invalid rules, invalid card definitions, verifying map references, and verifying the arguments of functions. Analysis also looks for circular map references - maps that reference one another.

Remember: A map rule is required for each rule cell on an output card. When no data transformation is needed for an output object, a map rule is not required. In this case, enter = NONE as the map rule.

Rebuilding a map that has changed

After you have changed a map or updated it because you changed a schema used in it, build and save the map before running it again.

Building maps for a single platform

To build a map that runs on a single platform, use Map > Build for Specific Platform and choose the platform you want the map to support. To build all maps in the map source file for a single platform, use Map > Build All for Specific Platform and choose the platform.

A map that is compiled for a single platform has a platform-specific file extension. The compiled map name is the executable map name with the platform-specific extension, such as .aix. For example, building the map "MyMap" for the Linux® platform creates the file MyMap.lnx in the map directory. These default file extensions prevent you from inadvertently overwriting your original compiled map. They also help identify the compiled map file to transfer to the specific platform environment.

A map that is compiled for multiple platforms has the same .mmc file extension as maps that are compiled for a Windows platform.

When deploying systems that contain maps with XML types to non-windows platforms, the XML Schema, and XML input files must be transferred in binary format.

- [Platform-specific file name extensions](#)

Platform-specific file name extensions

Maps built for execution on Windows platforms use the default .mmc compiled map file name extension.

The **Build for Specific Platform** command creates a compiled map file in the format required for a specified platform, which accounts for byte-order and character set differences on that platform.

By default, the Map Designer assigns a platform-specific file name extension to maps built for specific platforms. You can change the default file name extension under Window > Preferences > Transformation Extender Compiled Map Extensions.

The default file name extensions for compiled maps are listed in the following table.

Platform	Extension
HP-UX (Itanium processor)	.hpi
IBM® AIX® (RS/6000® processor)	.aix
IBM z/OS®	.mvs
Linux® (Intel processor)	.lnx
Microsoft Windows	.mmc
pLinux	.pxn
Sun Solaris (SPARC processor)	.sun
zLinux	.znx

Create a folder exclusively for platform-specific maps.

To build a map for a specific platform:

1. Select the map in the Outline view or the Composition view.
2. Select the Build for Specific Platform option in one of the following ways:
 - From the menu bar, click Map > Build for Specific Platform.
 - Right-click the map and click Build for Specific Platform.
3. Select the specific platform on which you plan to run the map.

After the map is built for the specific platform, perform a binary file transfer of the compiled map file to the remote server environment. For example, to transfer the ported map to your UNIX server, you can use the File Transfer Protocol (FTP). You can also deploy the compiled maps from the Map Designer to the remote servers. Double-click the .dtxdeployment file that is under your project in the Navigator view, or in the Misc folder under your project in the Extender Navigator view. In the Deployment Editor, click Deploy to build and transfer the maps to the remote servers that you specified under Window > Preferences > Transformation Extender Deployment.

- A platform-specific build is not required if all of the items referenced in map rules have specific character sets and a specific byte order. No items have a native definition.
- For example, you might have a map whose only purpose is to convert data from ASCII to EBCDIC. However, to avoid mistakes, always use the Build for Specific Platform option when your target platform has a different byte order or character set than the byte order or character set in your map development environment.
- A platform-specific build is not required if you select a platform whose byte order and character set are the same as the byte order or character set on your development machine.

Sample JCL

When you build a map for the IBM z/OS platform, a job control language (JCL) template is generated in the map directory.

The JCL template is customized with information about the executable map (*mapname.jcl*), including the map name, input file name, and output file name. The following example JCL template was created by the Map Designer based on the following files:

```
//DTXSMJCL JOB
//***** Sample JCL to execute the SINKMAP map using the Transformation Extender Command Server for z/OS (OS/390).
//*
//***** MODIFY THE FOLLOWING GLOBAL VARIABLES FOR YOUR INSTALLATION
//*
// SET DTXLIB=DTX.SDTXLOAD           <== DTX LOAD LIB
// SET MAPLIB=DTX.SDTXSAMP          <== DTX MAP DSN
// SET EMPLOYEE=DTX.SDTXMAPS(DTXSMTXT) <== EMPLOYEE FILE
// SET ALLEMPS=DTX.SDTXMAPS(DTXAETXT) <== ALLEMPS FILE
// SET CLASS=*                      <== SYSOUT CLASS
//*
//*
//** NOTE: If the LE runtime library is not in the MVS LINKLIST,
//**        the LE runtime library must be added to the STEPLIB
//**        concatenation.
//*
//***** This job will execute the sample SINKMAP map using the
//** Transformation Extender Command server program.
//** The map output is directed to sysout.
//*
//DTX EXEC PGM=DTXCMDSV,REGION=48M,
//PARM='SINKMAP -IF1 ALL_EMPL -IF2 EMPLOYEE -OF1 WEEK -OF2 TTLHOURS
//OF3 EMP_HOUR' //STEPLIB DD DISP=SHR,DSN=&DTXLIB;
//** Transformation Extender SINKMAP example map
//SINKMAP DD {specify map location here }
//** Transformation Extender Log/Audit/Trace/Debug outputs
//DTXLOG DD SYSOUT=&CLASS;
//DTXAUD DD SYSOUT=&CLASS;
//DTXTRACE DD SYSOUT=&CLASS;
//DTXDEBG DD SYSOUT=&CLASS;
//*
//SYSPRINT DD SYSOUT=&CLASS;
//SYSUDUMP DD SYSOUT=&CLASS;
//CEEDUMP DD SYSOUT=&CLASS;
//*
//** Map inputs
//ALL_EMPL DD {specify location here } DISP=SHR,DSN=&ALL_EMPL;
//EMPLOYEE DD {specify location here } DISP=SHR,DSN=&EMPLOYEE;
//** Map outputs
//WEEK DD {specify location here } SYSOUT=&CLASS;
//TTLHOURS DD {specify location here } SYSOUT=&CLASS;
//EMP_HOUR DD {specify location here } SYSOUT=&CLASS;
//** Transformation Extender work files
//SYSTMP00 DD DSN=&&TEMP00:,DISP=(,DELETE,DELETE),
//           DCB=(RECFM=FBS,LRECL=23476,BLKSIZE=23476),
//           UNIT=SYSDA,SPACE=(TRK,(20,1),RLSE)
//*
//SYSTMP01 DD DSN=&&TEMP01:,DISP=(,DELETE,DELETE),
//           DCB=(RECFM=FBS,LRECL=23476,BLKSIZE=23476),
//           UNIT=SYSDA,SPACE=(TRK,(20,1),RLSE)
//*
//SYSTMP02 DD DSN=&&TEMP02:,DISP=(,DELETE,DELETE),
//           DCB=(RECFM=FBS,LRECL=23476,BLKSIZE=23476),
//           UNIT=SYSDA,SPACE=(TRK,(20,1),RLSE)
//*
//SYSTMP03 DD DSN=&&TEMP03:,DISP=(,DELETE,DELETE),
//           DCB=(RECFM=FBS,LRECL=23476,BLKSIZE=23476),
//           UNIT=SYSDA,SPACE=(TRK,(20,1),RLSE)
//*
//SYSTMP04 DD DSN=&&TEMP04:,DISP=(,DELETE,DELETE),
//           DCB=(RECFM=FBS,LRECL=23476,BLKSIZE=23476),
//           UNIT=SYSDA,SPACE=(TRK,(20,1),RLSE)
//*
//SYSTMP05 DD DSN=&&TEMP05:,DISP=(,DELETE,DELETE),
//           DCB=(RECFM=FBS,LRECL=23476,BLKSIZE=23476),
//           UNIT=SYSDA,SPACE=(TRK,(20,1),RLSE)
```

```

/*
//SYSTMP06 DD DSN=&&TEMP06;,DISP=(,DELETE,DELETE),
//          DCB=(RECFM=FBS,LRECL=23476,BLKSIZE=23476),
//          UNIT=SYSDA,SPACE=(TRK,(20,1),RLSE)
*/
/*
//SYSTMP07 DD DSN=&&TEMP07;,DISP=(,DELETE,DELETE),
//          DCB=(RECFM=FBS,LRECL=23476,BLKSIZE=23476),
//          UNIT=SYSDA,SPACE=(TRK,(20,1),RLSE)
*/
/*
//SYSTMP08 DD DSN=&&TEMP08;,DISP=(,DELETE,DELETE),
//          DCB=(RECFM=FBS,LRECL=23476,BLKSIZE=23476),
//          UNIT=SYSDA,SPACE=(TRK,(20,1),RLSE)
*/
/*
//SYSTMP09 DD DSN=&&TEMP09;,DISP=(,DELETE,DELETE),
//          DCB=(RECFM=FBS,LRECL=23476,BLKSIZE=23476),
//          UNIT=SYSDA,SPACE=(TRK,(20,1),RLSE)
*/

```

Schema errors

Each card in the map specifies a type and a schema. Schemas that are referenced by the map must be analyzed and saved. If you build a map, and a schema referenced by that map has not been analyzed, or it has been analyzed but produced at least one error, an error message displays.

You are not prevented from building the map. However, you may get unpredictable results when running the map. For example, if a group's components are not distinguishable, it is not clearly defined where one component ends and the next one begins. The data may not validate correctly.

A schema that has not been analyzed as error-free includes the following:

- A schema that was analyzed and produced at least one error.
- A schema that has been analyzed, but not saved.

Map build analysis errors

If build errors occur, view the errors, correct the problem(s), and build the map again. A map cannot be run if it has not been built.

Build results of a map provide information on errors and warnings that occur during the map build process.

During the map build process, the map and all of the functional maps that are referenced within that map are analyzed.

- Build errors prevent a map from being compiled
- Build warnings alert you of a condition that might be problematic.

You can ignore warnings to allow the map to be built even if there are warnings.

If you build a map and there are warnings, but no errors, the map is compiled.

Map build warning messages

A build warning alerts you about a potentially problematic condition. If you build a map with warnings, but no errors, the map is successfully compiled. You can ignore warnings.

Map build compile errors

Maps with compile errors may not be built. Maps that are not compiled may not be run.

For example, a compile error results when a map is built and the input contains a delimited group whose delimiter has not been specified. To correct this compile error, specify the delimiter in the schema, analyze it, and save the schema. Then build the map again.

Compiled map files

If no errors occur, invoking a **Build** option creates a compiled map file. The compiled map file is written to the location of the map source file (.mms). The name of the compiled map name is the source map name with a platform-appropriate file extension, or .mmc for a multi-platform map.

Note: If you have more than one executable map with the same name, make sure that you save the different map source files in different folders. A compiled map file is an input for the Command Server or Launcher. Running a map from a command line uses the compiled map file.

Running a map

When you run a map, the information in the compiled map file is used to create the output data.

To run a map, use the Map Designer, Launcher, Command Server, an integration or transformation server, or an API.

If you have a Launcher, use the Integration Flow Designer to configure maps to run.

You can run a map from within an application by using the TX programming interface. You can use the **RUN** function to run a map within another map. You can also run maps from any supported integration or transformation products with which IBM Transformation Extender is integrated.

- [Running in a multi-threaded environment](#)
You can run maps in a multi-threaded environment.
- [Run results](#)

Running in a multi-threaded environment

You can run maps in a multi-threaded environment.

- [Thread-safe map](#)
For a map to run optimally in a multi-threaded environment, when multiple occurrences of the map are instantiated, none of the resources that an individual instance of the map uses can be used by another instance of the map.
- [Verifying that the map meets basic thread-safe checks](#)
Before you run your map, you can verify that the map meets at least the basic thread-safe checks.

Thread-safe map

For a map to run optimally in a multi-threaded environment, when multiple occurrences of the map are instantiated, none of the resources that an individual instance of the map uses can be used by another instance of the map.

By default, map resources are managed. This helps facilitate the concurrent running of maps. However, you should specify unique names for the map's work files, audit logs, backup files, and trace logs, within each instance of the map.

Verifying that the map meets basic thread-safe checks

Before you run your map, you can verify that the map meets at least the basic thread-safe checks.

You can do some of the same checks of the map and card settings that these functions in the Design Studio do to find potential threading issues. Other map usages, such as when the map is working in a flow context, or if the map rules have the **RUN**, **GETANDSET**, or **PUT** functions, or if the **PUT** function has other overrides, might also cause threading issues.

Verify that the settings for the properties have specific values for map and card settings.

- [Map settings to optimize concurrent performance](#)
There are values for map settings properties that you can verify to optimize concurrent performance.
- [Card settings to optimize concurrent performance](#)
There are values for card setting properties that you can verify to optimize concurrent performance.

Map settings to optimize concurrent performance

There are values for map settings properties that you can verify to optimize concurrent performance.

Verify that the settings for the WorkSpace and AuditLocation properties for the map settings have these specific values.

Table 1. WorkSpace

Property	Value
WorkSpace	File or Memory
Directory	Map or Custom
WorkFilePrefix	Unique

Table 2. AuditLocation (If Switch is set to ON)

Property	Value
AuditLocation	File or Memory
Directory	Map or Custom
FileName	Unique

Card settings to optimize concurrent performance

There are values for card setting properties that you can verify to optimize concurrent performance.

Verify that the settings for the Backup properties for the card settings have these specific values.

Table 1. Backup (If Switch is set to ON)

Property	Value
BackUpLocation	File
Directory	Map or Custom
FileName	Unique

Run results

Run results are the input data used in the map and the output data generated by the map. The run results are the actual data as described by the data objects defined in the input and output cards.

Do not confuse run results with build results, which provide information about errors and warnings that occur during the map build process.

Map Profiler

The Map Profiler is a user-configurable utility that captures and reports map execution statistics.

The profiler focuses on component and mapping rules and the functions and types within those rules. The resulting information enables you to see where performance is lagging and in turn make improvements in your maps.

Map Profiler output

The profiler can capture the following statistics:

- Amount of time spent processing component and mapping rules and subordinate functions.
- Number of times rules and functions are executed (from validation through output).
- Number of times type objects are accessed.
- Depth (nesting level) of the object being profiled.
- [Configuring the Map Profiler in the Design Server UI](#)
Configure the Map Profiler in the Design Server UI:
- [Map Profile Settings](#)
To run a map using the Map Profiler utility to capture and report map execution statistics, you must configure the Map Profile settings in the Map Profile window as follows:

Configuring the Map Profiler in the Design Server UI

Configure the Map Profiler in the Design Server UI:

1. Open the project in Design Server UI.
2. Go to Maps and open a map that you want to build using Map Profiler utility.
3. Click the Build button on the top right corner to build the map.
4. Click the Profile Map button on the top right corner.
The Map Profile window displays.
5. In the Map Profile window, enable a Function Times option or a Type Times option to see statistics in the output (Otherwise the report will be empty).
You can also configure the settings to view more statistics, see "Map Profile Settings", for more details.
6. Click OK to run the map using the Map Profiler utility to capture and report map execution statistics.
The Profiler window opens with a full report with Download, Open New Tab and Search options. A full report contains both function and type information.
7. Click the Download button to download the map profile report.
Note: If the map profile report is big and exceed the expected limit, you cannot see the report in the Profile window. To see the map profile report, you must download it using Download button.
8. Click the Open New Tab button to see the map profile report in new tab.
9. Search the required content using Search option.

Map Profile Settings

To run a map using the Map Profiler utility to capture and report map execution statistics, you must configure the Map Profile settings in the Map Profile window as follows:

Function times

Enable the All or Above option to view function time statistics in the profiler report.

Table 1. Function times

Option	Description
--------	-------------

Option	Description
None	(Default setting) When None is enabled, function time statistics are not reported.
All	When you select All, each function and each function call is provided in the profiler output. Every occurrence of every function is displayed with the depth (level), number of iterations, elapsed time, map name, and type name.
Above	Use the Above option to obtain a smaller output file. When you specify a value in 1/10 milliseconds (10000 = 1 second), function times are only recorded when they exceed that value. You can reduce profiler output by specifying a time limit so that nothing is reported until it exceeds the time limit that you specify.
Summary	When you choose the Summary option, the report contains a summary for each function instead of listing every occurrence. A function is listed once with the total number of iterations and total elapsed time for all iterations.

Type times

Enable the All or Above option to view type time statistics in the profiler report.

Table 2. Type times

Option	Description
None	(Default setting) When None is enabled, function time statistics are not reported.
All	When you select All, each type per rule is provided in the profiler output. For each rule, the time, map name, and type name are reported.
Above	Use the Above option to obtain a smaller output file. When you specify a value in 1/10 milliseconds (10000 = 1 second), types times are only recorded when they exceed that value. You can reduce profiler output by specifying a time limit so that nothing is reported until it exceeds the time limit that you specify.
Summary	When you choose the Summary option, the report contains a summary of types per rule. A type name is displayed once with the total number of iterations and total elapsed time for those iterations.

File output

Table 3. File output

Option	Description
Fixed Width	This option produces a readable, flat file output.
Comma Delimited	This option produces a comma-delimited output which is useful for exporting to spreadsheet programs.

Map performance

You can use the Map Profiler to analyze map execution behavior.

- [Map Profiler overview](#)
- [Map Profiler output](#)
- [Configuring the Map Profiler](#)
- [Profiler preferences](#)
Use the Profiler preferences to configure the default Map Profiler utility settings.
- [Command line](#)
- [Using the Map Profiler](#)
- [Profiler output example](#)
- [Profiler summary report example](#)
- [For best results...](#)

Map Profiler overview

The Map Profiler is a user-configurable utility that captures and reports map execution statistics. The profiler focuses on component and mapping rules and the functions and types within those rules. The resulting information enables you to see where performance is lagging and in turn make improvements in your maps.

For example, in the profile report you see that the processing time for a **LOOKUP** function is significantly more than other functions. You can modify the rule to use a better choice, such as the **SEARCHUP** function (because in this specific case the data in the **LOOKUP** is ascending), and as a result the processing time is greatly reduced.

Map Profiler output

The profiler can capture the following statistics:

- Amount of time spent processing component and mapping rules and subordinate functions.
- Number of times rules and functions are executed (from validation through output).
- Number of times type objects are accessed.
- Depth (nesting level) of the object being profiled.
- [Type names](#)
- [Function times](#)
- [RUN maps](#)

Type names

Similar to the map trace file, the type names referred to in the profile output represent the full type name in the original schema, not that of the type object that is visible in the Map Designer.

Function times

The processing time of any subfunctions rolls up into the processing time of the top level function. For example, if an EITHER function contains an IF function, the recorded time for the EITHER function would include the processing time of the IF function.

RUN maps

When a parent map contains a RUN map, the RUN map is not profiled. To profile a RUN map, it must be done outside of the RUN rule. (You can do this by running it as a top level executable map.)

Configuring the Map Profiler

The profiler can be configured and invoked from the Map Designer interface or from the command line.

From the Map Designer interface, you can enable or disable the profiler from the Map menu or the toolbar. Before running the map, you must open the Map Profiler options and enable either the **Function Times** option or **Type Times** option to see statistics in the output (otherwise the report is empty).

Profiler preferences

Use the Profiler preferences to configure the default Map Profiler utility settings.

Before profiling a map, you must enable a **Function Times** option or a **Type Times** option to see statistics in the output, otherwise the report is empty.

To access user preferences, click Window > Preferences > Transformation Extender > Map > Profiler.

To run a map using the Map Profiler utility to capture and report map execution statistics, you must configure the profiler settings in the Map Designer for the following profiler options:

Function times

Enable the **All** or **Above** option to view function time statistics in the profiler report.

Option	Description
None	(Default setting) When None is enabled, function time statistics are not reported.
All	When you select All, each function and each function call is provided in the profiler output. Every occurrence of every function is displayed with the depth (level), number of iterations, elapsed time, map name, and type name.
Above	Use the Above option to obtain a smaller output file. When you specify a value in 1/10 milliseconds (10000 = 1 second), function times are only recorded when they exceed that value.
Summary	When you choose the Summary option, the report contains a summary for each function instead of listing every occurrence. A function is listed once with the total number of iterations and total elapsed time for all iterations. Sample report .

Type times

Enable the **All** or **Above** option to view type time statistics in the profiler report.

Option	Description
None	(Default setting) When None is enabled, type time statistics are not reported.
All	When you select All, each type per rule is provided in the profiler output. For each rule, the time, map name, and type name are reported.
Above	Use the Above option to obtain a smaller output file. When you specify a value in 1/10 milliseconds (10000 = 1 second), types times are only recorded when they exceed that value.
Summary	When you choose the Summary option, the report contains a summary of types per rule. A type name is displayed once with the total number of iterations and total elapsed time for those iterations. Sample report

File output

Option	Description
Fixed Width	This option produces a readable, flat file output.
Comma Delimited	This option produces a comma-delimited output which is useful for exporting to spreadsheet programs.

Command line

The dstxprof console application enables you to profile maps from the command line.

See the Utility Commands documentation for information about using the dstxprof utility.

Using the Map Profiler

To profile a map:

1. Select a map and click Map>Enable Profiler.
2. Build and run the map.
3. Open the Organizer and select the Profile tab to view results.

You can also open Map Profiler results in a text editor from the map directory. The filename is a combination of the map name and a "PROF" extension. For example, *map_name.MPROF*.

Profiler output example

A full report contains both function and type information. The Profiler produces the following output when you run a full report against the sinkmap.mms file that is installed in *install_dir\examples\general\map\sinkmap* directory:

```
Profiling for install_dir 9.0.0\examples\general\map\SINKMAP\sinkmap.mmc
Total entries: 66
Total map sets: 1

Individual function breakdown
Function      Depth Iterations     Time      Map          Type
-----
Validation
VALIDATE_CARD    0      1        2720    sinkmap
VALIDATE_CARD    0      1         34    sinkmap

Mapping
BUILD_CARD       0      1        2064    sinkmap
LOOKUP           1      1        1501    sinkmap          week example
EQ_OP            2      1        185    sinkmap          week example
TYPE_NAME        3      2        13     sinkmap          week example
EQ_OP            2      1         3    sinkmap          week example
TYPE_NAME        3      2         1    sinkmap          week example
EQ_OP            2      1         2    sinkmap          week example
TYPE_NAME        3      2         2    sinkmap          week example
EQ_OP            2      1         2    sinkmap          week example
TYPE_NAME        3      2         1    sinkmap          week example
BUILD_CARD       0      1        1536    sinkmap
NUMBERTOTEXT     1      1        1524    sinkmap          total_hours element example
ADD_OP           2      1        1521    sinkmap          total_hours element example
ADD_OP           3      1        1503    sinkmap          total_hours element example
ADD_OP           4      1        1489    sinkmap          total_hours element example
ADD_OP           5      1        1473    sinkmap          total_hours element example
DIV_OP           6      1        1461    sinkmap          total_hours element example
SUE_OP           7      1        1444    sinkmap          total_hours element example
TEXTTONUMBER     8      1        1368    sinkmap          total_hours element example
TIMETOTEXT       9      1        1322    sinkmap          total_hours element example
TYPE_NAME        10     1        1318    sinkmap          total_hours element example
TEXTTONUMBER     8      1         15    sinkmap          total_hours element example
TIMETOTEXT       9      1         7     sinkmap          total_hours element example
TYPE_NAME        10     1         2     sinkmap          total_hours element example
FLOAT_CONSTANT   7      1         13    sinkmap          total_hours element example
DIV_OP           6      1         11    sinkmap          total_hours element example
SUB_OP           7      1         9     sinkmap          total_hours element example
TEXTTONUMBER     8      1         5     sinkmap          total_hours element example
TIMETOTEXT       9      1         4     sinkmap          total_hours element example
TYPE_NAME        10     1         2     sinkmap          total_hours element example
TEXTTONUMBER     8      1         3     sinkmap          total_hours element example
TIMETOTEXT       9      1         2     sinkmap          total_hours element example
TYPE_NAME        10     1         1     sinkmap          total_hours element example
FLOAT_CONSTANT   7      1         0     sinkmap          total_hours element example
DIV_OP           5      1         14    sinkmap          total_hours element example
SUB_OP           6      1         9     sinkmap          total_hours element example
TEXTTONUMBER     7      1         4     sinkmap          total_hours element example
TIMETOTEXT       8      1         3     sinkmap          total_hours element example
TYPE_NAME        9      1         1     sinkmap          total_hours element example
TEXTTONUMBER     7      1         3     sinkmap          total_hours element example
TIMETOTEXT       8      1         2     sinkmap          total_hours element example
TYPE_NAME        9      1         1     sinkmap          total_hours element example
FLOAT_CONSTANT   6      1         1     sinkmap          total_hours element example
DIV_OP           4      1         12    sinkmap          total_hours element example
SUB_OP           5      1         10    sinkmap          total_hours element example
TEXTTONUMBER     6      1         5     sinkmap          total_hours element example
TIMETOTEXT       7      1         3     sinkmap          total_hours element example
```

TYPE_NAME	8	1	2	sinkmap	total_hours element example
TEXTTONUMBER	6	1	4	sinkmap	total_hours element example
TIMETOTEXT	7	1	2	sinkmap	total_hours element example
TYPE_NAME	8	1	1	sinkmap	total_hours element example
FLOAT_CONSTANT	5	1	1	sinkmap	total_hours element example
DIV_OP	3	1	16	sinkmap	total_hours element example
SUB_OP	4	1	10	sinkmap	total_hours element example
TEXTTONUMBER	5	1	6	sinkmap	total_hours element example
TIMETOTEXT	6	1	6	sinkmap	total_hours element example
TYPE_NAME	7	1	1	sinkmap	total_hours element example
TEXTTONUMBER	5	1	3	sinkmap	total_hours element example
TIMETOTEXT	6	1	2	sinkmap	total_hours element example
TYPE_NAME	7	1	1	sinkmap	total_hours element example
FLOAT_CONSTANT	4	1	4	sinkmap	total_hours element example
BUILD_CARD	0	1	32	sinkmap	
TYPE_NAME	1	1	1	sinkmap	employee example
TYPE_NAME	1	1	24	sinkmap	total_hours element example

Type per rule breakdown

Time	Map	Type
------	-----	------

Validation

Mapping

4818	sinkmap	
1710	sinkmap	week example
1536	sinkmap	
14624	sinkmap	total_hours element example
32	sinkmap	
1	sinkmap	employee example
24	sinkmap	total_hours element example

Type per rule summary

Iterations	Time	Map	Type
------------	------	-----	------

Validation

Mapping

1	1710	sinkmap	week example
2	14648	sinkmap	total_hours element example
1	1	sinkmap	employee example

Function summary

Function	Iterations	Time
----------	------------	------

VALIDATE_CARD	2	2754
BUILD_CARD	3	3632
TYPE_NAME	20	1372
FLOAT_CONSTANT	5	19
EQ_OP	4	192
ADD_OP	4	5986
SUB_OP	5	1482
DIV_OP	5	1514
LOOKUP	1	1501
NUMBERTOTEXT	1	1524
TIMETOTEXT	10	1353
TEXTTONUMBER	10	1416

Profiler summary report example

A summary report containing both function and type information looks similar to the following example:

Individual Function Breakdown					
function	depth	iterations	time	map	type
<hr/>					
validation					
VALIDATE_CARD	0	1	208	Executable	
VALIDATE_CARD	0	1	192	Executable	
TYPE_NAME	1	16	4	Executable	Detail Record Input Data
<hr/>					
mapping					
BUILD_CARD	0	1	193	Executable	
MAP_NAME	1	1	59	Executable	PO Data
TYPE_NAME	2	1	1	Executable	PO Data
CHOOSE	2	1	4	Executable	PO Data
<hr/>					
Type per Rule Breakdown					
	time	map		type	
	---	--		----	
validation	400	Executable			
<hr/>					
mapping	4	Executable		Detail Record Input Data	
<hr/>					
mapping	193	Executable		PO Data	
	65	Executable		Company Field Data	
	16	MakePO			

1	MakePO	PO# Field Data
0	MakePO	PODate Field Data
18	MakePO	Detail Record Data
Type per Rule Summary		
iterations time map type		
validation		
1	4	Executable Detail Record Input Data
mapping		
3	140	Executable PO Data
3	16	MakePO Company Field Data
3	3	MakePO PO# Field Data
3	0	MakePO PODate Field Data
Function Summary		
function iterations time		
VALIDATE_CARD 2 400		
BUILD_CARD 1 193		
INDEX 3 1		
TYPE_NAME 73 49		
STRING_LITERAL 15 3		

For best results...

- If you have a large amount of data, the profiler output could be very large or unmanageable. For this reason, it is best to profile only a subset of the original data.
- You can reduce profiler output by specifying a time limit so that nothing is reported until it exceeds the time limit that you specify. See section "[Configuring the Map Profiler](#)" for details.
- When running the profiler, always limit or terminate other processes running on your machine to avoid adverse results.

Managing invalid data

You can map invalid data to an output, just like any other data. For example, you can place all invalid data in a separate output for editing or debugging purposes. You can map invalid data to an error report.

To map invalid data, it must be contained in a valid object. A map does not run unless all input card objects are valid. For example, if you want to map an invalid purchase order in a file of many purchase orders, the file itself must be valid.

- [Using the Restart Attribute](#)
- [Using the REJECT function](#)
- [Using the ISERROR function](#)
- [Using the CONTAINSERRORS function](#)

Using the Restart Attribute

To map invalid data of a particular object, assign the **Restart** attribute at the component level. Errors must be ignored for invalid data objects during validation. You can then map the invalid data using any or all of the error functions **REJECT**, **ISERROR**, and **CONTAINSERRORS**.

IBM Transformation Extender information, including the **Restart** information, is stored in temporary, read-only native schema (.mtx) files. These files have the same name as the native schema, but with the .mtx extension. Any changes to schemas referenced in these .mtx files might cause the process that maps items from the schema, to update the .mtx file, and remove the **Restart** information that was stored there. If this occurs, or if the file gets deleted, you must recreate the **Restart** attribute by right-clicking the type in the card, and clicking Restart.

For additional information about the **Restart** attribute, see the Type Designer documentation.

Using the REJECT function

To map invalid data, use the **REJECT** function. The **REJECT** function is used in conjunction with the restart attribute. The **REJECT** function operates on an invalid data object contained within a valid object. The output of the **REJECT** function is a series of text items.

The syntax of the **REJECT** function is the following:

```
REJECT (component_with_restart_attribute)
```

Use the **REJECT** function with expressions that map valid data. For example, you can generate an error report that contains the invalid input data, in addition to the valid company name and date from the input.

Using the ISERROR function

You may conditionally map data based on whether or not a certain object is invalid using the **ISERROR** function. The **ISERROR** function has one input argument and returns a boolean value. It returns "true" if the argument is invalid or returns "false" if the argument is valid.

The **ISERROR** function is used with the **Restart** attribute. The **ISERROR** function operates on an invalid data object contained within a valid object.

In this input file of four customers, the third customer record is invalid:

```
Gisela,1252 S. Broward/Ft. Lauderdale,523-2622,22  
Dede,1513 Palatine Rd./Boca Raton,252-6560,86  
Lewis,74099 S. 67th Ave/Ft. Lauderdale,332-8665,3,3  
Eric,6933 Main St./South Miami Beach,291-7281,56
```

To generate a report that indicates whether each customer object is valid, assign a **Restart** attribute to **Customer(s)** in the input.

The output report is composed of **Line(s)**, where **Line** is defined as a text item. The map rule on **Line** uses the **ISERROR** function to check whether **Customer** is invalid. If **Customer** is invalid, the result is **INVALID Customer Data** and the **REJECT** of **Customer**. If **Customer** is valid, the result is **GOOD Customer Data** and the **TEXT** conversion of the **Customer** object itself.

Using the **CONTAINSERRORS** function

You may not want to map the input data if it contains any invalid data. In this example or situation, use the **CONTAINSERRORS** function. The **CONTAINSERRORS** function has one input argument that is the name of the object you want to test. The **CONTAINSERRORS** function returns "true" if the argument contains any invalid object or returns "false" if the argument does not contain any invalid object.

The **CONTAINSERRORS** function only operates on a valid object.

For example, if you have the same input file as in the example above, the third **Customer** record is invalid. You are mapping the input to the output file of **CustomerEntry(s)**. If the entire input contains any invalid object, do not produce any output. If the entire input is valid, map the **Customer(s)** to **CustomerEntry(s)**, using the functional map **MakeEntry**.

The map rule on **CustomerEntry(s)** uses the **CONTAINSERRORS** function to check if the entire **List** contains any errors. If **List** contains an error, no output is produced. If **List** does not contain errors, the **Customer(s)** will be mapped to **CustomerEntry(s)**.

The result is no output, because **List** does contain an error - the third **Customer** record is invalid.

Map audit overview

A map audit log is created when a map is executed. The log information is based on a particular map and provides information about map execution, map settings, card settings, and specific data objects. Map audit logs are in XML format and the contents can be mapped in such a way as to provide performance statistics and information about the content of your data.

- [Audit log contents](#)
- [Data settings in the audit log](#)
- [Map settings in the audit log](#)
- [Mapping from the audit log](#)

Audit log contents

A map can be configured to produce a map audit log consisting of the following:

- Execution statistics in summary or burst mode
- Burst information
- Map settings
- Data settings
- Data content information
- [Controlling audit log content](#)
The map settings that control the map audit. You can also specify audit options from the command line. See Execution Commands documentation for command options. The audit log can be stored in a file or returned in memory if you use the **RUN** function or an API.
- [Execution summary in the audit log](#)
- [Elapsed execution time in the audit log](#)
- [Source data report by card](#)
- [Target data report by card](#)
- [BurstAudit Data in the audit log](#)
- [Data content information in the audit log](#)

Controlling audit log content

The map settings that control the map audit. You can also specify audit options from the command line. See Execution Commands documentation for command options. The audit log can be stored in a file or returned in memory if you use the **RUN** function or an API.

Execution summary in the audit log

The MapAudit Switch = On and the SummaryAudit Execution = On map settings produce an audit log with execution summary information. The **ExecutionSummary** element of the audit log provides performance data for the map as a whole.

Execution summary information includes:

- Map status information
- Compiled map name
- Map return code
- Number of input objects found
- Elapsed execution time
- Number of output objects built
- Burst restart count
- Source data report by card
- Map return code message
- Target data report by card
- Work area information by card

The format of the elapsed time is presented differently on Windows platforms than on non-Windows platforms.

Some information is optional, and does not always appear in the **ExecutionSummary** element of the audit log. For example, information about objects found and built is provided only when using a command engine. **WorkArea** data is provided only when a work area is used during map processing.

Elapsed execution time in the audit log

The execution audit log records the elapsed time. The elapsed time format is presented differently on Windows platforms than on non-Windows platforms.

- On Windows platforms, the elapsed time of map execution is described to four decimal places.
- On non-Windows platforms, the elapsed time of map execution is presented in whole seconds only (rounded).

Source data report by card

In the audit log, the source data report by card provides information on:

- Value of the adapter **Source** card setting
- Source data byte count
- Adapter return code and message
- Adapter command line or file path
- Data source time stamp
- [Validation Errors in the Source Report Section of the Audit Log](#)

Validation Errors in the Source Report Section of the Audit Log

The Message section of the Source Report documents the validation errors generated by the engine for each card. This functionality enables users to conveniently review the validation errors after executing the map. It is essential to clarify that the recording of these error messages in the audit log does not require the enabling of engine trace.

To enable the audit option that reports the validation errors, in the Map audit settings set the SummaryAudit.Execution to Always, OnError or OnErrorOrWarning.

The number of error messages can be regulated through a setting in the config.yaml configuration file:

```
runtime:  
  validationErrors:  
    maximumCount: 20
```

The default value for this parameter is set at 20. If you wish to deactivate validation error reporting, simply set this configuration parameter to zero.

The following is a sample of the map audit when the validation fails for a choice group. SourceReport sections shows three validation error messages. The choice group had three types and all failed to validate.

```
<ExecutionSummary MapStatus="Error" mapreturn="8" ElapsedSec="0.0018" BurstRestartCount="0">  
  <Message>One or more inputs was invalid</Message>  
  <CommandLine>C:\maps\validationerrors\choicegroups\fix_tag.mmc</CommandLine>  
  
<SourceReport  card="1"    adapter="File"    bytes="6"    adapterreturn="0">  
  <Message>  
    Data read successfully  
    Data at offset 4 of size 1 failed RESTRICTION check for text [relative_size valid_values IOIQty Tag FIX]  
    Data at offset 4 of size 1 failed item presentation test for number [qty float DataTypes FIX]  
    Data at offset 4 of size 1 failed RESTRICTION check for text [a_b_or_c valid_values IOIQty Tag FIX]  
  </Message>  
<Settings>C:\maps\validationerrors\choicegroups\fix_tag_all_3_invalid.inp</Settings>  
<TimeStamp>2024-01-04T18:36:13Z</TimeStamp>
```

```
</SourceReport>
<WorkArea type="Memory">
<inputarea card="1" bytes="131344"/>
</WorkArea>
```

In addition to messages describing the validation error, the last known data that was correctly validated can be reported. In order to enable this message, user needs to set in the config.yaml file:

```
runtime:
  validationErrors:
    lastFound: true
```

The default value for lastFound is false. If the lastFound option is set to true, the example map audit file looks like:

```
<ExecutionSummary MapStatus="Error" mapreturn="8" ElapsedSec="0.0058" BurstRestartCount="0">
  <Message>One or more inputs was invalid</Message>
  <CommandLine>test.mmc</CommandLine>

<SourceReport card="1" adapter="File" bytes="91" adapterreturn="0">
  <Message>
    Data read successfully
    Data at offset 62 of size 0 is INVALID data for text [lastname textblobs Root] (A required object is missing.)
    Data at offset 89 of size 0 is INVALID data for text [lastname textblobs Root] (A required object is missing.)
    Data at offset 86 of size 2 was last found to be number [month textblobs Root]
  </Message>
  <Settings>inout.txt</Settings>
  <TimeStamp>2024-06-10T14:05:11Z</TimeStamp>
</SourceReport>
```

The last message points to the offset and size of data that was last found to be valid.

If the Audit data type validation is enabled on output cards, the engine validation errors audit report is not supported.

Target data report by card

In the audit log, the target data report by card provides information on:

- Value of the adapter **Target** card setting
- Target data byte count
- Adapter return code and message
- Adapter command line or file path
- Data target time stamp
- [Work area information by card](#)
- [Execution summary audit log sample](#)
- [BurstAudit execution in the audit log](#)

Work area information by card

In the audit log, the work area information includes:

- Work area type (file or memory)
- Work area file name and time stamp if applicable
- Work area byte count

Execution summary audit log sample

The following is a sample of the execution summary portion of the audit log.

```
<ExecutionSummary MapStatus="Valid" mapreturn="0" ElapsedSec="0.0373" BurstRestartCount="0">
  <Message>Map completed successfully</Message>
  <CommandLine>'install_dir\examples\CallsSummary.mmc'</CommandLine>
  <ObjectsFound>18</ObjectsFound>
  <ObjectsBuilt>12</ObjectsBuilt>
<SourceReport card="1" adapter="File" bytes="52" adapterreturn="0">
  <Message>Data read successfully</Message>
  <Settings>install_dir\examples\stores.txt</Settings>
  <TimeStamp>18:10:04 December 26, 2000</TimeStamp>
</SourceReport>
<SourceReport card="2" adapter="File" bytes="69" adapterreturn="0">
  <Message>Data read successfully</Message>
  <Settings>install_dir\examples\CALLS.TXT</Settings>
  <TimeStamp>18:10:04 December 26, 2000</TimeStamp>
</SourceReport>
<TargetReport card="1" adapter="File" bytes="119" adapterreturn="0">
  <Message>Data written successfully</Message>
  <Settings>install_dir\examples\summary.txt</Settings>
  <TimeStamp>10:14:34 April 27, 2001</TimeStamp>
</TargetReport>
```

```

<WorkArea type="File">
<inputarea card="1" Path="install_dir\examples\CallsSummary.I01"
TimeStamp="10:14:34 April 27,
2001" bytes="65695"/>
<inputarea card="2" Path="install_dir\examples\CallsSummary.I02"
TimeStamp="10:14:34 April 27,
2001" bytes="65695"/>
<outputarea card="1" Path="install_dir\examples\CallsSummary.O01"
TimeStamp="10:14:34 April
27, 2001" bytes="65695"/>
</WorkArea>
</ExecutionSummary>

```

BurstAudit execution in the audit log

The MapAudit Switch = On and the BurstAudit Execution = On map settings produce an audit log with burst execution information. The burst element provides information on the performance of the burst.

- If all input cards have the **FetchAs** mode of **Integral**, there is one burst element in the audit log.
- If any input card has the **FetchAs** mode of **Burst**, there is one element for each burst.

The execution log contains the return code of the burst, the amount of time it took to process the burst (in seconds), and a status of each input and output. The input and output status include the card number, the bytes processed, the adapter return code, and the content return code.

For example, if a burst fails because an adapter could not connect, the audit log defines which adapter failed. If an input is invalid, the content return code specifies which input is invalid. If an output rule fails, the content return code tells you which output failed.

A sample of the burst audit execution information in the audit log is:

```

<Burst count="1">
<ExecutionLog burstreturn="0" ElapsedSec="0.0118">
<inputstatus card="1" bytes="52" adapterreturn="0" contentreturn="0"/>
<inputstatus card="2" bytes="69" adapterreturn="0" contentreturn="0"/>
<outputstatus card="1" bytes="119" adapterreturn="0" contentreturn="0"/>
</ExecutionLog>
</Burst>

```

BurstAudit Data in the audit log

The MapAudit Switch = On and the BurstAudit Data = On map settings produce an audit log with burst audit data information. The burst audit data information includes a burst count, the card number, the object index, the audit log status code, the object level, the size of the object, and the object name.

A sample of the burst audit data information is:

```

<Burst count="1">
<DataLog>
<output card="1">
<object index="1" level="1" size="24" status="V00">Text1 Field</object>
</output>
</DataLog>
</Burst>

```

Data content information in the audit log

You can audit input and output data. The data audit settings determine what information about the data is written to the audit log.

Auditing data keeps track of specified objects in the input(s) and/or output(s) of a map. The burst audit data information in the audit log can track every occurrence of a record in a file and record the total count of records.

In the resulting audit log, the **<DataLog>** section follows the **</Burst>** tag. Within **<DataLog>**, there is an entry for the first input card. Within the input entry, there is an entry for each occurrence of the object.

If item data is specified in an audit setting, and the item is to be interpreted as binary, the line following the object line has the start tag **<Hex>** and the data is presented as pairs of hex digits for each byte.

In the audit log, the **<DataLog>** section follows the **</Burst>** tag. Within **<DataLog>**, there is an entry for the first input card. Within the input entry, there is an entry for each occurrence of the object.

If item data is specified in an audit setting, and the item is to be interpreted as binary, the line following the object line has the start tag **<Hex>** and the data is presented as pairs of hex digits for each byte.

Data settings in the audit log

The MapAudit Switch = On and the SettingsAudit Data = On map settings produce an audit log with input and output card setting information. Data settings are defined on the input and output cards of a map and include:

- Card number
 - Input card mode (as defined with the **FetchAs** card setting)
 - **WorkArea** reuse setting
 - Backup card settings
 - Source and target adapter settings
 - Transaction settings
 - Resource details
 - Retry settings
 - [Data settings in the audit log sample](#)
-

Data settings in the audit log sample

```
<DataSettings>
<InputData card="1" CardMode="Integral" WorkArea="!Reuse">
  <Backup switch="OFF"></Backup>
  <Source adapter="File" OnSuccess="Keep" OnFailure="Rollback" Scope="Map"
Warnings="Ignore">
    <Resource>install_dir\examples\stores.txt</Resource>
    <Retry switch="OFF"></Retry>
  </Source>
</InputData>
<InputData card="2" CardMode="Integral" WorkArea="!Reuse">
  <Backup switch="OFF"></Backup>
  <Source adapter="File" OnSuccess="Keep" OnFailure="Rollback" Scope="Map"
Warnings="Ignore">
    <Resource>install_dir\examples\CALLS.TXT</Resource>
    <Retry switch="OFF"></Retry>
  </Source>
</InputData>
<OutputData card="1">
  <Backup switch="OFF"></Backup>
  <Target adapter="File" OnSuccess="Keep" OnFailure="Rollback" Scope="Map"
Warnings="Ignore">
    <Resource>install_dir\examples\summary.txt</Resource>
    <Retry switch="OFF"></Retry>
  </Target>
</OutputData>
</DataSettings>
```

Map settings in the audit log

The MapAudit Switch = On and the SettingsAudit Map = On map settings produce an audit log with map setting information.

Map settings include:

- Map audit settings
- Map trace information
- Paging scheme
- Validation specifications
- Audit log location
- Work space information
- Century information

Map settings are viewable in the audit log:

```
<MapSettings>
<MapAudit switch="ON">
  <Log execution="OFF" data="OFF" mapsettings="ON" datasettings="OFF"/>
  <Location type="File" action="Create">
    <Directory type="Map">install_dir\examples</Directory>
    <FileName type="Default" prefix="MapName">CallsSummary.log</FileName>
  </Location>
</MapAudit>
<MapTrace switch="OFF"></MapTrace>
<WorkSpace>
  <Location type="Memory">
  </Location>
  <Paging>
    <PageSize>64</PageSize>
    <PageCount>8</PageCount>
  </Paging>
</WorkSpace>
<SlidingCentury switch="OFF"></SlidingCentury>
<CustomValidation switch="OFF"></CustomValidation>
<Retry switch="OFF"></Retry>
<BurstRestart switch="OFF"></BurstRestart>
<Warnings type="Every" action="Warn"></Warnings>
</MapSettings>
```

Mapping from the audit log

Mapping information from the audit log can provide useful performance statistics and information about the content of your data. For mapping purposes, the tryaudit.mtt schema provided in the **examples** directory defines the audit log data objects.

The format of the audit log is XML-based to enable use with other applications. The audit log is defined as a series of **MapAudit** elements. Each **MapAudit** element represents the log data for one map instance. You can get audit data for many map instances when the **AuditLocation** map setting for **FileName Action = Append**.

The **MapAudit** general element in the input card shown below displays the sequence of the information in the audit log.

- [ExecutionSummary in the Tryaudit schema](#)
- [Configuring data audit](#)
- [Status codes in the audit log](#)

ExecutionSummary in the Tryaudit schema

The audit log is created for a specific map. During map execution, after all bursts complete, the status of the map is summarized in the **ExecutionSummary**. The **ExecutionSummary** element in the audit log provides performance data for the map as a whole. This information appears in the audit log when the **SummaryAudit_Execution** map setting = **ON**.

Open *install_dir\examples\general\audit\tryaudit.mtt*, where *install_dir* refers to the path where IBM Transformation Extender is installed, to view the structure of **ExecutionSummary**.

The **ExecutionSummary** element contains:

- Map status
- Map return code
- Amount of map processing time
- **BurstRestartCount** indicator of how many times a burst restart occurred
- Return message for the map
- Command line that was used to re-configure any options compiled in the map
- Report for each source and target
- Work-area data

Summary information provides performance data for the map as a whole. For example, summary information might include the fact that all bursts succeeded, but the map failed because **Scope** was set to **Map**, and an adapter failed.

Summary source and target reports the adapter used for each, the total number of bytes processed over all bursts, and the final adapter return code.

Configuring data audit

Use the Data Audit Settings tab in the Organizer window to specify what information about the data to include in the audit log. Audit logs are for a single map.

To specify that you want to audit a particular object, drag that object from a card into the **Data Audit Settings** tab.

To audit a particular data object:

1. For the specific map with cards that contain the data object, define the **MapAudit** map settings in the Map Settings dialog box:
 - **MapAudit > Switch = On**
 - **BurstAudit > Data = On**

2. On the Map menu, click **Organizer**.
The **Organizer** window is displayed.

3. Drag the data object from the card into the **Audit** field on the Data Audit Settings tab.
4. From the **Track** list, select what information to track.
5. From the **Detail** list, select what information to report about the object's nested components.
6. If possible, from the **Item Data** list, specify when to show the item data of that object.

Data audit settings are compiled into the map file. If you change the audit settings in a map, you must build the map again for those settings to take effect.

The name of an object in the Data Audit Settings tab is the complete object name. Names are not entered with ellipses, even if the **Use Ellipses** option is enabled.

- [Detail](#)
- [Item data](#)

Detail

In the **Detail** list on the Data Audit Settings tab in the Organizer window, specify what information to include about an object's nested components.

Value

Description

Occurrence

A line appears in the audit log for each occurrence of that object of each nested component.

Warning

A line appears in the audit log for any nested component that causes a warning.

Error

A line appears in the audit log for any nested component that causes an error.

None

No information about nested components is included for that particular object.

Item data

In the **Item Data** list on the Data Audit Settings tab in the Organizer window, specify what actual *data* to show in the audit log. This applies to all items nested within the tracked object.

Value

Description

Occurrence

The data of each nested item within that object appears in the audit log.

Warning

The data of each nested item within that object that causes a warning appears in the audit log.

Error

A line appears in the audit log for each nested item within that object that causes an error.

None

No information about nested components is included for that particular object.

Status codes in the audit log

The status attribute in the Audit Log is a letter followed by a code number. The letter of the status attribute tells whether the data is valid V, or caused an error E, or caused a warning W.

Troubleshooting

There are various methods of troubleshooting that can assist you in situations where when running a map, the following types of errors occur regarding input or output data:

- Data is invalid
- Data contains invalid objects
- Unknown data exists

About data validation

When you run a map, the first thing that happens is the input data is validated by comparing the data to its definition in the schema. After validating data, the output card map rules are evaluated, and the output objects are built.

If you run a map and invalid or unknown input data is encountered, you can analyze what happened during the validation process. In addition, you can analyze the output data.

- [Overview of debugging aids](#)
- [Command Server window](#)
- [Run results](#)
- [Audit log](#)
- [Customized validation settings](#)
- [Map Debugger](#)
- [Trace files](#)

Overview of debugging aids

If you run a map and get a runtime error, or you do not get the expected output, use any or all of the following debugging tools:

- [Command Server window](#)
- [Run results](#)
- [Audit Log](#)
- [Customized validation settings](#)
- [Map Debugger](#)
- [Trace files](#)

Command Server window

After a map is run, a runtime message appears in the Command Server window. This message provides information about the map.

For example, the message "Map completed successfully" indicates that all inputs are valid. However, you might see the message "One or more inputs was invalid"- indicating that an input card object is invalid.

The message "One or more inputs was invalid" can occur even if you are absolutely sure that the input data is valid. In that case, there may be an error in the data definitions in the Type Designer. For example, if you mistakenly defined an item to be an integer but the data is actually a decimal number, that data object is considered invalid because it does not match its type definition. In general, when data is diagnosed as "invalid", either the data is invalid or the type definition is incorrect.

The Command Server window also displays the progress of the data being mapped. For example, you can see when input objects are found and when output objects are built.

Run results

Run results are the input data used in the map and the output data generated by the map. The run results are the actual data as described by the data objects defined in the input and output cards.

By viewing the results of a map execution, you can gain important information about how the data was mapped. For example, when no output objects are built, your input data might be invalid. Or, if you see only a portion of the output data, your input data might be valid but it could contain unknown data.

For more information about run results, see "[Run results](#)".

Audit log

The audit log stores the information about your data that you specify from within the map settings. For example, you can specify that the audit log only records failure information.

For a details about using the audit log, see the Map Audit topic.

Customized validation settings

Another way to troubleshoot a map is to customize the validation settings. Validation can be set to ignore certain aspects of the data during validation by setting custom options in the **Validation** map settings.

For example, you can choose that the validation process ignores item restrictions. This means that even if an item has restrictions defined, the restrictions are not considered when that item is validated.

Map Debugger

The debug functionality in the Map Designer can assist you in creating and troubleshooting maps. Use the Map Debugger when you have narrowed a problem down to a certain object or function call. For example, when you know that an object is not producing the correct output.

The map debug functionality allows you to break the execution of a map and inspect the current object values (both input and output objects) used within the currently executing output rule.

You cannot use the Map Debugger when the Map Profiler is enabled.

By enabling the Map Debugger, you can do the following:

- Identify breakpoints.
 - Step through rule execution.
 - View object values at a rule level.
- [Breakpoints](#)
 - [Rule execution](#)
 - [Viewing object values at a rule level](#)
 - [Running the Map Debugger](#)
 - [Remote debugging](#)

You can also use the Map Debugger to debug maps that run remotely on the supported operating systems.

Breakpoints

You can set breakpoints on the output objects of functional and executable maps.

- [Saving breakpoints](#)
- [Conditional BreakPoints](#)

To add a breakpoint

You can only add a breakpoint when the Map Debugger is enabled. See "[Running the Map Debugger](#)".

From an output card, right-click on an output object and select Add BreakPoint.
The output object rule with the breakpoint becomes highlighted.

To delete a breakpoint

Right-click on the output object and select Delete BreakPoint.

Saving breakpoints

When you save a map after adding breakpoints, the breakpoints are also saved. Therefore, the next time you open the map and enable the debugging option, the saved breakpoints will reappear. To remove the breakpoints, right-click on an object and select Delete All BreakPoints.

Conditional BreakPoints

You can configure the Map Debugger to stop on a specific iteration of a rule. For example, a rule repeats 99 times and the ninety-ninth occurrence is wrong. You can configure the Map Debugger to stop on the ninety-ninth iteration and subsequent iterations of the rule.

To set a conditional breakpoint:

1. Right-click on a card object and select Show BreakPoints.
The BreakPoints dialog box is displayed.
2. Select an object from the list and select Condition.
The BreakPoint Condition dialog box is displayed.
3. In the **Condition** field, select On Iteration.
4. In the **Value** field, enter a numeric value that represents the iteration of the rule that you want the Map Debugger to begin stopping at. For example, enter 99 for the Map Debugger to begin stopping on the ninety-ninth iteration of the rule.
5. Click OK.
When the map runs in this case, the Map Debugger stops at the ninety-ninth iteration of the rule and then every occurrence of the rule thereafter.

Rule execution

Starting from the first breakpoint, the debugging process can step through every rule on every card. The debugging process can step into functional maps and follow RUN maps that are defined within the same map source file (provided that the maps have been built with the Debug option turned on).

If any referenced functional maps or external RUN maps are found, the map enters step-over mode and stops at the end of the originating rule.

Viewing object values at a rule level

When processing a rule, the debugger displays relevant data. Relevant data is data that is read and written in the rule. The data currently being processed by a rule or function is displayed in the Debug Rule window.

The Rule editor in the Debug Rule window displays the current rule in a format of a hierarchical tree of objects, functions, and maps. At each breakpoint at which the map pauses, input and output values of objects, functions, and rules are shown.

Running the Map Debugger

When output from map execution is not what was expected, you can use the debug functionality to analyze each step of map execution.

To analyze map execution

1. In the Outline view, Composition view, Navigator view, or Extender Navigator view, select the specific map to debug.
 2. Click the Debug icon on the toolbar to enable debug mode.
 3. To set a breakpoint, right-click an output object and click Add BreakPoint on the context menu.
The object rule is highlighted in yellow or the color that you select in Preferences > Run/Debug.
 4. Build and run the map.
Map execution pauses when it encounters the object for which the break was positioned.
- The current rule is displayed in the Debug Rule window.
5. At this point, you can choose from the following options:
 - **Step In** to execute the next object or function in the rule. For objects, the current value appears. For functions, the output displays the result of the particular function.
 - At a rule level, the debug process proceeds to the next function or returns the value of the next object in the current rule.
 - At the function level, the object or map evaluated is updated, displaying the data that was returned.
 - At the object level, the value of the object is displayed.
 - At the function and map level, the result of the particular operation is displayed.
 - **Step Thru** to step through the rule until you recognize the problem. When the current rule executes the results are displayed. At a function level this option does not debug the remainder of the functions/objects in the rule.

- **Step Over** to move to the next rule in the current map.
 - **Continue** to continue map execution and stop at the next breakpoint if one exists, otherwise continue to the end of the map.
 - **Stop** to stop map execution. Execution will cease when all RUN maps have completed.
-

Remote debugging

You can also use the Map Debugger to debug maps that run remotely on the supported operating systems.

The following list includes the supported operating systems:

- HP-UX (Itanium processor)
- IBM® AIX® (RS/6000® processor)
- IBM z/OS®
- Linux® (Intel processor)
- Microsoft Windows
- Linux for Power Systems
- Linux for System z
- Sun Solaris (SPARC processor)

There are similarities between using the Map Debugger to debug maps that run locally on Windows, and remotely on the supported operating systems. You can set breakpoints on rules, inspect data during the transformation process, and use the same commands.

To use the Map Debugger, Design Studio (version 8.1 or later) is required on the local computer. Also, a minimum of Transformation Extender with Command Server (version 8.1 or later) must be installed on the remote computer where the executable map is run. The map must be built with version 8.1 or later of the Map Designer and then deployed to the remote computer.

On the remote computer, you must first start the debugging server (for example, with Telnet) before you can begin the debug process from the Map Designer. When you use Telnet, you can stop the debugging server by entering `Ctrl+C` in the Telnet command window.

- **Debugging a map remotely.**

These instructions assume that you have built a map using the Map Designer, version 8.1 or later, and have deployed the executable map file to the remote host.

Debugging a map remotely

These instructions assume that you have built a map using the Map Designer, version 8.1 or later, and have deployed the executable map file to the remote host.

Before you begin, access the remote host, for example, using Telnet, and set up your IBM Transformation Extender environment.

1. Start the debugging server by running the applicable file, located in the product installation directory:
 - `StartDebugServer.bat` (Windows)
 - `StartDebugServer.sh` (UNIX, z/OS UNIX)

By default, the debugging server uses port 2590. Specify `-PORT port_number` to use a different port. To resolve resource aliases while running a map under the debugger, use the `-MRC .mrc_file` option to specify the resource configuration file. For example, run `StartDebugServer -PORT 1234 -MRC MYALIASES.MRC`

To stop the debugging server, press `Ctrl+C`.

2. On the local computer, open the map source file for the remote executable map to be debugged.
3. In the Outline view, Composition view, Navigator view, or Extender Navigator view, select the specific map to debug.
4. Click the Debug icon on the toolbar to enable debug mode.
This also enables the Remote option.
5. Add one or more breakpoints to the output.
From an output card, right-click an output object and click Add BreakPoint.
6. Click Map>Remote.
The Remote Debugging window is displayed.
7. Under Remote Host, enter the remote computer information pertaining to the map to be debugged. In the Host field, enter the alias name or IP address of the remote computer. In the Port field, the default port number (2590) to be used by the debugging server is displayed.
8. Under Platform, select the platform of the remote host.
9. Under Security, enter a user name and password that provides access to the remote host where the executable map resides.
10. In the Map field, select the browse button and navigate to the executable map to debug.
11. Click OK to begin the remote debugging process.
Similar to the regular map debugging process, a Debug window is displayed from where you can select various options, such as Step In, Step Thru, Step Out, Step Over, or Continue, after a break point is reached.

Trace files

The trace file is a text or XML file that records the steps of map execution. You can use the information in the trace file to diagnose invalid data or incorrect type definitions.

- **Trace for general data**

The trace file records map execution progress. You can enable the trace option from the map settings or from the adapter command line. You can trace the input data, the output data, or both.

- **Static file validation and tracing**

Transformation Extender (TX) validates a static input file in a compiled map only once, when the map loads, and produces a trace file only if the static file fails validation. TX validates other map inputs each time the map runs, and logs them in the map trace file as specified by the map settings.

- **Trace for XML data**

- [Trace for XML parser errors only](#)

Related reference

- [MapTrace settings](#)

Trace for general data

The trace file records map execution progress. You can enable the trace option from the map settings or from the adapter command line. You can trace the input data, the output data, or both.

An input data trace file includes a message for each input data object. The trace file contains the sizes and counts of data objects, and the data object position in the data stream. If the data object is invalid, the trace file includes the reason the data is invalid.

Static input files produce a trace file only if the static input fails validation when the map loads. A map trace file describes a valid static input as a reused work file.

When output data is traced, the trace file specifies the objects that are built and the output objects that evaluate to `none`.

Because performance can be impacted, enable the trace option only during debugging.

- [Creating a trace file](#)

You specify the name and location of the trace file in the map settings. If you ran a map and got a message that an input was invalid, turn on the input trace only. If you got a message that an output was invalid, turn on the output trace only. If you trace both input and output, the input messages are at the beginning of the file, followed by the output messages.

- [Trace file contents](#)
- [When data is valid](#)
- [Group validation](#)
- [Valid card objects](#)
- [Invalid data](#)
- [Invalid card objects](#)
- [Finding the important messages](#)
- [Existence of data](#)
- [Validation for invalid objects](#)
- [Unknown data](#)
- [Negative messages not fatal](#)
- [Example of optional occurrences](#)

Related concepts

- [Static input cards](#)

Creating a trace file

You specify the name and location of the trace file in the map settings. If you ran a map and got a message that an input was invalid, turn on the input trace only. If you got a message that an output was invalid, turn on the output trace only. If you trace both input and output, the input messages are at the beginning of the file, followed by the output messages.

Trace file contents

A trace file of input messages contains a message for each input data object. A message gives a date and time stamp, describes the level of the object in the card, the offset of the data object in the data stream, the length of the data object in bytes, its component number, the index of the component, a portion of the actual data, and the name of the type to which it presumably belongs.

In the trace file, the symbol `\r` indicates a carriage return. The symbol `\n` indicates a linefeed.

The DI number and X number are not used for debugging purposes and can be ignored.

For example, shown below is trace message for the Name Data object. The length (`len`) of the data (`Nicole`) is 6 bytes.

```
18:22:17.837 Feb 13 2000
(Level 2: Offset 0, len 6, comp 1 of 2, #1, DI 00000001:)
Data at offset 0 ('Nicole') was found to be of TYPE
X'0004' (Name Data).
```

The level of an object indicates the position of the data object with respect to the entire card object. A card object has level 0; a component of the card object has level 1, and so on.

The trace file describes a static input card as a reused work file. See the static input card description for details.

Related concepts

- [Static input cards](#)

When data is valid

During data validation, a given data object is examined, and compared to its type definition. If a data object matches the type definition, the object is valid and the trace file indicates the object was found to be of that type. For example, in the trace message below, the data **Nicole** is a valid **Name** data object. It was found to be of the type **Name**.

```
18:22:17.837 Feb 13 2000
(Level 2: Offset 0, len 6, comp 1 of 2, #1, DI 00000001:)
Data at offset 0 ('Nicole') was found to be of TYPE
X'0004' (Name Data).
```

Group validation

A group is valid if the syntax is valid and all of its components are valid. A group is validated component-by-component. A component is valid if its type is valid, its component range is valid, and, if the component has a rule, the rule evaluates to "true".

When a component rule is evaluated, any invalid data used in a component rule is evaluated with a value of "none".

After all of the components of a group have been found, a message in the trace file indicates that the group itself was found.

For example, the components of **Row** are **Name** and **Phone#** and the data for **Row** is:

```
Nicole,352-2929
```

To find the object **Row** in the trace file, first the components of **Row** are found. The trace file contains a message for **Name**, then one for **Phone#**, and one for **Row**.

Valid card objects

If all of the objects that make up a card data object are found, the card object is valid. There is a trace message indicating this status. For example, if the data object for card 1 was valid, the trace message "INPUT 1 was valid" appears in the trace file. The message for a card object appears after all of the messages for the objects within that card.

For the Database object of an input card, the type **Database** is composed of **Row(s)**, and **Row** is composed of **Name** and **Phone#**.

The data for the entire input database is:

```
Nicole,352-2929
David,377-8098
```

In the trace file, there are messages for each **Row** and then there is a message for the entire card object, **Database**.

Invalid data

When a data object does *not* match the definition of the type to which it presumably belongs, it could be invalid.

Possible reasons for an invalid data object:

- Item value is not in restriction list
- Item failed the presentation test
- Item is the wrong size
- Data object has the wrong delimiter
- Data object has the wrong terminator
- Data object failed the component rule
- Group is missing a required component
- Group contains one or more invalid component(s)

Invalid card objects

If an entire card object is invalid, a runtime message appears in the Command Server window, indicating that the data is invalid. For example, if an input source is a file and the file is invalid, when the map is run, the message "One or more inputs was invalid" may appear.

If you trace the card containing the invalid object, a message in the trace file indicates that the card data object was invalid. If the data object of input card 1 is **File** and is invalid, the trace file would indicate this with the message "Input 1 exists, but it is invalid".

Prior to the message that the entire card object is invalid, there is an explanation of why it is invalid. For example, it might be that a required component of the card object is in error. The message about the invalid component will contain a negative word such as *not*, *failed*, *wrong*, or *invalid*.

Finding the important messages

The best way to read the trace file is to open it in a word processor and use the search facilities. The following tables explain the trace options to choose and what to look for in the trace file, based on the runtime message.

Table 1. Runtime message: One or more inputs was invalid

Limit trace to:	Read this in the trace file:
The summary	Read the card summaries to find out which card is invalid.
The invalid card	Start searching from the end of the trace file and look for the word <code>invalid</code> in the trace file.

Table 2. Runtime message: Input valid, but unknown data found

Limit trace to:	Read this in the trace file:
The summary	Read the card summaries to find out which card contains the unknown data.
The card containing the unknown data	Start searching from the end of the trace file, and look for the word <code>invalid</code> in the trace file.

Existence of data

When a certain data object is validated, determining the existence of that object occurs before fully validating all of the components within that object.

For example, if a data object is a partition and includes an initiator, as soon as the initiator is found, it is known that the data object exists. A message exists in the trace file, indicating that the data object is "known to exist."

The existence of a certain object can be determined if that object has certain properties. For a detailed explanation of how the existence of an object is detected, see the Type Designer documentation.

Example of existence of record objects

The following example shows how the existence of record objects can be determined. An input file is composed of records with three kinds of records: **A**, **B**, and **C**.

Record is partitioned into **A**, **B**, and **C**, the three types of records. Each record begins with a letter followed by an asterisk. The letter indicates the kind of record. This value is defined as the initiator for each record type. For example, an **A** Record begins with `A*`.

Shown below is the data:

A*couch,1
B*plate,12
A*chair,6
C*sweater,15
B*glass,8

Because the initiator partitions **Record**, as soon as the initiator of a given record is found, that record is known to exist.

Validation for invalid objects

If an object is determined to be invalid, and that object is known to exist, validation continues. For example, there was an error in the first **B Record**.

It is known that **B Record** exists. When **B Record** is found to be invalid, validation continues with the subsequent records.

If an object is *not* known to exist, the object is found as unknown data.

Unknown data

When input data is validated, additional data that does not validate as part of the input may also be found. This data is unknown data.

An example input is composed of **Product(s)** and **Product** is *not* defined with properties that allow its existence to be predetermined.

The second **Product** is invalid because the **Quantity** field is missing.

When the invalid **Product** is found, it is not known to exist. Therefore, it is assumed to be unknown data.

Sometimes a message about unknown data is not significant. You may get a message about unknown data because there are extra carriage returns or "junk" data at the end of an input file. You should read the trace file to determine whether all of the pertinent input data was found.

Negative messages not fatal

Sometimes validation of a given data object interprets the given data object as belonging to a type to which it does not really belong. In this situation, the data object will fail to belong to that type - but that is because the data does not belong.

In the following two situations, you may see messages that a data object fails to belong to a certain type because it is not really data of that type:

- When an optional occurrence of a component that does not exist in the data is sought during validation
- When an object of a partitioned type is sought during validation

Example of optional occurrences

If an occurrence of a component is optional, there may be more than one possibility for a given data object. For example, suppose a type has the following components:

A (1:3)

B

After **A** is found, the next object in the data stream can either be **A** or **B**. Suppose it is **B**. Therefore, the object fails to be **A**. A message in the trace file indicates that the object does not match the definition of **A** in some way. After the validation process has determined that the object is not **A**, validation occurs to determine whether it is **B**. Then the object is validated as **B**.

It is acceptable if an *optional* occurrence of a component does not exist in the data; it is optional. However, if a *required* occurrence of a component does not exist, the data object containing that component is invalid.

The optional **Message** is missing from the data. After finding the **Header**, the next data object searched for is **Message**. However, the next object in the data is not a **Message**, it is a **LineItem**.

The trace file indicates that the initiator of the data object is not **M** which is the initiator of a **Message**. After it is determined that the data object is not a **Message**, the **LineItem** is validated as a **LineItem**.

If you see a message in the trace file saying that a data object does not exist, this is not necessarily the error causing the entire input object to be invalid. Keep in mind the data being validated, and the optional data. Your understanding of optional data is important when reading the trace file.

Example of partitioned types

Another case in which a data object might fail to belong to a certain type is when the type is partitioned. If an object that is presumably a partition is validated, the validation process goes down the list of partitions to determine the partition to which the object belongs. You might see a message that indicates an object failed to be a certain partition. That means it does not belong to *that* partition, it may belong to one of the other partitions. The validation process checks to see whether it belongs to the next partition, and so on.

For example, suppose **Record** is partitioned into **A**, **B**, and **C**.

Each record begins with an initiator: **A***, **B*** or **C***.

First, validation checks the initiator to determine whether it is an **A Record**. The initiator will be the wrong value for an **A Record**. Next, validation checks the initiator to determine whether it is a **B Record**. The initiator will be the wrong value for a **B Record**. Next, validation checks to determine whether it is a **C Record** and the object is validated as a **C Record**.

Viewing results in hex mode

Run results can be viewed in hex mode, which displays the offset values of the data. This may be helpful if the trace file indicates a problem at a particular offset in the data. To view run results in hex mode, select the **Display In Hex Mode** check box in the **Run Results** dialog box.

Static file validation and tracing

Transformation Extender (TX) validates a static input file in a compiled map only once, when the map loads, and produces a trace file only if the static file fails validation. TX validates other map inputs each time the map runs, and logs them in the map trace file as specified by the map settings.

When a static input file fails validation, the map does not load. TX logs the validation error in the appropriate log for the runtime environment for example, the Launcher's CompoundSystem log or the IBM Integration Bus log. The error includes the location of the static input trace file. TX creates the static input trace file as *map_name.mtr* in the map directory. If the map loads from memory, TX creates the *map_name.mtr* file in the map working directory.

When a static input file fails validation on a z/OS® system running in batch, CICS® or IMS, the trace is written to the DD name DTX#TRCE. Because the trace is written as a byte stream, it lacks formatting if the DTX#TRCE is directed to SYSOUT. If formatting is required, direct the DD card to a zFS file. For example:

```
//DTX#TRCE DD PATH='/tmp/dtx#trce.mtr',
//          PATHDISP=(KEEP),
//          PATHOPTS=(ORDWR,OCREAT),
//          PATHMODE=(SIRWXU,SIRWXG,SIROTH)
```

When you trace a static input file and the static file passes validation, the map trace file describes the static input as an existing work file. For example, you might trace two map inputs. Input 1 is not a static file and the input is invalid. Input 2 is a static file that passed validation during load. The trace file includes messages similar to the following:

```
.
.
.

(Level 2: Offset 389, len 6, comp 3 of 3, #1, DI 000000000053:)
Data at offset 389 ('00000<CR>') is INVALID data of TYPE
  X'0012' (total_hours element example).

(Level 1: Offset 297, len 98, comp 1 of 1, #4, DI 000000000054:)
Data at offset 297 ('Work A.        ...') is INVALID data of TYPE
```

```

X'0003' (timesheet example).

(Level 0: Offset 0, len 396, comp 1 of 0, #1, DI 000000000055:)
Data at offset 0 ('FirstName      ...') is INVALID data of TYPE
X'0002' (all_employees example).

INPUT 1 exists, but its type is in error.

End of Validation messages for INPUT CARD 1.

Reusing an earlier work file for INPUT CARD 2.

End of Execution messages.

```

Trace for XML data

An XML-formatted trace file that contains XML validation information can be created during map execution for maps that contain XML schemas. When enabled, the XML trace file (*mapname_log.xml*) is generated in the map directory in addition to the general trace file. The XML trace file is not generated when no input card is present in the map.

For maps that contain XML schemas (created using Xerces), the input XML validation information that is generated to the regular trace file (.mtr) or that is viewable on the Trace tab of the Organizer is not valid. See the XML trace file (*mapname_log.xml*) for accurate XML input validation information.

When viewed with Microsoft Internet Explorer, the XML trace file:

- Contains a validation log.
 - Displays the XML validation sequence.
 - Presents validation results.
- When using a browser other than Microsoft Internet Explorer, only the content of the XML file is displayed.

Similar to the general trace, you must enable certain **MapTrace** settings to create an XML trace file.

- [Creating an XML trace file](#)
 - [XML validation log](#)
 - [Validation sequence](#)
 - [Validation results](#)
-

Creating an XML trace file

Enable certain **MapTrace** settings to create an XML trace file.

To create an XML trace file:

1. In the Extender Navigator view of the Design Studio, right-click the map that contains references to XML schema, and click Map Settings.
2. Set the MapTrace> Switch setting to ON.
3. Set the MapTrace> InputContentTrace setting to All.

After you run the map, the XML trace file is created. You can find the XML trace file (*mapname_log.xml*) in the map directory, unless otherwise specified. The file opens in your default browser. When you run the map again, the existing file is overwritten.

XML validation log

The Source tab displays the XML validation log. The XML validation log is a step-by-step representation of the XML validation process. This log presents the time of map execution, input file information, and other validation details.

Validation sequence

The Validation Sequence tab displays the reverted order in which the XML validation library identified the types from the XML input file.

The XML validation sequence can be useful in troubleshooting, for example, to see where (or on which type) the validation process stopped.

Validation results

The Validation Result tab displays the result of the XML validation process. When XML validation does not complete successfully, the applicable error messages are displayed. If map execution failed early in the validation process, this tab might not contain any information.

To locate the error, open the XML input file, go to the fourteenth line, and then 10 characters into the line.

The following XML code represents the XML input file for this example. The validation error is associated with **GenDate**. **GenDate** is required and there is no value present.

```

<?xml version="1.0" ?>
- <PO>
```

```

- <Header>
  <PONumber>C1234-3452545</PONumber>
  <PODate>08-08-01</PODate>
  <Contact Name="Darryl Brown" Phone="555-444-1212" />
- <Instruction Type="NONE">
  <Addressee>2000 E. Overlook Hwy</Addressee>
  <AddressLine />
  <City>Boca Raton</City>
  <State>FL</State>
  <Zip>33431</Zip>
</Instruction>
<GenDate />
</Header>
- <LineItem>
  <CatalogUPC />
</LineItem>
</PO>

```

Trace for XML parser errors only

At map run time, IBM Transformation Extender automatically generates an error log when an XML input file that the map is processing contains an XML error and the `DTX_XML_LOG_ON_ERROR=TRUE` environment variable is set. The error log file contains information about the XML parser error only. You can use the information to diagnose the invalid data or incorrect type definitions in the XML input file.

In distributed environments, the log file is called `wtxParserError_log.xml` and is created in the map directory. If the log file exists already, the error is appended to it.

In z/OS Batch environments, the log file is directed to the `DSTXXLOG` DD name.

Error and warning messages

- [Map execution error and warning messages](#)
- [Map execution warning messages](#)
- [Map build error messages](#)
- [Map build warning messages](#)
- [Map compile error messages](#)
- [Audit log status codes](#)

Map execution error and warning messages

After a map is executed, map execution messages indicate the execution results. If audit is enabled, these messages are also recorded in the appropriate audit log for the map.

The following list describes the codes and messages that can be returned as a result of running a map.

The return codes marked with an asterisk (*) are warning codes. See [Map Execution Warning Messages](#) for a consecutive list of the warning codes for a map.

The return codes marked with a dagger symbol (†) are those codes that, when returned from a failed WebSphere DataPower map, indicate that there are additional map failure causes specific to DataPower maps. See [Error messages for WebSphere DataPower maps](#) for a consecutive list of the error codes for a WebSphere DataPower map.

Return Code

Message

0	Map completed successfully
1†	User aborted map If you cancel the Server before the map completes, this message is displayed.
2	Not enough memory to execute map There is not enough memory to initialize the map. Try closing all other running applications.
3	Could not open map The .mmc file cannot be found. Ensure the file name or path is correct.
4†	Could not read map If you have ported your map and sent it with FTP to another platform, make sure you have done so using the BINARY option so the map's contents will not be corrupted.
5	Could not read inputs
6	Invalid map handle
7	Invalid card number was specified Your map may be corrupt.

- 8 One or more inputs was invalid
Turn the input trace on, run the map again and read the trace file.
- 9† Target not available
An incorrect file name or path for a target may be specified.
- 10† Internal error
- 11 Could not build one or more outputs
An attempt may have been made to write to a read-only directory, or an output file directory may be missing.
- 12† Source not available
An incorrect file name or command for a source may be specified.
- 13 Could not open work files
An invalid path for work files may be specified.
- 14* One or more outputs was invalid
A number produced an overflow condition. Enable the **Trace Output** option, run the map again, and read the trace file.
- 15 Map must be recompiled
You may be trying to run a map with a server that has a different version from the Map Designer version used to compile the map. Make sure the Map Designer and the Server versions match.
- 16 Disk write error
An attempt may have been made to write to a read-only directory or file. Also, check the amount of available disk space.
- 17 Disk read error
An attempt may have been made to access a file on a shared resource that is not accessible. Also, check the amount of available disk space.
- 18* Page usage count error
Your data mapped correctly; however, an internal paging problem exists.
- 19 Internal calling error
- 20 Reopen file failed
Your map did not run correctly.
- 21* Input valid but unknown data found
Enough of the input data to conform to the card definition was recognized, but there was more data at the end of the input stream. Enable the **Trace Input** setting, re-run the map and read the map trace file (.mtr).
- 22 Page size too small
Increase the page size or decrease the number of types in the schema that are used to define the input.
- 23 Unable to reuse work file
Use the same page settings that were used when you created the existing work file.
- 24 Database error
A problem was encountered attempting to write to an output database. Read the database trace file (.dbl) for information about possible causes.
- 25 File attribute error
The Command Server or Launcher failed to write data to either the work or data file. Check the output card definition on the map to see if it matches the platform-specific data definition specified for the file the map is attempting to create.
- 26* Output type in error
The entire card object is in error.
- 27* Output type contains errors
One or more components failed restrictions or a component rule.
- 28* Input type contains errors
This occurs when mapping data of a component that has a RESTART assigned to it, and at least one instance of the component in the input data is invalid.
- 29*

	Output valid, but unknown data found This occurs when enough of the output data is recognized to conform to the card definition, but there was more data at the end of the output stream.
30	FAIL function aborted map This occurs when the FAIL function is used within the map to cause map execution to cease. The return message specified as the second argument of the FAIL function appears in place of the xxx in the Execution Audit section of the Audit Log. See the Functions and Expressions documentation for more information about using the FAIL function.
31	(No longer used)
32	Invalid map instance handle
33	Map instance handle in use
34*	Too few pages requested, more allocated The map can complete but it can be better optimized by increasing the page count.
35	Insufficient number of pages to execute map The map could not complete because not enough pages were specified.
36	Could not open work files A scratch file for the HANDLEIN function could not be created.
37†	Operation is not supported on the platform
50	Not enough memory to execute map There is not enough memory to initialize the map. Try closing all open applications.
51	Card override failure There is a problem with the command line entered for an -I or -O option. (Applicable to Command Server)
52	I/O initialization failure A run map was called and could not allocate memory for input or output.
53†	Open audit failure The audit file was not accessible.
54	No command line There was no map or command line specified. (Applicable to run maps and the Command Server)
55	Recursive command files A command file was specified within a command file. (Applicable to run maps and the Command Server)
56	Invalid command line option - <option> An invalid argument was entered on the command line. (Applicable to run maps and the Command Server)
57	Invalid 'W' command line option An invalid parameter for the -W option was specified. (Applicable to run maps and the Command Server)
58	Invalid 'B' command line option An invalid parameter for the -B option was specified. (Applicable to run maps and the Command Server)
59	Invalid 'R' command line option An invalid parameter for the -R option was specified. (Applicable to run maps and the Command Server)
60	Invalid 'A' command line option

- An invalid parameter for the **-A** option was specified.
 (Applicable to run maps and the Command Server)
- 61 Invalid 'P' command line option
 An invalid parameter for the **-P** option was specified.
 (Applicable to run maps and the Command Server)
- 62 Invalid 'Y' command line option
 An invalid parameter for the **-Y** option was specified.
 (Applicable to run maps and the Command Server)
- 63 Invalid 'T' command line option
 An invalid parameter for the **-T** option was specified.
 (Applicable to run maps and the Command Server)
- 64 Invalid 'G' command line option
 An invalid parameter for the **-G** option was specified.
 (Applicable to run maps and the Command Server)
- 65 Invalid 'I' command line option for input <2>
 An invalid parameter for the **-I** option was specified for
 input card <number>.
 (Applicable to run maps and the Command Server)
- 66 Invalid size in echo command line for input < 2 >
 A size (nnn) was specified for **-IE2Snnn** that is outside of the boundaries for input card <number>.
 (Applicable to run maps and the Command Server)
- 67 Invalid adapter type in command line for input <2>
 Using the **-IAadapter_name** or **-IMadapter_name**, <adapter_name> is not a valid adapter type.
 (Applicable to run maps and the Command Server)
- 68 Invalid 'O' command line option for output <1>
 An invalid parameter for the **-O** option was specified for output card <number>.
 (Applicable to run maps and the Command Server)
- 69 Invalid adapter type in command line for output <1>
 Using the **-OAadapter_name** or **-OMadapter_name**, adapter_name is not a valid adapter type.
 (Applicable to run maps and the Command Server)
- 70 Command line memory failure
 Memory allocation failure during processing of command line.
 (Applicable to run maps and the Command Server)
- 71 Invalid 'D' command line option
 An invalid parameter for the **-D** option was specified.
 (Applicable to run maps and the Command Server)
- 72 Invalid 'F' command line option
 An invalid parameter for the **-F** option was specified.
 (Applicable to run maps and the Command Server)
- 73 Resource manager failure
 The resource manager could not reserve the resources (sources/targets) due to a resource contention that is most likely a run map.
 (Applicable to run maps and the Launcher)
- 74 Invalid 'Z' command line option
 An invalid parameter for the **-Z** option was specified.

(Applicable to run maps and the Command Server)

75	Adapter failed to get data on input During the commit/rollback phase, the source had a failure
76	Adapter failed to put data on output During the commit/rollback phase, the target had a failure.
77	Invalid map name: <map_name> The map path and name exceed 2000 bytes.

(Applicable to run maps and the Command Server)

Map execution warning messages

The following table lists the return codes that are warnings only - meaning that the map did complete, but some problems occurred. You can change the default behavior for the warning codes by specifying **Fail on Warnings** or **Ignore Warnings** through the map settings or by using the -F or -Z execution command.

The warning codes marked with an asterisk (*) are returned only when at least one output is listed in the **Data Audit Settings** of the **Organizer** and in the **Map Settings** dialog box, **MapAudit** is enabled with **Data** enabled for the **Log** setting.

Return Code	Message
14	One or more outputs was invalid A number produced an overflow condition. Enable the Trace Output option, run the map again, and read the trace file.
18	Page usage count error Your data mapped correctly; however, an internal paging problem exists.
21	Input valid but unknown data found Enough of the input data to conform to the card definition was recognized, but there was more data at the end of the input stream. Enable the Trace Input setting, run the map again and read the map trace file (.log).
26*	Output type in error The entire card object is in error.
27*	Output type contains errors One or more components failed restrictions or a component rule.
28	Input type contains errors This occurs when mapping data of a component that has a RESTART assigned to it, and at least one instance of the component in the input data is invalid.
29*	Output valid, but unknown data found This occurs when enough of the output data is recognized to conform to the card definition, but there was more data at the end of the output stream.
34	Too few pages requested, more allocated The map can complete but it can be better optimized by increasing the page count.

Map build error messages

Build errors prevent a map from being compiled. If build errors occur, view the errors, correct the problem(s), and build the map again. A map cannot be run if it has not been built.

In the following build messages; these letters are used as variables:

- x = the map where the error occurred
- y = the output that contains the rule in error
- w = number
- z = number

DataPower maps in addition to the ones listed below. See [Map build errors for DataPower maps](#) for a consecutive list of these additional map build error codes for a DataPower map.

Return Code	Message
M100	Analysis aborted by user.
M101	

Map: x
Unresolved rules exist.

Hint: Go to map x. If you want to use the unresolved rules, copy and paste them. Delete any unwanted unresolved rules.

M102

Map: x Output: y
Functional map referenced in rule must have one (and only one) output.

Map referenced: *map-name*.

Hint: Go to the functional map. Delete the extra card(s).

M103

Map: x Output: y
Number of arguments for functional map *map-name* is incorrect.

Number specified: #*specified* Number required: #*required*

Hint: Add or delete arguments from the rule, or add or delete input cards in the functional map.

M104

Map: x Output: y
Empty rule cell. Rule required.

Hint: Go to map x. Enter rules in each empty rule cell.

M105

Map: x Output: y
Rule references unknown type: *type name*.

Hint: You may have made a change in the schema. If the card object is unknown, edit the card, and choose a valid type. You may have incorrectly typed in the name. If you drag and drop the type, the correct name is automatically entered.

M106

Map: x
Circular map reference from map *map-name*.

Analysis must terminate.

Hint: Two different maps contain rules that reference each other. Remove the reference from one of the maps.

M107

Map: x Output: y
Rule syntax is invalid starting at character position: *position#*.

Rule

Hint: This error may occur if a previous version did not catch the syntax error. Generally, syntax errors are caught as rules are entered. See the [release notes](#) for any changes in rule syntax.

M108

Map: x Output: y
Rule references unknown map: *map-name*.

Hint: Re-enter the map name in the rule. You may have typed the map name incorrectly. Map names are case-sensitive. You may have forgotten to create the functional map. If so, create it.

M109

Map: x Card: *cardname*
Card is invalid.

Hint: The type name is either misspelled or you may have made a change in the schema by deleting or renaming the type. Edit the card, and choose a valid type.

M110

Map: x Card: *cardname*
Card type is not a group or item.

Hint: You probably changed your schema. Edit the card, and choose a group or item.

M111

Map: x Output: y
Argument #w for map *map-name* does not match type of input card #z.

Hint: Change argument #w so that it matches the type of input card #z, or change the type of input card #z so that it matches argument #w.

M112

Map: x Output: y
Argument #w for map *map-name* does not match item sub-class of card #z.

Hint: Change argument #w so that it matches the item sub-class of input card #z, or change the type of input card #z so that its item sub-class matches argument #w.

M113

Map: x Output: y
Argument #w for function *function name* is invalid.

Hint: Select the rule for output *y*. Right-click the map rule and choose Properties on the context menu. The Rule Properties dialog box allows you to view function arguments and object properties. You have either referenced an invalid object, or used the wrong type of argument.

M114

Map: *x* Output: *y*
Output argument of rule does not match output type: *rule*.

Hint: The map rule is resulting in a data object of the wrong type. Check the functions and the maps used within the map rule and make sure that the entire rule evaluates to the correct type.

You may have dragged and dropped a group that is a different type to this output. Use a functional map instead. If the rule contains the name of a functional map, go to the functional map and edit the output card. Make sure the type of the output card is the output *y*.

M115

Map: *x* Output: *y*
Output argument of rule does not match output item sub-class: *rule*.

Hint: Notice the output *y*. The output argument of the rule must have the same item sub-class as *y*.

M116

Map: *x* Output: *y*
Type Tree for this card is not used in executable map.

Hint: Edit the card and select a schema that is used in the executable map.

M117

Map: *x* Output: *y*
Cannot assign an output to itself.

Hint: You might have accidentally dragged and dropped an output into its own rule cell. Edit the rule.

M118

Map: *x* Output: *y*
Rule references unknown comment: *commentname @ inputname*

Hint: The input card type does not have a floating component, or you have incorrectly typed in the name. If the input card has a floating component, on the Card menu, the Show Floating Component Type option will be activated.

M119

Map: *x* Output: *y*
Output argument of rule is not a partition of output type.

Hint: If the rule contains the name of a functional map, go to the functional map and edit the output card. Make sure the type of the output card is the output *y*, or a partition of the output *y*.

M120

Map: *x* Output: *y*
Operand is an invalid *operand*.

Hint: For information on operands, see the Functions and Expressions documentation.

M121

Map: *x*
Input card #*cardnumber* uses invalid schema.

Hint: You may have moved the schema or map source from its original location. You may have incorrectly entered the name of the schema.

M122

Map: *x*
Output card #*cardnumber* uses invalid schema.

Hint: You may have moved the schema or map source from its original location. You may have incorrectly entered the name of the schema.

M123

Map: *x* Card: *cardname*
Card updates input, but card type and data file do not match an input.

Hint: Edit the card and re-define the update information.

M124

Map: *x* Output: *y*
Rule references type later in output: *rule*

Hint: If possible, enter an equivalent rule in a later output. An output referenced in a rule must appear above the rule within the output card.

M125

Map: *x*
Invalid audit: *audit statement*

Hint: Look at the Audit Settings window. An asterisk * appears next to any invalid audit statement. Re-enter the audit statement.

Map build warning messages

A build warning alerts you of a condition that might be problematic. If you build a map and there are warnings, but no errors, the map is compiled. You can ignore warnings. To ignore warnings, from Window > Preferences > Transformation Extender, select Suppress Build Warnings.

Return Code
Message

M200

Map: x Output: y
WARNING: Size of input item is greater than size of output item.

M201

WARNING: Functional maps that return groups or partitioned items must not be used as input to functions.

Map compile error messages

The compile errors display on the Build Results tab of the **Organizer**. Maps with compile errors may not be run. Correct all compile errors to run a map.

The following table lists the errors than can result when a map is built:

Return Code

Message

M1001

Compile aborted by user.
Hint: You cancelled the build process before it completed.

M1002

Not enough memory available to compile.
Hint: Close other applications that you have open, and build the map again.

M1003

Couldn't open the map source file.
Hint: The map may be corrupt. Please contact Technical Support.

M1004

Couldn't find the executable map.
Hint: The map source file may be corrupt. Please contact Technical Support.

M1005

Duplicate map names in output file.
Hint: If you get this error, please contact Technical Support.

M1006

Tree used in functional map not used by executable map.
Hint: Edit the card in the functional map or the executable map so both cards reference the same schema.

M1007

No delimiter defined for delimited type.
Hint: In the Type Designer, specify the delimiter value for the given type.

M1008

Unable to open the schema for input card.
Hint: The schema may not exist in the directory specified, or the schema may be corrupt. If your schema is corrupt, please contact Technical Support.

M1009

Unable to open the schema for output card.
Hint: The schema may not exist in the directory specified or the schema may be corrupt.

M1010

Unable to find a type in an input schema.
Hint: If you get this error, please contact Technical Support.

M1011

Unable to find a type in an output schema.
Hint: If you get this error, please contact Technical Support.

M1012

Could not retrieve query information.
Hint: Either you have not installed the Database Interface Designer or the .mdq file is incorrect.

M1013

Could not retrieve database information.
Hint: Either you have not installed the Database Interface Designer or the .mdq file is incorrect.

M1014

Input data file for executable map missing or invalid.
Hint: Check the file names used for sources and targets.

M1015

Output data file for executable map missing or invalid.
Hint: Check the file names used for sources and targets.

M1016

Parser found an invalid rule.

Hint: If you get this error, contact Technical Support.

M1017

Output card referenced in rule not found.

Hint: If you get this error, please contact Technical Support.

M1018

Invalid component or partition subtype encountered.

Hint: If you get this error, please contact Technical Support.

M1019

Unable to resolve type references in rule.

Hint: If you get this error, please contact Technical Support.

M1020

Couldn't open the compiled map file.

Hint: The compiled map file may be corrupt. Please contact Technical Support.

M1021

Error writing to the compiled map file. (Disk full?)

Hint: Delete unnecessary files from your hard disk.

M1022

Unable to resolve type references in audit.

Hint: If you get this error, please contact Technical Support.

M1023

Unable to open .MDQ file.

Hint: The .mdq file does not exist or is otherwise inaccessible.

M1024

Database not found.

Hint: The named database is not defined in the specified .mdq file.

M1025

Database Error.

Hint: An unexpected error occurred while processing and .mdq file.

M1026

Input MDQ file not found.

Hint: The .mdq file does not exist or is otherwise inaccessible.

M1027

Input query not found.

Hint: The named query is not defined in the specific .mdq file.

M1028

Output MDQ file not found.

Hint: The .mdq file does not exist or is otherwise inaccessible.

M1029

Output database not found.

Hint: The named database is not defined in the specified .mdq file.

M1030

At least one syntax object literal is too long.

Hint: Type Designer analysis is required.

Audit log status codes

The status attribute in the Audit Log is a letter followed by a code number. The letter of the status attribute tells whether the data is valid (V), or caused an error (E), or caused a warning (W).

- [Valid audit log status codes](#)
 - [Warning audit log status codes](#)
 - [Error audit log status codes](#)
-

Valid audit log status codes

The following table explains the number in the status code if the letter is V.

Status Code	Description
-------------	-------------

V00

The object and all contained objects are valid

V01

The object is valid, but it contains invalid object(s)

Warning audit log status codes

The following table explains the number in the status code if the letter is W.

Status Code	Description
W01	Object failed restriction
W02	Object failed presentation
W03	Object failed size check

Error audit log status codes

The following table explains the number in the status code if the letter is E.

Status Code	Description
E00	Object is an unidentified foreign object (UFO) - data not associated with any particular type
E01	Object failed restriction
E02	Object failed presentation
E03	Object failed size check
E04	Invalid or missing initiator
E05	Invalid or missing terminator
E06	Object missing required component
E07	Object is invalid because it contains components in error
E08	Object failed partitioning
E09	Object failed component rule
E10	This required object is missing
E11	Invalid or missing delimiter
E12	More instances of an object exist than have been specified (Only implemented for unordered groups)
E13	Object failed to meet minimum size requirement.
E14	Object failed to meet maximum size requirement.

Flows

A flow node invokes a sub flow within a flow. Sub-flows need not be wired to other nodes; it is valid to have a flow that is comprised of multiple sub-flows that each have listeners.

It is equally valid to wire up the flow -terminals of sub-flows to other sub-flows or other nodes.

Introduction to Flows

This documentation describes how Flows can be run from REST API, listeners, or they can be scheduled.

- [Nodes](#)
- [Terminals](#)
- [Flow terminals](#)
- [Scheduled flows](#)
- [Listener](#)
 - A Flow that has a Map node as its first node, and uses a File adapter for an input, can enable that node's input to be a Watch.
- [Flow audits](#)
 - Flow audits are a way to retrieve more verbose information about a flow instance.

Nodes

- [**Map**](#)
A Map node invokes a ITX map within a flow.
- [**Flow**](#)
A flow node invokes a sub flow within a flow. Sub-flows need not be wired to other nodes; it is valid to have a flow that is comprised of multiple sub-flows that each have listeners.
- [**Source and Target**](#)
Source and Target nodes provide IBM Transformation Extender Design Server with an outside-in approach to developing integrations. The inside-out approach means that you begin with a map, add inputs and outputs, and then configure these to access resources. Connections might be created in the process. The outside-in approach implements the integration from the opposite direction. You begin with the outside interfaces, then connects them to provide maps and other artifacts.
- [**Request node**](#)
The request node has a single input request terminal and a single output response terminal. Similarly, to the source node, it allows data to be retrieved from a data source. The request node differs from the source node in that it accepts input data from a request. This input data can influence or determine the data that is retrieved from the data source. The ability of the request node to accept request input is one way to introduce dynamic queries into the execution of a flow.
- [**Cache Read and Write**](#)
This documentation describes the function and use of the Cache Read and Cache Write nodes.
- [**Clone**](#)
The Clone node has one input and two output terminals. This node clones, or copies the input data from the single input terminal to both output terminals.
- [**Decision**](#)
The flow variable on which to make a decision.
- [**Fail**](#)
Fails the flow execution.
- [**Format Converter Node**](#)
The Format Converter Node can be used to quickly convert data from one format to another.
- [**JSON Read Node**](#)
This documentation describes the function and use of the JSON Read node.
- [**JSON Transform Node**](#)
This documentation describes the function and use of the JSON Transform node.
- [**JAVA**](#)
Invokes a java class, performing user defined functionality based on the properties specified for the java class to act on the input.
- [**Join**](#)
The Join node gathers the individual results and appends to a single output file or terminal.
- [**Log**](#)
This node logs the raw data from the node input into the file and propagates the data from the input to the output terminal.
- [**Passthrough**](#)
The Passthrough Node propagates data from the input to the output terminal.
- [**REST Client**](#)
The REST Client flow node provides a simple and powerful way to access REST services. It can be used to directly invoke REST APIs or can use prebuilt configurations that define the APIs of the service. The node supports:
- [**Route**](#)
The Route node provides a way to route data conditionally to one or more outputs of the node. The node bases decisions by evaluating a condition for a flow variable and determining whether to send the data to output 1 or 2 or both, based on the result of the condition.
- [**Sleep**](#)
This node suspends the execution of the flow for the specified number of milliseconds
- [**SOAP Client Node**](#)
- [**Split**](#)
A Split Node should be used when there is a need to split CSV or JSON data processing or when a map can run in a burst mode to process data in batches. This might occur when data processing becomes excessively time consuming.
- [**Status Node**](#)
A Status Node can only be used as the first node in a “status flow”, as that what makes a flow a status flow. Similar to an initialization flow, a status flow may be referenced from any other main flow. A status flow has a Status node as its first node which has Success, Error and Always output terminals that provide the flow status JSON to a downstream node.

Map

A Map node invokes a ITX map within a flow.

Using the Map node

Once the Map node is deployed to the flow canvas, you can select the map from the list of maps existing in the project. To replace a map in the Map node, use the Replace Node setting and select a different map from the list of maps. To control the map's behavior, use the Settings option. Map settings are explained in the map documentation under Project > Maps.

When the map is executed within the Map node, the transaction scope of adapters invoked from the map rules or from input and output cards that are not wired or are not attached to the flow terminals is set to map. This means that the adapters commit, or rollback transactions based on the Map node's status, not the flow's final status. If the input or output card is attached to the flow terminal, the transaction scope of that card is flow. This means that the transaction will be committed or rolled back based on the flow's status. The map audit is produced in JSON format and is part of the overall flow audit.

Map node Terminals

You can use the Edit Terminals option to show or hide map input or output terminals. If the terminal is shown, it can be wired to other nodes or set as a flow terminal. If the Map node is the first node, multiple map node terminals set as flow terminals. Multiple output map node terminals can also be set as flow terminals. Only one map

node input terminal can be wired to the previous node. If multiple map node input terminals are shown, those that are not wired can have the input card input modified. Multiple output terminals can be wired to other nodes.

When the input or output Map node terminal is wired, the adapter originally assigned to the map is overwritten with the internal stream adapter that passes data in memory in or out, depending on the terminal type. To control the maximum size of data being passed on this stream in memory, in megabytes, use the streamSize setting from the config.yaml file.

- Log terminal: Produces the full map audit in the JSON format.
- Status terminal: Produces the map status and values of flow variables attached to the Map node in the JSON format, if the map execution is successful. If the map fails, no data is produced on this terminal, and any downstream nodes wired to this terminal are not executed.
- Error terminal: Produces the map status and values of flow variables attached to the Map node, if the map execution is unsuccessful. If this terminal is shown, the map error code will not be propagated to the flow, and the map failure will not result in the flow's failure. Essentially, the error is captured within the Map node.

Types of Maps

The Map node supports both maps that have map source code and compiled maps. If the Map node uses a map that has the source code, the Open Map option can be used to modify the map. If the map is an existing compiled map, the Open Map option is not available.

Flow

A flow node invokes a sub flow within a flow. Sub-flows need not be wired to other nodes; it is valid to have a flow that is comprised of multiple sub-flows that each have listeners.

It is equally valid to wire up the flow-terminals of sub-flows to other sub-flows or other nodes.

Note: Flows are not supported on AIX and z/OS.

Source and Target

Source and Target nodes provide IBM Transformation Extender Design Server with an outside-in approach to developing integrations. The inside-out approach means that you begin with a map, add inputs and outputs, and then configure these to access resources. Connections might be created in the process. The outside-in approach implements the integration from the opposite direction. You begin with the outside interfaces, then connects them to provide maps and other artifacts.

- [Introduction to nodes](#)
The function of these nodes are as follows.
- [Node settings](#)
Access the node settings by right-clicking on the node in the flow.

Introduction to nodes

The function of these nodes are as follows.

- [Source Node](#)
- [Target Node](#)

Source Node

The source node has a single output terminal, and it supports watches if the selected connection type supports watches. Source node has an input terminal. By having an input terminal, it enables the existing watch mechanism and allows the input to be overridden, such as when running the flow from the designer when you want to bypass the node and pass data from a file instead. This node supports success, error, and log terminals.

Target Node

The Target node has a single input terminal and a single output terminal.

By having an output terminal, the output can be overridden when it is run. This node supports success, error, and log terminals.

Node settings

Access the node settings by right-clicking on the node in the flow.

The settings are not displayed in a modal dialog like other node settings. Instead settings are displayed in an accordion dialog. When displayed it looks like the card settings of a map. This should be like the edit card settings for a map card in a flow.

There are differences in the properties:

1. The action properties should not include map related properties such as transaction scope, backup, and retry. These properties are only available in the context of a map.
2. The schema allows you to select a schema or generate a schema for certain connection types (for example JDBC, and Salesforce). The schema page, however, is not required in order to specify a schema. This allows a map to be automatically generated between existing nodes.

Request node

The request node has a single input request terminal and a single output response terminal. Similarly, to the source node, it allows data to be retrieved from a data source. The request node differs from the source node in that it accepts input data from a request. This input data can influence or determine the data that is retrieved from the data source. The ability of the request node to accept request input is one way to introduce dynamic queries into the execution of a flow.

The request node is analogous to a map node whose map uses the GET function.

```
GET ("adapter-alias", "command-line", request-data)
```

Like the GET function in maps, the request node supports a variety of data source types.

As for the source and target nodes, the settings categories of the request node are Connection, Action Properties, and Schema. The Connection category allows an existing connection or a command line to be used. When a command line is used, the type of the data source of the request node must be elected. The syntax and meaning of the command line value depends on the data source type. The available options for the Action Properties category vary depending on the data source type.

The transactions are committed at the node level if the data was retrieved or modified successfully (adapter's GET operation was successful).

Example Use cases:

- REST APIs where the PUT or POST requires a body, and a response is returned in the HTTP response
- A SAP BAPI call has input parameters and a response to the BAPI call
- A converter that converts data from one form to another (e.g. SOAP adapter)
- A cipher adapter encrypts or decrypts the requested data

Process to create a request node

Request node can be placed as a first node, last node or between any other two nodes on canvas.

Node settings are:

- Connection: choose existing connection or select Use Command Line to specify the adapter and its command line.
- Define success and failure actions.
- Optimally, choose a schema that defines the node's data.

Logging

The node produces a log that consists of the node's internal messages and messages that underlying adapter generates. The adapter messages are prefix with the adapter name. The level of details depends of the flow execution log level.

Example log may look like (subject to change):

0: Called adapter EXCEL.
1: The adapter command line was: -WRKSHEET invoice1
2: The adapter success_action was set to: create
3: The adapter failure_action was set to: commit
4: EXCEL: Worksheet | invoice1
5: EXCEL: Excel Format | false
6: EXCEL: Trace file | __memory__
7: EXCEL: Trace append | true
8: EXCEL: Trace error | false
9: EXCEL: Trace verbose | false
10: EXCEL: Excel document data passed as GET parameter from the map
11: EXCEL: Reading Excel document data from the worksheet...
12: EXCEL: Total Rows in the Excel document: 5
13: EXCEL: Excel document data has been successfully passed to map!
14: EXCEL adapter performed the commit operation.
15: EXCEL retrieved 193 bytes.

Cache Read and Write

This documentation describes the function and use of the Cache Read and Cache Write nodes.

- [**Cache Read and Cache Write nodes configuration**](#)

This documentation describes how to configure the Cache Read node and the Cache Write node.

- [**Cache Read**](#)

The Cache Read node reads key/value pairs from the global cache, or from the flow variables. Optionally, it can delete keys and read from the cache.

- [**Cache Write**](#)

The Cache Write node writes key/value pairs to the global cache, or to the flow variables based on the scope property

Cache Read and Cache Write nodes configuration

This documentation describes how to configure the Cache Read node and the Cache Write node.

When a Cache Read or Cache Write node scope is set to Cache, Redis is used as an external cache to store the key/value pairs. In this case, the following are applicable:

<install_dir>/config.yaml file or environment variables are used to configure the cache.

<install_dir>/config.yaml file:

redis.host=localhost - defines the redis server hostname.

redis.port=6379 – defines the redis server port.

Environment variables can be used to override the settings from config.yaml file:

server.variableCacheType=External

redis.host=hip-server-redis

redis.port=6379

By default, the Cache is enabled, and it uses the local Redis server installed with the product.

Cache Read

The Cache Read node reads key/value pairs from the global cache, or from the flow variables. Optionally, it can delete keys and read from the cache.

The Cache Read node consists of one input terminal (Input) where the key values are passed in, and two output terminals (Matched and Unmatched).

The matched keys are written to the Matched terminal. Unmatched keys are written to the Unmatched terminal.

Data is sent to the Input terminal as a set of keys or a single key and each key is looked up and returned in the response. If the Key Prefix property is set, it is prepended to each key value to provide name-spacing. The input can optionally contain values, but they will be ignored.

Example 1: If the Data Format is delimited, and the delimiters are set to defaults, the data can either be sent as shown here:

Key1\n

Key2\n

or:

Key1,valA,ValB\n

Key1,valA,ValB\n

In the later case, everything between the first comma and the newline (value) is ignored.

Example 2: If the Data Format is JSON array:

```
[  
  {  
    "key": "Key1",  
    "value": "Value1"  
  },  
  {  
    "key": "Key2"  
  }]
```

The Key1 value set to Value1 is ignored.

Example 3: If the Data Format is None, the node could then be used as a source node, with no data passed to input terminal. A new property “Key1” is enabled to specify the name of the key.

Example 4: If the Key Pattern is specified, the node could then be used as a source node, with no data passed to input terminal. If the scope is Flow Variable, the pattern is a regular expression pattern (regex) and all keys that match the regex are retrieved. If the scope is Cache, the pattern is a glob style pattern and all keys that match the glob are retrieved.

Example regular expression:

a.b matches a1b or acb.

a*b matches acdefb

Example glob pattern:

a?b matches a1b or acb.

a*b matches acdefb

- **Settings**

This documentation describes the settings available for the Cache Read node.

- **Node Logging**

When executed the node reports matched keys into the log.

Settings

This documentation describes the settings available for the Cache Read node.

- **Cache Scope**

Determines whether to read values from flow variables, or from the Cache.

- **Data Format**

The format of the incoming input and output data.

- **Key Delimiter**

The Key Delimiter is enabled when the Data Format is Delimited.

- **Record Delimiter**

The Record Delimiter is enabled when Data Format is Delimited.

- **Include Key**

The Include Key is enabled when the Data Format is Delimited.

- **Key Prefix**

Prefix the key values with this value. This property can include flow variables.

- **Delete Key**

Determines whether or not to delete the matched keys from the cache after reading the value.

- **Default Key Value**

If the Default Key Value is set, nothing is ever sent to the unmatched terminal. All unmatched keys will be assigned the Default Key Value and sent to the matched terminal.

- **Key Pattern**

The key pattern meaning depends on the Cache Scope.

Cache Scope

Determines whether to read values from flow variables, or from the Cache.

If the Cache is selected, the node will fail if the connection to the external cache cannot be established.

Data Format

The format of the incoming input and output data.

Supported data formats are JSON, and Delimited and None. Whatever format is presented to the input terminal will be produced on the output terminal as well. For example, if a JSON array is sent to the input terminal, the output terminal will also produce a JSON array. If the Data Format is None, a new property "Key" is enabled to specify the name of the key to be retrieved and send to the output.

- **JSON data format**

JSON array and JSON object formats are described in this documentation.

- **Delimited data format**

The data sent to the input terminal is delimited according to the Record Delimiter properties.

JSON data format

JSON array and JSON object formats are described in this documentation.

JSON Array

If an array, each object in the array must have a key named **key**:

```
[  
  {"key": "MyKey1"},  
  {"key": "MyKey2"}  
]
```

The matched terminal produces an array of JSON objects with **key** and **value** keys:

```
[  
  {"key": "MyKey1", "value": "MyVal1"},  
  {"key": "MyKey2", "value": "MyVal2"}  
]
```

```
{"key": "MyKey2", "value": "MyVal2"}
```

The unmatched terminal produces an array of JSON objects with a single key named **key**:

```
[  
  {"key": "MyKey1"},  
  {"key": "MyKey2"}  
]
```

If there are no results on matched or unmatched terminals, an empty array will be returned.

JSON Object

If a JSON object is sent to the input terminal the names of the fields of the object specify which keys to lookup in the cache. The value of the keys is irrelevant and can be null, or any other value:

```
{  
  "MyKey1": null,  
  "MyKey2": 1  
}
```

The matched terminal will populate the values of the fields:

```
{  
  "MyKey1": "MyVal1",  
  "MyKey2": "MyVal2"  
}
```

The unmatched terminal will provide an object that has fields with null values:

```
{  
  "MyKey1": null,  
  "MyKey2": null  
}
```

If there are no results on matched or unmatched terminals, an empty object will be returned.

Delimited data format

The data sent to the input terminal is delimited according to the Record Delimiter properties.

It is comprised of one or more keys, for example:

```
MyKey1,MyVal1  
MyKey2,MyVal2
```

MyKey1

MyKey2

or if Include Key property is set to false, then the response will just be values:

```
MyVal1  
MyVal2
```

The data produced on the unmatched terminal will be a list of keys:

```
MyKey1  
MyKey2
```

Key Delimiter

The Key Delimiter is enabled when the Data Format is Delimited.

The Key Delimiter is the delimiter between the key and the value.

The default is a comma.

Record Delimiter

The Record Delimiter is enabled when Data Format is Delimited.

The Record Delimiter is the delimiter for each end of each record.

The default is newline.

Include Key

The Include Key is enabled when the Data Format is Delimited.

The Include Key determines whether or not the output should include the key.

Key Prefix

Prefix the key values with this value. This property can include flow variables.

For example: %_FLOWID_%

Delete Key

Determines whether or not to delete the matched keys from the cache after reading the value.

Default Key Value

If the Default Key Value is set, nothing is ever sent to the unmatched terminal. All unmatched keys will be assigned the Default Key Value and sent to the matched terminal.

Key Pattern

The key pattern meaning depends on the Cache Scope.

If the Cache Scope is Flow Variable, the pattern is a regular expression pattern (regex). If the scope is Cache, the key pattern is a glob style pattern. When the pattern is defined, the node can be a 1st node and the input data is not required. If the node is not a 1st node or the data exist on the node input terminal, the data is ignored. Matched keys in a format defined with Data Format property is sent to the Matched terminal. No data is sent to the Unmatched terminal. Consult online documentation for details about regex and glob style patterns.

Node Logging

When executed the node reports matched keys into the log.

Cache Write

The Cache Write node writes key/value pairs to the global cache, or to the flow variables based on the scope property

The Cache Write node consists of one input terminal, where the key values are passed, and one output terminal. The node propagates data from input terminal to the output terminal.

Data is sent to the Input terminal as a set of keys and values. If the Key Prefix property is set, it is prepended to each key value to provide name-spacing.

Example 1: If the Data Format is delimited, and the delimiters are set to defaults, and the data format is expressed as follows:

```
Key1,Value1\nKey2,Value2
```

The record delimiter is an option on the last record.

Example 2: If the Data Format is a JSON array

```
[  
  {  
    "key": "Key1",  
    "value": "Value1"  
  },  
  {  
    "key": "Key2",  
    "value": "Value2"  
  }]
```

Key1 is set to Value2 and so forth.

Example 3: If the Data Format is a JSON Object.

```
{"Key1":"Value1", Key2:123}
```

Key1 value is set to string "Value1" and Key2 integer value 123 is set to string "123".

All cache values are stored as strings, but the value provided could be in any format supported by JSON.

Example 4: If the Data Format is None, a new property Key is displayed. The entire input is written to the key. A flow variables can be used to define a key.

Scope = Cache

Key = Code%COLOR%

%COLOR% = "Red"

Input data sent to the terminal is: "Extremely Hot Temperature"

The node will create cache entry:

CodeRed="Extremely Hot Temperature"

- **Settings**

This documentation describes the settings that are available in the Cache Write node.

- **Node Logging**

When executed, the node reports the keys that are set in the cache.

Settings

This documentation describes the settings that are available in the Cache Write node.

- **Cache Scope**

Cache scope determines whether to write values to flow variables or to the cache.

- **Data Format**

The format for the incoming and output data. Supported data formats are JSON, Delimited and None.

- **Key Delimiter**

The Key Delimiter is enabled when the Data Format is Delimited.

- **Record Delimiter**

The Record Delimiter is the delimiter for the end of each record. The value is read from the Key Delimiter to the Record Delimiter.

- **Key Prefix**

Prefix the key values with this value. The property can include flow variables.

Cache Scope

Cache scope determines whether to write values to flow variables or to the cache.

If caching is only required within a single flow run, flow variables are used.

Data Format

The format for the incoming and output data. Supported data formats are JSON, Delimited and None.

If the Data Format is delimited, and the delimiters are set to defaults, the example data format is the following:

```
Key1,Value1\nKey2,Value2
```

The record delimiter is an option on the last record.

If the Data Format is a JSON array, for example the following:

```
[  
  {  
    "key": "Key1",  
    "value": "Value1"  
  },  
  {  
    "key": "Key2",  
    "value": "Value2"  
  }  
]
```

Key1 value is set to Value1, Key2 value is set to Value2.

If the Data Format is a JSON object, for example:

```
{ "Key1": "Value1", Key2: 123 }
```

Key1 value is set to string "Value1" and Key2 integer value 123 is set to string "123".

All cache values are stored as strings, but the value provided could be in any format supported by JSON.

If the Data Format is None, a new property Key is displayed. The entire input data is written to the key defined with the Key property.

Key Delimiter

The Key Delimiter is enabled when the Data Format is Delimited.

The Key Delimiter is the delimiter between the key and the value.

The default is a comma.

Record Delimiter

The Record Delimiter is the delimiter for the end of each record. The value is read from the Key Delimiter to the Record Delimiter.

Key Prefix

Prefix the key values with this value. The property can include flow variables.

For example: %_FLOWUID_%

Node Logging

When executed, the node reports the keys that are set in the cache.

Clone

The Clone node has one input and two output terminals. This node clones, or copies the input data from the single input terminal to both output terminals.

Decision

The flow variable on which to make a decision.

If the condition is met, routes the input data to the true terminal. If the condition is not met, routes the data to the false terminal. The decision is made based on the Flow Variable value and the operator.

- [Settings](#)

Settings

This section lists the settings available for the Decision node.

- [**Flow Variable**](#)
The decision is made based on the Flow Variable value.
- [**Operator**](#)
The decision is made based on the Operator.
- [**Case Sensitive**](#)
Whether to perform case sensitive or insensitive comparison.
- [**Value**](#)
Specifies the value of the property.

Flow Variable

The decision is made based on the Flow Variable value.

Operator

The decision is made based on the Operator.

Supported operators are:

- Equals
- Not equals
- In List
- Not In List

- Less Than
 - Greater Than
 - Less Than or Equal To
 - Greater Than or Equal To
-

Case Sensitive

Whether to perform case sensitive or insensitive comparison.

Value

Specifies the value of the property.

You can compare the Flow variable value using the Operator and Value of this property.

If the Operator is In List or Not In List, provide a comma-separated list of values.

Fail

Fails the flow execution.

When the Fail node executes, the flow fails immediately. The Error Message property provides the fail message. Flow variable substitution is supported in the flow message property.

- [Settings](#)
This section lists the settings available for the Fail node.
-

Settings

This section lists the settings available for the Fail node.

- [Error Message](#)
Provides the information text about the flow failure.
-

Error Message

Provides the information text about the flow failure.

The default error message is Fail from [nodename] node.

Format Converter Node

The Format Converter Node can be used to quickly convert data from one format to another.

The Format Converter Node supports these formats:

- CSV
- Delimited
- JSON
- YAML
- XML

When converting from JSON, YAML, or XML to CSV or delimited, there are limitations because CSV data is flat and cannot convey the hierarchical structures that may be in the JSON, YAML, or XML documents.

When converting from JSON or YAML to CSV or delimited the location of an array of objects is provided to the node by using the Input Array Path property. The node looks at the first element of the array to determine the set of fields that are included in the output CSV. This assumes that each object within the array has the same set of fields. If subsequent objects have more fields than the first node, these additional fields are ignored.

For example, given the JSON:

```
{  
    "addresses": [  
        {  
            "street": "10 Main St",  
            "city": "Smallville",  
            "state": "GA"  
        },
```

```

        {
            "street": "20 High St",
            "city": "Springville",
            "zip": "12345"
        }
    ]
}

```

The generated CSV is as shown:

```
street,city,state
10 Main St,Smallville,GA
20 High St,Smallville,
```

Since the second object does not contain a state key, that value is null additionally, although the second record contains a zip key, that is ignored because the fields are determined from the first object.

- [Converting from XML to CSV or Delimited](#)

When converting from XML to CSV or delimited, the XML elements to convert to CSV are selected by an XPath expression.

- [Using the Format Converter Node](#)

These instructions describe how to use the Format Converter Node in a flow.

Converting from XML to CSV or Delimited

When converting from XML to CSV or delimited, the XML elements to convert to CSV are selected by an XPath expression.

After obtaining a list of elements that are described by the XPath expression, the node looks at the first element it encountered to determine the set of fields that will be included in the output CSV.

The node first examines the element to see if it contains child elements. If it does, then these child elements define the set of CSV fields. If it does not contain child elements, the node examines the element to see if it contains attributes. If it does, then these attributes define the set of CSV fields. This assumes that each element within the set of elements has the same set of child elements or attributes. If subsequent objects have more fields than the first node, these additional fields are ignored.

For example, either of these XML documents produce the save CSV as shown in the JSON example:

XML with child elements:

```
<addresses>
    <address>
        <street>10 Main St</street>
        <city>Smallville</city>
        <state>GA</state>
    </address>
    <address>
        <street>20 High St</street>
        <city>Springville</city>
        <zip>12345</zip>
    </address>
</addresses>
```

XML with attributes:

```
<addresses>
    <address street="10 Main St" city="Smallville" state="GA"/>
    <address street="20 High St" city="Springville" zip="12345"/>
</addresses>
```

Using the Format Converter Node

These instructions describe how to use the Format Converter Node in a flow.

To use the Format Converter Node:

1. Select the Format Converter Node from the palette and drag it onto the canvas.
 2. Right-click on the node to select Settings.
 3. Configure the node as required, depending upon the data being converted. The following common properties are available:
 - Input Format - Select CSV, JSON, YAML, XML or Delimited to correspond to the format of the data that you want to pass to the input terminal of the node.
 - Output Format - Select CSV, JSON, YAML, XML or Delimited to correspond to the format of the data that you want to pass to the output terminal of the node.
- [CSV input format - available properties](#)
These properties are available if the desired input format is CSV
 - [CSV input format Delimited - available properties](#)
These properties are available if the input format is Delimited:
 - [CSV output format - available property](#)
One property is available if the output format is CSV.
 - [CSV output format delimited - available properties](#)
These properties are available if the output format is Delimited:
 - [JSON or YAML input, CSV output - available properties](#)
One property is available if the input format is JSON or YAML and the output format is CSV.
 - [JSON or YAML output, CSV input or delimited- available properties](#)
This property is available if the output format is JSON or YAML and the input format is CSV or Delimited:

- [XML input, CSV output or delimited- available properties](#)

This property is available if the input format is XML and the output format is CSV or delimited:

CSV input format - available properties

These properties are available if the desired input format is CSV

- Input Has Header - Set this property if the CSV data contains a header row in the first row of the data.
- Fields - If the CSV data does not contain a header row, the header can be specified in this property as a comma-separated list. This provides a means of specifying the header if it is not in the incoming data.

CSV input format Delimited - available properties

These properties are available if the input format is Delimited:

- Fields - Specify the names of the fields that are in the delimited data as a comma-separated list of field names, e.g. "id,name,age".
- Input Field Delimiter - The character or string that delimits fields. If not specified, the default is the pipe character '|'.
- Input Row Delimiter - The character or string that terminates a row. If not specified, the default is the newline character '\n'.
- Input Escape Character - The character to escape field and row delimiters if they appear in the data. By default, there is no escape character. If the field and row delimiters cannot appear in your data, do not define an escape character, since doing so adds processing overhead.

CSV output format - available property

One property is available if the output format is CSV.

Available property:

- Output Has Header - Set this property if you want the output data to include a header row.

CSV output format delimited - available properties

These properties are available if the output format is Delimited:

- Output Field Delimiter - The character or string that delimits fields. If not specified, the default is the pipe character '|'.
- Output Row Delimiter - The character or string that terminates a row. If not specified, the default is the newline character '\n'.
- Output Escape Character - The character to escape field and row delimiters if they appear in the data. By default, there is no escape character. If the field and row delimiters cannot appear in your data, do not define an escape character, since doing so adds processing overhead.

JSON or YAML input, CSV output - available properties

One property is available if the input format is JSON or YAML and the output format is CSV.

Property available:

- Input Array Path - When you convert from JSON or YAML to CSV, the incoming JSON or YAML must contain an array of JSON or YAML objects. This array can either be at the root of the document, or elsewhere in the document. If the entire document is an array of objects, the Input Array Path should not be set. Otherwise, specify the path of the array using dot-notation. For example, if the JSON data is:

```
{  
    "data": {  
        "id": 1,  
        "people": [  
            {  
                "name": "Fred",  
                "sex": "male"  
            },  
            {  
                "name": "Susan",  
                "sex": "female"  
            }  
        ]  
    }  
}
```

the path is specified as data.people.

JSON or YAML output, CSV input or delimited- available properties

This property is available if the output format is JSON or YAML and the input format is CSV or Delimited:

Property available:

- Output Array Path - When converting from CSV or delimited to JSON or YAML the output that is produced will convert the input rows to an array of JSON or YAML objects. This property determines the location of that array in the produced JSON or YAML. If the property is not set, the JSON or YAML array will be at the root of the document. If a path is specified, intermediate JSON or YAML objects will be created. For example, if the output array path is specified as "people", the produced JSON will look like this:

```
{  
    "people": [  
        {  
            "name": "Fred",  
            "sex": "male"  
        },  
        {  
            "name": "Susan",  
            "sex": "female"  
        }  
    ]  
}
```

XML input, CSV output or delimited- available properties

This property is available if the input format is XML and the output format is CSV or delimited:

Property available:

- Input XML XPath - When converting from XML to CSV or delimited, the output will be created from repeating node elements in the XML. The Input XML XPath property is an XPath expression that selects nodes from the XML to be converted to the output. The path can either be an absolute path or some other valid XPath specification. For example, consider this XML:

```
<XML>  
    <Winners>  
        <Winner>  
            <Name>Janet Gaynor</Name>  
            <Movie>Seventh Heaven</Movie>  
        </Winner>  
        <Winner>  
            <Name>Mary Pickford</Name>  
            <Movie>Coquette</Movie>  
        </Winner>  
    </Winners>  
</XML>
```

If we want to create CSV or delimited data that contains the Winner nodes the Input XML XPath can be set to:

- /Winners/Winner – this is an absolute path that navigates to a specific location in the XML
- //Winner – this XPath expression will find Winner elements anywhere in the XML document

JSON Read Node

This documentation describes the function and use of the JSON Read node.

The JSON Read node is used to set flow variables or cache variables to the value of elements extracted from a JSON document. It can also be used to extract elements from a larger JSON document, to perform simple JSON transformation.

The JSON Read node performs following:

- Examine the JSON document that is sent to the node's input link. It then processes each path or query expression that is provided in the node's Paths property to find the specified JSON element in the input document.
- If the Variable Scope property is set to Flow Variables, the named flow variables are set with the value of the JSON element. Alternatively, if the Variable Scope property is set to Cache, the named cache variables are set with the value of the JSON element.
- If the Output Content property is set to Filter the Input, the node's output is a JSON object with fields containing the JSON elements specified by the Paths. Alternatively, if the Output Content property is set to Propagate the Input, the input JSON is copied to the output terminal without modification.
- **Settings**
This documentation describes the settings available for the JSON Read node.
- **Examples**
These examples show the functionality of JSON Read node.

Settings

This documentation describes the settings available for the JSON Read node.

- **Paths**
Specifies as a table of paths to find in the input JSON document.
- **Variable Scope**
Variable Scope specifies the target for the path expressions.

- **Output Content**

Output Content specifies whether to propagate the input JSON document to the output, or whether to output a new JSON object that contains just the elements specified by the paths.

Paths

Specifies as a table of paths to find in the input JSON document.

The table has two columns:

- Path – The path or query expression that references elements in the JSON input document. Such expressions should comply with the syntax defined by Stefan Goessner's de-facto standard defined here: <https://tools.ietf.org/id/draft-goessner-dispatch-jsonpath-00.html>. Filter expressions are supported. The path might reference either a JSON text, numeric or boolean primitive, or can specify an object or array.
- Variable/Key – The name of the flow variable or cache variable. If filtering the output, this also specifies the name of the field in the filtered output. The specified name can include flow variable references which are resolved when the flow is run. For example, if you specify %_FLOWUUID_%_id to prefix the name of the variable with the ID of the flow run.

Variable Scope

Variable Scope specifies the target for the path expressions.

It can be set to one of these values:

- Flow Variables – Set flow variables.
- Cache – Set cache variables.
- Set no Variables – Do not set any variables. Use this option if you want to filter the input JSON and have no need to set flow variables or cache variables.

Output Content

Output Content specifies whether to propagate the input JSON document to the output, or whether to output a new JSON object that contains just the elements specified by the paths.

Simple JSON transformation can be accomplished by specifying this setting as Filter the Input and Variable Scope to Set no Variables.

Examples

These examples show the functionality of JSON Read node.

Example 1: The use case is to set a flow or cache variable variables to the Person_ID, City and Full_Address of some record.

The input data is:

```
{  
  "id": 123,  
  "name": "fred",  
  "address": {  
    "street": "101 Main St",  
    "city": "London"  
  }  
}
```

and the Paths table is set as:

Path	Variable/Key
\$.id	Person_ID
\$.address.city	City
\$.address	Full_Address

If the Variable Scope is set to flow or cache variables it will set these 3 variables:

Path	Variable/Key
Person_ID	123
City	London
Full_Address	{"street": "101 Main St", "city": "London"}

Note: If the path returns a JSON object or array, the value is the serialized form of that object or array.

If the Output Content property is set to filter the input, then the output in this example will be:

```
{  
  "Person_ID": 123,  
  "City": "London",  
  "Full_Address": {  
    "street": "101 Main St",  
    "city": "London"  
  }  
}
```

Example 2: This example demonstrates the query expressions.

The use case is to set a flow variable to the mobile phone number of some record.

The input data is:

```
{  
  "phones": [  
    {  
      "type": "mobile",  
      "number": "111-222-3333"  
    },  
    {  
      "type": "home",  
      "number": "444-444-5555"  
    }  
  ]  
}
```

The path expression to achieve this is:

Path	Variable/Key
<code>\$.phones[?(@.type == "mobile")].number</code>	mobile

This means in the "phones" array, find the object whose "type" field is "mobile" and then return the "number" field from this object. This will result in this flow variable:

Name	Value
mobile	111-222-3333

If the Output Content property is set to filter the input, then the output in this example will be:

```
{  
  "mobile": "111-222-3333"  
}
```

JSON Transform Node

This documentation describes the function and use of the JSON Transform node.

The JSON Transform node is used to transform JSON documents from one form to another. The JSON Transform node uses JSON path expressions embedded within a JSON template to build the output document. The template is specified as a property of the node and describes what the output data should be, and where the contents of the JSON should come from.

The JSON Transform node also supports setting the value of fields to the value of flow variables.

- [JSON Template Property](#)
The JSON template property must be a valid JSON. It can be either a JSON object or a JSON array. The template property can be specified as %variable-name%, in which case the template will be obtained from the named flow variable or configuration variable.
- [Examples](#)
These examples show the functionality of JSON Transform node.

JSON Template Property

The JSON template property must be a valid JSON. It can be either a JSON object or a JSON array. The template property can be specified as %variable-name%, in which case the template will be obtained from the named flow variable or configuration variable.

The template can include any arrays, objects, primitives, and these can also be specified as JSON path expressions. Such expressions should comply to the syntax defined by Stefan Goessner's de-facto standard defined here: <https://tools.ietf.org/id/draft-goessner-dispatch-jsonpath-00.html>.

The JSON Transform node uses the syntax "@@path" to indicate that the string contains a JSON path expression. The path is then evaluated, and its results are included in the output document to replace the path expression in the template.

As can be seen from the examples, where the JSON path expression is specified in the template determines the expected result of the JSON path expression.

- [Path to a Primitive](#)
To fetch a field that is a primitive (string, number, boolean) specify the JSON path as a string primitive:
- [Path to an Object](#)
There are two ways to specify an object in the template.
- [Path to an Array](#)
If the path is specified as a string member of an array, then the path will return an array.

Path to a Primitive

To fetch a field that is a primitive (string, number, boolean) specify the JSON path as a string primitive:

```
"field": "@@$.id",
```

The value of field will be set to whatever is returned from the specified path.

Path to an Object

There are two ways to specify an object in the template.

The first is to use the special field value "*":

```
"object": { "*": "@@$.address" }
```

The special field value causes the node to replace the entire object in the template with object specified by the JSON path.

Specifying the object as shown has the exact same result.

```
"object": "@@$.address"
```

The former may be preferable since it is clearer from looking at the template that the expected type that the JSON Path references is an object.

Path to an Array

If the path is specified as a string member of an array, then the path will return an array.

For example:

```
"array_of_objects": [ "@@$.phones" ],
"array_of_primitives": [ "@@$.numbers" ],
```

This indicates that the phones field and numbers fields return arrays. Even if the path points to something that is not an array, the result will be an array because it was specified that way in the template.

If the path references an array, and the template element is not an array, then only the first element in the array will be returned. To include all elements in the array, specify the [`<path>`].

Examples

These examples show the functionality of JSON Transform node.

Example 1: If the template is set as

```
{
  "literal": "a literal",
  "field": "@@$.id",
  "var": "%my_var%",
  "array_of_objects": [ "@@$.phones" ],
  "array_of_primitives": [ "@@$.numbers" ],
  "object": { "*": "@@$.address" }
}
```

and the given JSON expression passed to the input of the node:

```
{
  "id": 123,
  "name": "fred",
  "address": {
    "street": "101 Main St",
    "city": "London"
  },
  "numbers": [ 1, 2, 3 ],
  "phones": [
    {
      "type": "mobile",
      "number": "111-222-3333"
    },
    {
      "type": "home",
      "number": "444-444-5555"
    }
  ]
}
```

the node will generate this as the output:

```
"literal": "a literal",
"field": 123,
"var": "Value of my_var",
"array_of_objects": [
  {
    "type": "mobile",
    "number": "111-222-3333"
  },
  {
    "type": "home",
    "number": "444-444-5555"
  }
],
```

```

"array_of_primitives": [
  1,
  2,
  3
],
"object": {
  "street": "101 Main St",
  "city": "London"
}
}

```

To create this output, the node traversed the template to build the output. It produced these fields:

- literal – The value was set to a literal value.
- field – The value of the field is obtained from the path expression `$.id`, which has the value 123 in the input document.
- var – The value is specified as the name of a flow variable named city, the value of which is set in the output.
- array_of_objects – The value of the field is obtained from the path expression `$.phones`. Because the template specified this path within an array, the path expression is expected to return an array.
- array_of_primitives – An array of primitive values is returned by specifying the path expression `$.numbers` within array syntax.
- object – An object is replaced by the fetched value by specifying the special key value "*" in the object. If this is found, the object in the template is replaced by the object returned by the `$.address` path.

Example 2: This example demonstrates the query syntax.

To determine the ID of the service named "Upload".

The JSON sent to the node is:

```

{
  "services": [
    {
      "_id": "5f9ae9765bffe304360d10d6",
      "name": "Upload"
    },
    {
      "_id": "5f9ae9765bffe304565656b3",
      "name": "Download"
    }
  ]
}

```

The template contains a query expression (as indicated by the '?')

```
{
  "id": "@@$.services[?(@.name == \"Upload\")._id"
}
```

This query finds the object in the "services" array which has its "name" field set to "Upload". It then returns the "_id" field from that object to produce this output:

```
{"id": "5f9ae9765bffe304360d10d6"}
```

JAVA

Invokes a java class, performing user defined functionality based on the properties specified for the java class to act on the input.

The Java node presents a simplified interface to invoke a Java class as part of the flow execution. It enables to perform a custom functionality written in Java programming language and be a part of the flow definition and execution. It is similar to an exit point where user can plug-in their custom functionality as a part of the flow execution to achieve the intended objective and the purpose of the flow.

- **[Settings](#)**
Following are the Java node settings that user must specify to create a flow definition.
- **[Copying Java nodes zip file under the non-docker environment](#)**
- **[Copying Java nodes zip file under the docker environment](#)**

Settings

Following are the Java node settings that user must specify to create a flow definition.

- **[Class name](#)**
The fully qualified Java class extending the abstract class SimpleNode.
- **[Properties](#)**
Properties are name-value pairs that indicate an action to perform, or any general-purpose information that user's Java class will understand and perform an activity that serves the purpose of the node in the flow definition and execution during the runtime.

Class name

The fully qualified Java class extending the abstract class SimpleNode.

The Java node requires user to specify the Java class name to load and provide the properties recognized by the Java class to perform actions accordingly on the input received and generate output, pass on the generated output to subsequent nodes in the flow, if any present for the further processing.

Java class name specified in the node settings must implement the abstract class SimpleNode to work during the flow execution. The SimpleNode interface provides the user defined custom Java class, the input stream to process the input and output stream to pass on the output. It generates the next node in the flow, besides the standard success, error, and log terminals to disseminate the success, error, and audit log information to the nodes connected to the those terminals.

The SimpleNode interface allows the Java class to set, get, increment, and decrement a flow variable from the Java node in a flow. Flow variables are process data variables that goes with the flow execution and accessible to all nodes in the flow that includes Java nodes while a flow is being executed. Flow variables are name-value pairs, where name is a case-sensitive string that is associated with a specific value in string format.

The location of the SimpleNode interface and documentation:

```
<install_dir>/examples/dk/nodes/java/simple/interface
```

Location of the example Java classes that implement the SimpleNode interface, which are part of *com.hcl.hip.examples* package:

```
<install_dir>/examples/dk/nodes/java/simple/nodes
```

Location of an example flow that utilizes multiple Java nodes using the example Java classes (that implements the SimpleNode interface):

```
<install_dir>/examples/flowengine/simplejava
```

The readme files for the example flow and Java classes that implement the SimpleNode interface provide the additional information and instructions on creating, building user specific custom Java classes, that are used as part of Java nodes in a flow definition and invoked during the flow execution.

The jar file or jar files housing the Java classes that implement the SimpleNode abstract class must be present under the following location for the Java nodes to invoke them, when the flow is being run:

```
<install_dir>/jars/com.hcl.hip.nodes.simplejava
```

User can either copy the jar file(s) manually to the above designated location for the flow to pick up during the execution or, create a zip file with the jar file(s) that hip-rest application will copy over the designated location during the startup.

The zip file should be in the following format:

```
<nodenode>_simplejava_<version>.zip
```

Where, *<nodenode>* is any string of user's choice and *<version>* is a version number of user's choice. *_simplejava_* is case-sensitive and should be in the middle of the *<nodenode>* and *<version>* when naming the zip file.

Properties

Properties are name-value pairs that indicate an action to perform, or any general-purpose information that user's Java class will understand and perform an activity that serves the purpose of the node in the flow definition and execution during the runtime.

Name

Specifies the name of the property.

Value

Specifies the value of the property.

Copying Java nodes zip file under the non-docker environment

This section provides the procedure to copy the user defined Java node class jar file(s) assembled in a zip file to the hip-rest that runs under non-docker environment.

1. Create a folder named modules under *<install_dir>*, if it is not available.
2. Copy the zip file *<nodenode>_simplejava_<version>.zip* in the *modules* folder.
3. Start the hip-rest application.

Copying Java nodes zip file under the docker environment

This section provides the procedure to copy the user defined Java node class jar file(s) assembled in a zip file to hip-rest that runs under the docker environment.

1. Create a folder *hipmodules* under */opt*, if it is not available.
2. Copy the zip file *<nodenode>_simplejava_<version>.zip* to */opt/hipmodules*.
3. Start the hip-rest container application.

Join

The Join node gathers the individual results and appends to a single output file or terminal.

- [Settings](#)

This section lists the settings available for the Join node.

Settings

This section lists the settings available for the Join node.

- [Write To File](#)
Writes the output to a file or the output terminal (The output terminal data content is the file name).
- [File Path](#)
Specifies the path to the output file.
- [Record Delimiter](#)
Specifies the character that delimits the records.
- [Output Header](#)
Specifies whether to write a header to the output.
- [Header Flow Variable](#)
The name of a flow variable that holds the header string.
- [Header](#)
Specifies the header to write to the output.
- [Header Delimiter](#)
Specifies the characters that delimit the header.

Write To File

Writes the output to a file or the output terminal (The output terminal data content is the file name).

To avoid high memory usage while processing the big data, you must use the Write To File property.

If the Write To File is set to true, the data created in individual batches is added to the file. If the Write To File is set to false, the data created in individual batches is added to the output terminal.

File Path

Specifies the path to the output file.

Record Delimiter

Specifies the character that delimits the records.

\r specifies for the carriage return.

\n specifies for the newline.

\t specifies for the tab.

The records and the header may have record delimiters by setting the Output and Header Delimiter properties. You must not set these properties, if the original data has the delimiter.

Output Header

Specifies whether to write a header to the output.

If the Output Header is set, the Join node creates the header.

If the Output Header is set to Header defined in flow variable, the Header Flow Variable property defines the flow variable that defines the header.

If the Output Header is set to Specified Header, the Header property defines the header.

Header Flow Variable

The name of a flow variable that holds the header string.

Header

Specifies the header to write to the output.

Header Delimiter

Specifies the characters that delimit the header.

\r specifies for the carriage return.

\n specifies for the newline.

\t specifies for the tab.

Log

This node logs the raw data from the node input into the file and propagates the data from the input to the output terminal.

Flow variable substitution is supported in the log file name. In order to generate unique log file name use the Flow internal variables _FLOWINSTANCE_ or _FLOWUUID_.

Flow variable FLOWLOGNODES changes the Log node behavior.

If the variable is set to false, the log file name will not be created. If the variable is set to true, the unique log will be created. The unique log fine name is [Nodename]_%_FLOWUUID_.log.

- [Log File Name](#)

Log File Name

Specifies the log file name. If the path is a relative path, it is relative to the flow home directory.

Passthrough

The Passthrough Node propagates data from the input to the output terminal.

There are no node properties. The node could be used as a placeholder for incomplete flow during the design phase.

REST Client

The REST Client flow node provides a simple and powerful way to access REST services. It can be used to directly invoke REST APIs or can use prebuilt configurations that define the APIs of the service. The node supports:

- Passing in request data on the input terminal or building the request directly in the node.
- Returning success and error messages to the different output terminals.
- The ability to loop through an array of requests to invoke an API repeatedly.

The node is an extension of the REST adapter. In many cases, it is much simpler to configure and use the node in a flow. Especially in cases, where an API call requires to send a request and handle the responses.

- [Functionality](#)

The REST Client node operates in 3 modes:

- [Terminals](#)

The REST Client node supports the terminals as given in below table.

- [Settings](#)

This section lists the settings available for the REST Client node.

- [Authentication](#)

Provides the options to authenticate the connection. This is an optional property.

- [Input Data Request Mode](#)

Input Data Request Mode is available, when you set Configuration Mode to Configuration Packages.

- [Request Assignments](#)

Provides a quick and easy way to build request data, when the request data is not provided to the node's input terminal.

- [Response Assignments](#)

Response Assignments are used to set flow variables based on values of elements of the response. This provides a simple way to parse responses to extract elements from the responses.

- [Logging](#)

Specifies the level of logging to use for the log (trace) file produced by the node.

- [Proxy server support](#)

This documentation describes the Proxy server support.

Functionality

The REST Client node operates in 3 modes:

- General: Specifies URL, method and other properties.
- Configuration Script: References a script (JSON file) that contains the endpoint definitions.
- Configuration Package: References a package (jar file) that contains the endpoint definitions, input/output formats and optionally Java plugin code.

Terminals

The REST Client node supports the terminals as given in below table.

Table 1. Terminals

Terminal	Direction	Description
Request data	In	The data sent as HTTP request.
Successful response	Out	The HTTP response for a successful request (HTTP status is 2xx).
Error response	Out	The HTTP response for an error condition (HTTP status is 4xx or 5xx).
Success	Out	Standard success terminal.
Error	Out	Standard error terminal. This terminal is invoked depends on whether the error response terminal is connected or not. If the error response is connected, the node fails, only if a processing error occurred. If the error response is not connected, this terminal is invoked, if either the node fails, or the HTTP call returns an error status.
Log	Out	The REST Client node log file is produced, if this terminal is connected.

Settings

This section lists the settings available for the REST Client node.

Authentication

Provides the options to authenticate the connection. This is an optional property.

Authentication options are as follows:

- None: The endpoint does not require any authentication.
- Basic: The endpoint requires username and password for authentication.
- OAuth 2.0: The endpoint requires an access token to access the REST APIs securely by using OAuth 2.0 Authentication. If the access token is expired, the action to be taken can be specified through Access Token Expiry Action.
- Use Endpoint Definition: When configuration script or packages are used in REST Client node, specifying this option will use the authentication specified in the endpoint configuration and any required connection properties need to be passed through Properties table.

Input Data Request Mode

Input Data Request Mode is available, when you set Configuration Mode to Configuration Packages.

It has three options as follows:

1. Single Request: The input data is read from the input terminal and treated as a single request.
2. Multiple Requests: It sends information to the node that the data read from the input terminal is a JSON array, and it should iterate through the array and call the API for each object in the array.
3. Template: If it is set to Template, the JSON or XML request template of the specified endpoint is used as the request. The contents of the Template can be dynamically modified by Request Assignments.

Request Assignments

Provides a quick and easy way to build request data, when the request data is not provided to the node's input terminal.

If the Input Data Request Mode is set to Template, this property is enabled. To use this option, the endpoint definition in the configuration file must include a template for the request which points to a JSON file defining the actual request data, or a template for the request which can be modified via request assignments. Assignments are specified through a table with two columns: Path and Value. The path is a JSON path to identify a field in the template. This uses dot-notation to specify the path. For example, if the template contains:

```
{ "address": { "city": "Springfield" } }
```

The city field would be identified by path address.city. The value specifies the value to substitute at the specified path, which is typically set to the value of a flow variable which is specified as %flow_var%.

Response Assignments

Response Assignments are used to set flow variables based on values of elements of the response. This provides a simple way to parse responses to extract elements from the responses.

This property is enabled only for Single Request or Template input data request mode. It is provided through a table with two columns: Path and Flow variable. The path is a JSON path of a field in the response, specified using the dot-notation defined above. The value of the field will be assigned to the specified flow variable.

For example, if response data contains:

```
{ "_id": "1111", "details": { "fname": "Steve", "lname": "Smith" } }
```

The response assignment table contains:

Path	Flow variable
_id	id
details.lname	last_name

This will result in flow variable "id" being assigned "1111" and "last_name" flow variable being assigned to "Smith".

Logging

Specifies the level of logging to use for the log (trace) file produced by the node.

The default is Off. The value Information means log informational, the value Errors Only means log error messages only, and the value Verbose means log debug and trace level messages along with the informational and error messages.

When Append Log is set to true, the log messages are appended to the file. When set to false, the file is truncated and the messages are written to the empty file. If the file path is not specified, the trace is logged to the default system logger.

Proxy server support

This documentation describes the Proxy server support.

- [Proxy server support in REST Client Node](#)

The Services and REST Client Node provide the option to configure a proxy server URL. A proxy server serves as an intermediary between the client and the server. It receives REST requests from IBM Sterling Transformation Extender and acts on behalf of IBM Sterling Transformation Extender when connecting to the end server. The Proxy server is utilized when access to the destination server is restricted, and connection must be made exclusively through a proxy server.

- [Troubleshooting of proxy server](#)

Proxy server support in REST Client Node

The Services and REST Client Node provide the option to configure a proxy server URL. A proxy server serves as an intermediary between the client and the server. It receives REST requests from IBM Sterling Transformation Extender and acts on behalf of IBM Sterling Transformation Extender when connecting to the end server. The Proxy server is utilized when access to the destination server is restricted, and connection must be made exclusively through a proxy server.

- [Proxy configuration using environment variables](#)

This documentation describes proxy configuration using environment variables.

- [Proxy configuration using flow variables](#)

This documentation describes proxy configuration using flow variables.

- [Proxy configuration using properties](#)

This documentation describes proxy configuration using properties.

- [Proxy configuration using the command line](#)

This documentation describes proxy configuration using the command line.

Proxy configuration using environment variables

This documentation describes proxy configuration using environment variables.

HTTP_PROXY

To establish connections to any HTTP URLs through the REST Client Node, you can set the HTTP_PROXY environment variable on the machine where IBM Sterling Transformation Extender is installed. This configuration allows the proxy server URL provided to be utilized for establishing connections.

Example: `HTTP_PROXY=http://proxy-server-ip:port`

HTTPS_PROXY

To establish connections to HTTPS URLs through the REST Client Node, you can configure the `HTTPS_PROXY` environment variable on the machine where IBM Sterling Transformation Extender is installed. By setting this variable, the provided proxy server URL will be utilized for making connections to any HTTPS URLs.

Example: `HTTPS_PROXY=http://proxy-server-ip:port`

NO_PROXY

To bypass the proxy server for specific URLs in the machine where IBM Sterling Transformation Extender is installed, you can set the `NO_PROXY` environment variable. This variable should be configured with a comma-separated list of URLs that do not need to be passed through the proxy server URL specified in `HTTP_PROXY` or `HTTPS_PROXY`.

Example: `NO_PROXY=localhost,some-domain.com`

Note: For adding these flow variables in the docker installation, one must edit `setenv.sh` file in hip-server and hip-rest containers.

Proxy configuration using flow variables

This documentation describes proxy configuration using flow variables.

proxy_url

When deploying and running flows, you have the option to configure the "`proxy_url`" flow variable for both HTTPS and HTTP URL connections using the REST Client Node.

Proxy configuration using properties

This documentation describes proxy configuration using properties.

Proxy URL

You can configure the Proxy URL property in Service Builder and REST Client Node.

Proxy configuration using the command line

This documentation describes proxy configuration using the command line.

-PURL

In the command line mode of the REST Client Node, you can utilize the "`-PURL`" command to configure the proxy server URL.

Example: `-PURL https://proxy-server-ip:3128`

Troubleshooting of proxy server

To verify the proxy configuration checks, follow these steps:

- Check if the Proxy server URL can be accessed from the machine where IBM Sterling Transformation Extender is installed. Use the Telnet command to access the proxy server.
For example: `telnet <proxy-server-ip> <proxy-server-port>`
- If you are unable to access your port, ensure that it is enabled if you are behind a firewall.
- If your proxy server IP is not accessible, make sure to include the DNS entry of the proxy server details in the hosts file where IBM Sterling Transformation Extender is installed.
- If the REST API requests are not being passed through the proxy server, ensure that there is an entry for the end resource URLs in your host's file on the proxy server.

Route

The Route node provides a way to route data conditionally to one or more outputs of the node. The node bases decisions by evaluating a condition for a flow variable and determining whether to send the data to output 1 or 2 or both, based on the result of the condition.

Route mode operates in two conditional modes:

- In Single Condition mode, data is sent to output 1, if the condition is set to true. Data is sent to output 2, if it is set to false.
- In Multiple Condition mode, output 1 and output 2 conditions are independently evaluated. If output 1 condition is set to true, data is sent to output 1. If output 2 condition is set to true, data is sent to output 2.

See the Route example to demonstrate the usage and behavior of Route node used in a flow. The Route node example contains a readme file that contains detailed instructions about how to run the example.

- **[Settings](#)**

This section lists the settings available in the Route node.

Settings

This section lists the settings available in the Route node.

Mode

Specifies the two modes.

- In Single Condition mode, data is sent to output 1, if the condition is set to true. Data is sent to output 2, if it is set to false.
- In Multiple Condition mode, output 1 and output 2 conditions are independently evaluated. If output 1 condition is set to true, data is sent to output 1. If output 2 condition is set to true, data is sent to output 2.

Output 1 Flow Variable

Specifies the flow variable on which to route for the output 1.

Output 1 Flow Operator

Specifies the operation to use in the comparison for output 1.

Output 1 Case Sensitive

Decides whether to perform the case sensitive or insensitive comparison.

Output 1 Value

Specifies the value to compare against. If the Operator is In List or Not In List, provides a comma-separated list of values.

Output 2 Flow Variable

Specifies the flow variable on which to route for output 2.

Output 2 Flow Operator

Specifies the operation to use in the comparison for output 2.

Output 2 Case Sensitive

Decides whether to perform the case sensitive or insensitive comparison.

Output 2 Value

Specifies the value to compare against. If the Operator is In List or Not In List, provides a comma-separated list of values.

Sleep

This node suspends the execution of the flow for the specified number of milliseconds

Time in milliseconds: Use this setting to provide the specified sleep time.

Flow variable FLOWSLEEPTIME overrides this value.

SOAP Client Node

The SOAP Client Node provides a simple and powerful way to access SOAP services.

The SOAP Client Node can be used to directly invoke SOAP APIs. It can also use prebuilt configurations that define the APIs of the service. The node supports:

- Passing in request data on the input terminal or building the request directly in the node
- Returning success and error messages to the different output terminals
- The ability to loop through an array of requests to invoke an API repeatedly

The SOAP Client Node is an extension of the SOAP adapter, it internally uses REST Adapter. In many cases, it is much simpler to configure and use the node in a flow, especially in cases where an API call requires to send a request and handle the responses.

The SOAP node and the SOAP adapter only supports Service Definition mode.

The SOAP node needs to:

- Handle SOAP faults and convert fault to endpoint failure message
- Handle SOAP faults and convert fault to endpoint failure message
- **Functionality**
The SOAP Client Node operates in service definition mode.
- **Terminals**
The SOAP Client Node supports a number of terminals.
- **Settings**
This section lists the settings available for the SOAP Client node.
- **Authentication**
Endpoint definition authentication is selected by default.
- **Logging**
Specifies the level of logging to use for the log (trace) file produced by the node.

Functionality

The SOAP Client Node operates in service definition mode.

Terminals

The SOAP Client Node supports a number of terminals.

The SOAP Client Node supported terminals are listed in the following table.

Terminals

Table 1. Terminals

Terminal	Direction	Description
Request data	In	The data sent as HTTP request.
Successful response	Out	The HTTP response for a successful request (HTTP status is 2xx).
Error response	Out	The HTTP response for an error condition (HTTP status is 4xx or 5xx).
Success	Out	Standard success terminal.
Error	Out	Standard error terminal. This terminal is invoked depends on whether the error response terminal is connected or not. If the error response is connected, the node fails, only if a processing error occurred. If the error response is not connected, this terminal is invoked, if either the node fails, or the HTTP call returns an error status.
Log	Out	The SOAP Client node log file is produced if this terminal is connected.

Settings

This section lists the settings available for the SOAP Client node.

Authentication

Endpoint definition authentication is selected by default.

Logging

Specifies the level of logging to use for the log (trace) file produced by the node.

The default is Off. The value Information means log informational, the value Errors Only means log error messages only, and the value Verbose means log debug and trace level messages along with the informational and error messages.

When Append Log is set to true, the log messages are appended to the file. When set to false, the file is truncated and the messages are written to the empty file. If the file path is not specified, the trace is logged to the default system logger.

Split

A Split Node should be used when there is a need to split CSV or JSON data processing or when a map can run in a burst mode to process data in batches. This might occur when data processing becomes excessively time consuming.

Such time-consuming behavior might occur when you send big data to the flow input terminal, or it might become memory intense when complex data validation is being performed. The Split Node can be used with these, and with other data processing tasks that can be done in parallel.

The Split Node splits data records into batches. Each batch of data is processed in parallel and passed for further processing by the downstream nodes. Using the Split node can improve the performance and scalability of your flow design.

The source of data that the split node consumes can be a map or data that the split node reads directly from the input terminal or a file.

If the map provides the data, the map must run in the burst mode. This is configured using the map input card settings. The action property 'fetch as' must be set to 'burst'. 'Fetch as' property can be set to 'burst' if the adapter supports burst mode. Example adapters that support burst mode are FILE, REST and all messaging adapters like Kafka, JMS, MQ. The fetch unit controls how many records to process with one batch. If the fetch unit is not set, the default value is 0 and it means fetch all records. The fetch unit must be set to split in order to split data into multiple batches

- [Configuring a Split node](#)
- [Split example](#)

See the Split example for a demonstration of how to design, deploy, and run flows that contain Splitnode.

Configuring a Split node

- [Consuming data through an input terminal](#)

A Split node receives data through an input terminal if it is connected to an upstream node. For example, from the map node, if its input terminal is enabled as a flow terminal.

- [Consuming data from a file](#)

A Split node reads csv data from the file specified by the File Path setting. This option is supported only if it is not connected to an upstream node, or if its input terminal is not enabled as the flow terminal.

- [Defining number of Split instances](#)

A Split node splits data into batches and passes each batch to its downstream node.

- [Settings](#)

This section lists the settings available for the Split node.

Consuming data through an input terminal

A Split node receives data through an input terminal if it is connected to an upstream node. For example, from the map node, if its input terminal is enabled as a flow terminal.

Consuming data from a file

A Split node reads csv data from the file specified by the File Path setting. This option is supported only if it is not connected to an upstream node, or if its input terminal is not enabled as the flow terminal.

The data file name specified by the File Path settings can be hardcoded as a literal, or it can be parameterized using a flow variable. Enclose the flow variable name with '%' characters, for example "File Path: %my_data_file%"

Defining number of Split instances

A Split node splits data into batches and passes each batch to its downstream node.

The produced batches are processed in parallel, with the maximum number of Split node instances defined with the Maximum Instances setting.

Settings

This section lists the settings available for the Split node.

Batch Size

Specifies the number of records to include in each batch.

Maximum Instances

Specifies the maximum number of parallel instances.

Read From File

Specifies the options to read the input from a file or from the input terminal.

File Path

Specifies the path of input file.

Source

Specifies the data source. Supported options are 'map' and 'data'.

If 'map' is a source, the supported settings are:

- **Map Name:** Defines a map that will run inside the split node in the burst mode
- **Input Card Name:** Defines which input card to use in the selected map to split data. (click on 'Fetch' to get the list of input cards). Default is the 1st input card
- **Output Card Name:** Defines which output card in the selected map will produce data for the split node output terminal. (click on 'Fetch' to get the list of output cards). Default is the last output card

If 'data' is a source, the supported settings depend on the data format. Supported data formats are delimited (CSV) and JSON.

Record Delimiter

Specifies the character that delimits records. \\r for carriage return, \\n for newline and \\t for tab.

Has Header

Specifies that the header record is available in data.

Include Header

Decides whether the header record should be included in each batch sent to the output.

Output Header Split

Controls the Split node header processing, if the Has Header property is set. If the Output Header Split is enabled, the Split node processes the header in the first batch. It sets Flow Internal variable ~SPLIT~ to value HEADER when processing the header. The Split node waits for the header to process before it executes rows processing in parallel. ~SPLIT~ variable is set to the batch index when rows are processed in parallel.

If the data format is 'JSON', the supported settings are:

- **Array Path:** The path of the array in the json document
- **Output JSON Object:** If this option is turned on, when the batch size is set to 1 output a JSON object, otherwise output an JSON array
- **Output Context Before:** Controls whether the output data includes the JSON context before the selected array

Split example

See the Split example for a demonstration of how to design, deploy, and run flows that contain Splitnode.

The Split node example contains a readme file that contains detailed instructions about how to run the example.

The example is located in the following directory:

install_dir/examples/flowengine/splitter_node

Where *install_dir* indicates the directory location of the base Transformation Extender product.

Note: To install this example, as well as the other flow examples, you must install Design Studio using the Custom installation.

Status Node

A Status Node can only be used as the first node in a "status flow", as that what makes a flow a status flow. Similar to an initialization flow, a status flow may be referenced from any other main flow. A status flow has a Status node as its first node which has Success, Error and Always output terminals that provide the flow status JSON to a downstream node.

The name of the status flow can be any valid flow name. If the project has a status flow, another flow can reference it from the Flow Settings > Status Flow Option.

A status flow cannot be referenced in a flow node (sub-flow) or an initialization flow, or another status flow.

If a flow has a status flow associated with it, when the main flow completes (either successfully or in error), the associated status flow is invoked. The status is output from this node to a subsequent node based on whether the error, status or always terminal is connected. Multiple terminals can be connected to handle both success and

failure statuses in the same status flow.

- **Functionality**
This documentation describes the functionality of the Status Node.
- **Terminals**
This documentation describes about the Status Node terminals.

Functionality

This documentation describes the functionality of the Status Node.

The Status Node can be used for the following actions, as an example, but can be linked to any other nodes that are appropriate for the required action needed:

- To send an email in the event of a flow failure
- To perform a cleanup operation
- To complete a transaction, such as sending a message

Terminals

This documentation describes about the Status Node terminals.

The following terminals are supported by Status node:

- **Success:** If the main flow is successful, The Status JSON is output from this terminal
- **Error:** If the main flow fails, the Status JSON is output from this terminal
- **Always:** Whether the main flow is successful or not, the Status JSON is output from this terminal

Note: The Status Node does not have any input terminals. The output for each terminal that is linked /executed from the Status node is the flow audit of the flow that referenced it.

Terminals

- **Error and Success terminals**

An error terminal is used on a node for cases where the error can be caught with another node, preventing the flow from failing. An error terminal will only run a node if the node that has the error terminal is the initial cause of the error.

- **Log terminals**

Log terminals, like Error and Success terminals, can be enabled on any node and are output terminals only.

Error and Success terminals

An error terminal is used on a node for cases where the error can be caught with another node, preventing the flow from failing. An error terminal will only run a node if the node that has the error terminal is the initial cause of the error.

A success terminal will run a node if the terminal that has the success terminal is successful. Multiple success terminals in a flow will run multiple nodes as long as each node is successful. A node is not considered successful until all nodes that are linked to it return back to that node successfully.

A node is considered successful if a linked node fails, but that linked node has an error terminal linked to catch the error.

See the Error and Success terminals example for more information.

See the Error and Success Terminals example for a demonstration of how to use Error and Success terminals in a flow. The Error and Success example contains a readme file that contains detailed instructions about how to run the example.

- **Error and Success terminals example**

The Error and Success terminals example demonstrates usage of Error and Success terminal used in a flow.

The Error and Success example contains a readme file that contains detailed instructions about how to run the example.

The example is located in the following directory:

install_dir/examples/flowengine/error_success_terminals

Where *install_dir* indicates the directory location of the base Transformation Extender product.

Note: To install this example, as well as the other flow examples, you must install Transformation Extender Design Studio using the Custom installation.

Log terminals

Log terminals, like Error and Success terminals, can be enabled on any node and are output terminals only.

If a node has a Log Terminal that is linked, it will run the node that is linked to it after it completes, before it returns to the node that it was called from.

The data that is sent out of a Log terminal depends on the type of node. For a Map node, it contains node information as well as the map's audit log in JSON format.

Flow terminals

Scheduled flows

Listener

A Flow that has a Map node as its first node, and uses a File adapter for an input, can enable that node's input to be a Watch.

When a package is deployed with a flow that has a Watch, the Flow Engine's File Listener is started to monitor for files that match the map's input file definition.

You can use wildcards in the file adapter's filename so the File Listener will start a flow instance with each file that matches the pattern defined for the filename.

After it is deployed, the flow will run an instance when a file that matches the configured pattern of the filename appears in the trigger directory. The trigger is the directory defined for the map's input.

If a wildcard is used in the definition of the filename, the system flow variable "_WILDCARD_" will be assigned the value of the wildcard for that flow instance. The value can then be retrieved throughout the flow instance. For a complete demonstration, see the Listener example.

See the Listener example for a demonstration of how to use the Listener. The Listener example contains a readme file that contains detailed instructions about how to run the example.

- [Listener example](#)
The Listener example provides a demonstration of usage and behavior of the Listener as used in a flow.
- [Amazon SQS Listener](#)
The Amazon SQS Listener polls the selected queue from Amazon SQS and retrieves the messages.

Listener example

The Listener example provides a demonstration of usage and behavior of the Listener as used in a flow.

The Listener example contains a readme file that contains detailed instructions about how to run the example.

The example is located in the following directory:

install_dir/examples/flowengine/listener

Where *install_dir* indicates the directory location of the base Transformation Extender product.

Note: To install this example, as well as the other flow examples, you must install Transformation Extender Design Studio using the Custom installation.

Amazon SQS Listener

The Amazon SQS Listener polls the selected queue from Amazon SQS and retrieves the messages.

User should be able to use SQS Adapter as a listener in Flow Engine for existing queue to which user has access to and the listener should be able to report SQS messages as and when it arrives to the SQS queue.

Amazon SQS Listener as a source

- To receive a message

Amazon SQS Listener as a target

- To publish a message
- [Listener Properties and Commands](#)
This chapter provides a detailed description of the listener commands and properties.
- [To continuously listen to a message from a queue](#)

Listener Properties and Commands

This chapter provides a detailed description of the listener commands and properties.

Access Key

Specifies the key of AWS user for the connection. This access key is used to sign programmatic requests to AWS API calls through the adapter. The corresponding adapter command is **-AK** (or **-ACCESSKEY**).

Secret Key

Specifies the key of AWS user for the connection. Like the username and password, both the access key ID and secret access key are mandatory to authenticate SQS queue for a Get or Put request. The corresponding adapter command is **-SK** (or **-SECRETKEY**).

Region Name

Specifies the name to connect to an AWS region for the IAM user account. The corresponding adapter command is **-R** (or **-REGION**).

Queue Name

Specifies the SQS queue name to or from which messages need to be accessed. During action configuration, SQS Adapter provides options to select queue name among available queues in the provided region. The corresponding adapter command is **-Q** (or **-QUEUE**).

Create Queue

If specified, this property indicates the SQS adapter to create a queue with the name passed in the 'queue_name' property, in case it doesn't already exist. This property is optional. When the create_queue property is not specified, adapter checks if the queue already exists; if not: it throws an error. The corresponding adapter command is **-CQ** (or **-CREATEQUEUE**).

Read Mode

Specifies whether you want to delete or keep a message after reading it. Default is Keep. There are two values:

- Keep: To keep a message after reading
- Delete: To delete a message after reading

The corresponding adapter command is **-RM** (or **-READMODE**).

Attributes

Specifies if the SQS message should be accessed along with the corresponding message attributes. When the attributes flag is set to true in output card, the passed data should be in predefined JSON format, else data should be in a plain text message. The corresponding adapter command is **-A** (or **-ATTRIBUTES**).

Limit

Specifies the total number of messages to consume. The limit property is applicable and utilized only in ITX deployments. Minimum value should be 1. If no limit is specified, adapter fetches only one message at a time from the queue. The corresponding adapter command is **-QTY**.

Timeout

Specifies the time duration (in seconds) to wait for before retrieving the message from the queue. The timeout property of SQS adapter is applicable and utilized only in ITX deployments. The SQS adapter would listen to the queue for specified seconds. The corresponding adapter command is **-LSN**.

Logging

Specifies the level of logging to be used for the log (trace) file produced by the adapter. The default is off. The value info means log informational and error messages, the value error means log error messages only, and the value verbose means log debug and trace level messages along with the informational and error messages.

Append Log

Specifies the flag that indicates the action to take when the specified log file already exists. When set to true, the log messages are appended to the file. When set to false: the file is truncated, and the messages are written to the empty file. The default value is true.

Log File Name

Specifies the name of the log file, where the log messages are written. If not specified, the default log file name m4sqsltrc is used, and the file is stored to the directory in which the executed compiled map resides.

To continuously listen to a message from a queue

To continuously listen a message from one queue and send it to another queue.

1. In the Flows workspace, select Source nodes type from the Palette and drag and drop it to the canvas.
2. Right click on the Source node and select Settings.
3. Select Use Command Line option or Use Existing Connection option if you want to reuse the already created connection. If you do not want to use existing connection, make Use Existing Connection option false.
4. Specify the Action properties. Select the source queue which will read the message.
5. Select Schema Select Type field as Plaintext.
6. Click OK.
7. Select Target nodes type from the Palette and drag and drop it to the canvas.
8. Right click on the Target node and select Settings.
9. Select Use Command Line option or Use Existing Connection option if you want to reuse the already created connection. If you do not want to use existing connection, make Use Existing Connection option false.
10. Specify the Action properties. Select the target queue which will send the message.
11. Select Schema Select Type field as Plaintext.
12. Click OK.
13. Connect nodes by dragging from the output terminal of one node to the input terminal of another node.
14. Right click on the Source node and select Enable Watch.
15. Click on the Save icon.
Flow definition gets saved.
16. Deploy the flow to the server.
17. You can now send and receive messages on Amazon SQS source queue and target queue respectively.
To disable this functionality, select Disable Watch in flows.

Once you push the message from the Amazon SQS source queue the flows will run in Design Server and the message will be sent to the Amazon SQS target queue. And you can continuously listen to the messages from one queue to another queue.

Flow audits

Flow audits are a way to retrieve more verbose information about a flow instance.

There are two ways to retrieve an audit:

- From install_dir/logs
- As a response from the REST API

Method 1 – Enable the audit setting from the flow settings dialog (from an open flow, click the gear icon in the upper right). If On Error is also enabled, the audit will only be produced if an error occurs while running the flow. A log file is created in the directory with the naming convention: flow name + instance uid.

Example: flow_e9bea8c6-2f8b-4c47-b776-dcd4f571190e.log

Using this method sets the default audit behavior for whenever the flow is run.

Method 2 – From the Swagger UI, specify “log” near the bottom (look for Add Item). This will again generate the file as above, but it will also return the audit in the response unless the flow is run asynchronously.

Using this method overrides the audit settings specified when designing the flow (method 1 above).

Flow variables

Flow Variables are process data variables that go along with the flow execution, accessible to all nodes in the flow while it is being executed under flow executor/engine context.

Flow Variables are name-value pairs, where name is a case-sensitive string that is associated with a specific value in string format. Even numbers as values are maintained in character string format.

Map nodes in a flow access and manipulate through a new set of functions included in the 'flowlib' dynamic function library. This example utilizes those functions to set, get and increment the value of the flow variable defined for the flow.

Flow Variables can be set initially before the flow instance starts, or by the nodes in the flow that are propagated to the following nodes in the flow until the flow execution comes to an end.

Each flow instance gets its own set of variables, no two flow instances being executed at the same time share their flow variables with others. Flow Variables exists until the flow completes after they are defined the first time under flow executor context. In a non-flow executor context like, for example, under Command Server, exist until the top map finishes execution. The variables are shared by the top map and all its RUN maps for that specific map instance execution.

Note: Flow variables are not supported on z/OS.

See the Flow variables example for a demonstration of how to use flow variables in a flow. The Flow variables example provides a readme file that contains detailed instructions on how to run the example.

- [Flow variables example](#)

See the Flow variables example for a demonstration of how to use flow variables in a flow.

Flow variables example

See the Flow variables example for a demonstration of how to use flow variables in a flow.

The Flow variables example contains a readme file that contains detailed instructions about how to run the example.

The example is located in the following directory:

`install_dir/examples/flowengine/flow_variables`

Where `install_dir` indicates the directory location of the base Transformation Extender product.

Note: To install this example, as well as the other flow examples, you must install Design Studio using the Custom installation.

Deployment

Introduction to deployment procedures, describing configuration variables, servers and server groups and packages.

Deployment overview

A *package* bundles a map and related files for deployment. *Deployment* sends a package to a server where a runtime engine (such as Command Server) runs the maps. An administrator sets up deployment environments by defining servers and, optionally, server groups, and chooses the maps and files to include in a package.

Deploying a package to a server automatically compiles the maps in the package, if necessary, for the server platform.

Deploying a package to a web server automatically registers map endpoints in the catalog.

A *configuration variable* is an alias that represents an environment-specific or sensitive (encrypted) value. Configuration variables resolve at run time. Maps and server definitions can use configuration variables to support multiple deployment environments.

- **Server groups**

A server definition can use configuration variables to represent the server's host name, user ID, and password. A *server group* links a server definition to a specific value defined for the configuration variable. When a map deploys to the server, the server group determines how the configuration variable resolves.

- **Server definition**

An administrator identifies the physical servers and virtual machines where maps can run.

- **Packages**

When the administrator adds a map to a deployment package, the package automatically includes any files that the map references, except files referenced in map rules. The administrator must manually add files referenced in map rules to the package.

Server groups

A server definition can use configuration variables to represent the server's host name, user ID, and password. A *server group* links a server definition to a specific value defined for the configuration variable. When a map deploys to the server, the server group determines how the configuration variable resolves.

For example, an administrator who wants to deploy maps to either a test server or a production server might set up deployment as follows:

1. Define two server groups:

- TEST_GROUP
- PROD_GROUP

2. Define a configuration variable called HOSTNAME with the following values for each server group:

Server Group	HOSTNAME Values
TEST_GROUP	TEST_HOST
PROD_GROUP	PROD_HOST

3. Define the servers, specifying each server's host name as the HOSTNAME configuration variable:

Server Name	Host Name	Server Group
TEST_SERVER	%HOSTNAME%	TEST_GROUP
PROD_SERVER	%HOSTNAME%	PROD_GROUP

- When the administrator deploys a map to the server named TEST_SERVER, the server host name resolves to the value TEST_HOST.
- When the administrator deploys a map to the server named PROD_SERVER, the server host name resolves to the value PROD_HOST.

Defining a server group is optional. A server group is not required for deployment.

Server definition

An administrator identifies the physical servers and virtual machines where maps can run.

A server definition has the following attributes:

Name and description

Identifies the server among other servers in a project.

Type

Web server

Deploys a map to a web server and uses the REST API to run the map. This server type requires the URL attribute.
Execution server
Represents all other map execution environments, such as the Transformation Extender programming interface or Command Server.
Platform
Specifies the operating system, such as Linux® x64.
Transfer type
Specifies the file transfer protocol as FTP or SFTP.
Host name or IP address
Identifies the server to deploy to.
Port number
The port of the FTP or SFTP server. If not specified, the port resolves to the default port for the selected protocol.
User name and password
Identifies the user ID and password for the selected FTP or SFTP protocol.
Target directory
Deploys files to this directory on the server. Map and file paths are relative to this directory.
HIP installation directory
The location where Transformation Extender is installed on the server.
Server group name
Adds this server to the specified server group. This attribute is optional.
URL
Identifies the URL of the catalog API where the deployment function registers a map. The URL is required when the server type is web server.

Packages

When the administrator adds a map to a deployment package, the package automatically includes any files that the map references, except files referenced in map rules. The administrator must manually add files referenced in map rules to the package.

A package defines a set of maps and flows to be collectively built and deployed to a server.

- [Package definitions](#)
A detailed definition of packages.
- [Deployment](#)
Deploying a package copies it to one or more Link runtime environments. Any maps within the package are compiled during the deployment process

Package definitions

A detailed definition of packages.

A package definition is comprised of the following:

- The project that defines the maps and flows.
- A base path under which the artifacts are deployed. This can be used to ensure that similarly named artifacts are kept distinct and do not overwrite each other.
- A set of flows and/or maps to be included in the package.
- A set of additional files from the project to be included in the package.
- A default flow to be executed using the flow command server if the '--flow' command line option is not used.

For maps and flows, a URL can be specified. This defines the endpoint URL through which the map or flow can be accessed on the IBM Sterling Transformation Extender runtime server.

Building a package causes the maps listed in the package to be compiled.

Deploying a package compiles all the maps listed in the package and add all of the artifacts to a package file which gets deployed to the specified server. Dependent objects are also included in the package even if they are not explicitly selected. For example, if a flow references other flows and maps, the referenced flows and maps are included in the package.

The flow command server (flowcmdserver) has a command-line option '--flow' that specifies the flow to execute from the package (sqlite file). If the '--flow' command-line option is not specified, the flow command server will execute the default flow if one has been specified.

Deployment

Deploying a package copies it to one or more Link runtime environments. Any maps within the package are compiled during the deployment process

The package will either be copied, sent by FTP, or by a REST API call, depending on the type of server being accessed.

When a package is deployed, any maps or flows that have a URL defined will be accessible using that URL. The Swagger page of the IBM Sterling Transformation Extender runtime is updated to include details of the endpoint.

Design Server UI

This documentation describes the Design Server User Interface.

Design workspace

The Design workspace includes the following components:

- [Schema Designer](#)
This documentation describes basics about the Schema Designer.
 - [Map Designer](#)
 - [Flow Designer UI](#)
This documentation describes the Flow Designer.
-

Schema Designer

This documentation describes basics about the Schema Designer.

- [Schema Designer dictionary](#)
 - [Schema Designer structure](#)
 - [Schema Designer properties](#)
 - [Schema Designer toolbar](#)
-

Schema Designer dictionary

Schema Designer structure

Schema Designer properties

Schema Designer toolbar

Map Designer

- [Map structure](#)
 - [Map Designer workspace](#)
 - [Build and run](#)
 - [Map Designe navigator](#)
 - [Map Designer toolbar](#)
-

Map structure

Map Designer workspace

Build and run

Map Designe navigator

Map Designer toolbar

Flow Designer UI

This documentation describes the Flow Designer.

The primary components of the user interface are:

- **Palette**
The Flow Designer Palette is the left-hand pane of the Flow Designer window.
 - **Canvas**
The Flow Designer canvas is the white display area immediately to the right of the Palette.
 - **Structure pane**
The Structure pane displays the terminals for the currently selected node and allows terminals to be added, removed and configured.
 - **Information pane**
The Information pane is in the lower right-hand corner of the Flow Designer window:
 - **Toolbar**
The Flow Designer toolbar allows you to perform various tasks.
-

Palette

The Flow Designer Palette is the left-hand pane of the Flow Designer window.

The Palette contains the list of available node types which can be dragged to the canvas. Available nodes are:

- Map
 - Flow
 - Parallelizer
 - Router
-

Canvas

The Flow Designer canvas is the white display area immediately to the right of the Palette.

This is the area that you use to construct flows by dragging and linking nodes.

If the diagram exceeds the size of the screen, the canvas can be scrolled by using horizontal and vertical scroll bars. The canvas can be enlarged by double-clicking on the gray bar to the right of the canvas, which hides the Structure and Information panes.

- **Nodes**
Hovering over a node on the canvas shows an expanded view of the node, revealing all of its terminals and connections.
 - **Context menu**
If you right-click on a node, the Context menu displays.
 - **Links**
If you right click on any of the links that connect the nodes you will be given the option to delete the link.
-

Nodes

Hovering over a node on the canvas shows an expanded view of the node, revealing all of its terminals and connections.

When you click on a node, the following occur:

- The structure diagram is updated to show the terminals of the node.
 - The Information Pane is updated with the name and description of the node.
-

Context menu

If you right-click on a node, the Context menu displays.

These options are available from the Context menu:

- Settings - edits the settings of the node. Each node type has different settings.
- Delete - prompts you to confirm that you want to delete the node. If you continue, the node is removed from the canvas, and any connected links are removed. If deleting the node leaves other nodes disconnected, an error indicator is shown on these nodes and they can be connected to existing nodes to complete the flow.

Other node types have additional options in the Context menu:

- Map node: Open Map – opens the map in the Map Designer workspace
 - Flow node: Open Flow – opens the flow in another tab in the Flow Designer
-

Links

If you right click on any of the links that connect the nodes you will be given the option to delete the link.

If you want to delete the link, you will be prompted to confirm the action before the link is removed from the canvas. If deleting the link leaves other nodes disconnected, an error indicator is shown on these nodes and they can be connected to existing nodes to complete the flow.

Structure pane

The Structure pane displays the terminals for the currently selected node and allows terminals to be added, removed and configured.

The Structure diagram includes three sub-components: menu, node icon, and terminal icons.

Click the menu icon at the top of the Structure pane menu to access these functions:

- Show/Hide Log Terminal – displays, or removes, a log terminal in the structure and the expanded node. The log node provides the log for the node's execution. For example, for a map node the log node produces a map audit log.
- Show/Hide Error Terminal – displays, or removes, an error terminal in the structure and the expanded node. The error terminal is invoked if the execution of the node fails. An error status message is sent to the error node.
- Show/Hide Success Terminal – this displays, or removes, a success terminal in the structure and the expanded node. The success terminal is invoked if the execution of the node succeeds. A successful status message is sent to the success node.

Information pane

The Information pane is in the lower right-hand corner of the Flow Designer window:

The information pane displays the following information:

Node Name – the name of the node, which must be unique in the flow.

Description – a multi-line description that explains the purpose of the node.

Map or Flow Name – if the node is a map or a flow, the name of the map or flow is displayed.

Toolbar

The Flow Designer toolbar allows you to perform various tasks.

Click the associated icon to do the following:

- Save -save the current flow.
- Save As - save the current flow under a different name and/or path.
- Snapshot - create a snapshot of the current state of the flow.
- Restore Snapshot - restore a snapshot to replace the current state of the flow with the state captured in the snapshot.
- Settings - show the flow settings in a dialog.
- Analyze - perform an analysis of the flow and display the results.

Design Server Properties

The Design Server properties includes:

- **Where Used**

Where Used allows a user to select the component where the subject component is used and go directly to it in the designer.

Where Used

Where Used allows a user to select the component where the subject component is used and go directly to it in the designer.

Where Used is applicable to the artifacts level, i.e, Schemas, Maps, Flows, Connections, and Actions.

Where Used can be viewed using following methods:

- In the Schemas/Maps/Flows/Connections/Actions workspace list, right click or select the stacked dot on the component to display the context menu and select Where Used from the menu. Where Used window opens.
- Click on the Where Used icon from the Schema/Flow designer toolbar. Where Used window opens.

Dashboard

This section describes a description of IBM Sterling Transformation Extender server dashboard.

The IBM Sterling Transformation Extender Server dashboard is a graphical user interface that provides users with a centralized and visual way to monitor and manage their server environment.

- [**Server Dashboard**](#)

- [Functionality of the Server Dashboard](#)
-

Server Dashboard

The server dashboard within IBM Sterling Transformation Extender serves as a centralized control panel for managing deployed flows and viewing the outcomes of flow runs. It offers the capability to execute maps and flows, specifically those without predefined inputs. However, for running other endpoints, users are directed to the respective Swagger page.

The server dashboard shares similarities with the Swagger page and the 'Run' dialog found in the flow designer. Moreover, the server page's user interface (UI) can double as the interface for the IBM Sterling Transformation Extender Commerce dashboard. In a Commerce context, the UI provides server status updates and run information, although functionalities for adding or editing servers are not applicable.

Functionality of the Server Dashboard

The server dashboard empowers users to perform a range of tasks, including:

1. Listing existing server definitions.
2. Creating new server definitions.
3. Editing server definitions.
4. Configuring values for server group-related configuration variables.
5. Viewing historical run statistics through graphical representations like pie and bar charts for user-specified time periods.
6. Displaying a comprehensive list of deployed packages/endpoints on the server. This includes features such as column-based sorting (e.g., by package or component name), package status indication (available or stopped), endpoint-specific statistics (successes, failures, total runs), and information about scheduling.
7. Enabling users to halt or initiate packages.
8. Scheduling endpoints, inclusive of complete schedule removal.
9. Triggering on-demand runs for endpoints.
10. Facilitating input file or data submission when endpoints require inputs.
11. Retrieving output data produced by endpoints.
12. Setting flow variables and query parameters within requests.
13. Providing both manual and automatic options for refreshing the server dashboard page. The auto-refresh toggle allows data updates at fixed intervals (for example, every 5 seconds).
14. Clicking on an endpoint discloses the following options:
 - Change schedule
 - Remove schedule (only shown if the endpoint is scheduled)
 - Run now

By default, the statistical data refreshes exclusively upon clicking the 'refresh' button. The UI also supports an 'auto refresh' toggle, ensuring that the UI autonomously updates data at predefined intervals.

Using Design Server

This documentation describes Design Server tasks.

Add a user

These procedures describe how to use the Design Server to add a new user.

To add a new Design Server user, login to Design Server as an Administrator, and then follow these steps in the Design Server Welcome page:

1. Click the drop-down arrow next to the Administrator tab in the upper right corner of the Design Server Welcome window.
 2. Select the Administration option from the menu. The Users window opens.
 3. Click the plus sign icon located to the right of the Search box at the top of the page to open Add User.
 4. Complete all of the fields in Add User area with the new user's information. Be aware of the following when completing the fields:
 - The password that you assign must be at least eight characters in length and contain at least one number and one character that is neither a number or a letter.
 - In most cases the Role field will remain in its default condition and will display the Designer option.
 5. Click the Add button at the bottom of the page. You are returned to the Users page. The user that you added appears in the list of users.
-

Create a project

Create a new project using Design Server.

To create a new project using Design Server follow these steps:

1. On the Welcome page, click the Go to your Workspace button.
The Projects tab opens.
2. Click the Create New Project button on the top of the Projects tab to create a new project.
3. Enter a name for the project in the Name field. You can also enter information in the Description and Tags fields if desired.
4. Click Ok. The new project appears in the Projects list.

5. You can edit, delete, or download the project by clicking the More options icon located to the right of the project name. The following are available options:

- Click Edit to return open the Edit Project page. Make any desired changes to the project name, description, or tags.
 - Click Delete to delete the project and remove it from the Project list.
 - Click the Export icon to export the project.
 - Click the Selective Export icon to open the Selective Export page. This page lets you select specific artifacts for export.
-

Import a project

Import a project using Design Server.

To import a project using Design Server follow these steps:

1. On the Welcome page, click the Go to your Workspace button.
The Projects tab opens.
2. Click on the Import Project button. The Import Project button is located before the Create New Project button on the top of the Projects tab.
The Import Project page opens.
3. In the Import Project page, click the browse link in the Project zip file area to navigate to the file that you want to import. Alternatively, you can drag and drop the file to be imported in to the Project zip file area.
4. Enter a project name in the Project Name field. You can also enter information in the Description field.
5. Click the Import Project at the bottom of the page to import the project.
The import progress bar will appear above the Import Project button. The message, 'Project Imported successfully', will appear upon completion of a successful import.
6. An existing project can be updated using the Import file option. Supported project files include:
 - Project files created using the **Export** or **Selective Export** options of an existing project.
 - A zip file that consists only of compiled maps (mmc files).
 - Individual compiled maps (mmc files).

Export a project

You can export existing projects in the form of a zip file.

To import a project using Design Server follow these steps:

1. On the Welcome page, click the Go to your Workspace button.
The Projects tab opens.
 2. Select the existing project to export. When you select the existing project, an Export icon gets visible before the Import Project button.
 3. Click the Export icon
The export progress bar will appear above the Export icon. The message, 'Project Downloaded', will appear upon completion of a successful export.
- **Selective Export**
Use the Selective Export feature to export the selected project artifacts.

Selective Export

Use the Selective Export feature to export the selected project artifacts.

The Selective Export function creates a .zip archive file and downloads it to the local file system. By using this function, you can export the artifacts in .mtt and .mms formats which are compatible with Design Studio.

To use Selective Export function:

1. Open an existing project. The existing project opens.
2. Select Selective Export from the drop-down menu next to project name. Selective Export window opens.
3. If you need to export artifacts with .json format, click Native radio button.
4. If you need to export artifacts with .mtt or .mms format, click Classic Files radio button.
Note: Schema, Maps, Flows, Files, and connections are disabled by default.

Create a flow

Use these steps to define a flow:

1. From the Welcome page, click the Create New Project icon.
2. Type a name for the project in the Name field. Optionally, enter information in the Description and Tags fields.
3. In the Design tab, select your project.
4. In the main Design Server workspace, choose Flows. Select New New Flow dialog opens.
5. Type a flow name in the Name field, and then click OK. The Flow Designer workspace for the flow opens.
6. From the Palette, on the left side of the page, drag the Map icon onto the canvas.
7. Right click on the map node to open a list of maps.

8. Select a map from the list.
 9. Click on the map node. The structure of the map is revealed in the structure panel, and details about the node are displayed in the details panel.
 10. Change the name of the node and give it a description by clicking on the Edit icon in the Details panel.
 11. Click on the drop-down menu in the structure panel and select Edit Terminals... to change which cards of the map are displayed as terminals in the map node.
 12. Select other node types from the Palette and drag them to the canvas.
 13. Connect nodes by dragging from the output terminal of one node to the input terminal of another node.
 14. Use the toolbar in the upper right corner of the canvas and select Save to save the flow.
 15. From the toolbar select Analyze, to analyze the flow for structural errors or incomplete definitions.
 16. Click the Home icon to return to the Welcome page. From here, the flow can be added to a package and deployed to the runtime environment.
-

Define mapping between source and target

This documentation describes how to use Design Server to define mapping between source and target.

This task introduces the concepts you will use to define mapping between a source and a target. Such mapping can be accomplished in multiple ways, however, the following requirements remain constant:

- For each input, a connection to the source resource must be defined and action properties must be defined, in order to determine how the data is read from the source.
 - For each output, a connection to a target resource must be defined and action properties must be defined, in order to determine how data is written to the target.
 - For each input and output, a schema must be specified to specify the format of the data.
1. On the Design Server Welcome page, create a new project, or select an existing project to open the main Design Server workspace.
 2. Upload a file containing input data by clicking on **Files > New**.
 3. Create a schema that describes the data in the input file by clicking on **Schemas > New**
 4. Click on **Connections** to create a source action to read from the input file.
 - a. Select the File connection, and then select **New Action** from the context menu.
 - b. In the Action Type section, select **Source** as the Direction, then click **Next**.
 - c. In the Properties section, select the file that you uploaded from the File Path drop down list, then click **Next**.
 - d. In the Schema section, select the file that you created and select the root type in the Schema Type drop down list, then click **Next**.
 - e. In the Identification section, name your action. Optionally, you can provide a description, search tags and a folder.
 5. Create a schema that defines the output data.
 6. Create a target action to write to an output file. Follow the same steps as for a source, but select **Target** as the Action Type.
 7. Create a new map by clicking **Maps > New**.
 8. Add the source action to the map by navigating to **Connection > Files <source_action>**, and then selecting **Add to Map** from the context menu.
 9. Add the target action to the map by navigating to **Connection > Files <target_action>**, and then selecting **Add to Map** from the context menu.
 10. To map from a source you must add the source to the mapping area. To do this, click on the context menu for the source in the Map Structure panel, and then click on **Add**. The input is added to the mapping area..
 11. Expand the schemas for input and output in the mapping area, and define your mapping by dragging types from the source fields to the output fields. More complex mappings can be defined by clicking on the output rules, or icons, to bring up the rule editor.
 12. Click **Save** on the toolbar to save the map.
 13. Click the **Build** icon to build the map. You will receive a success message if the build is successful. If it is successful, continue with the next step.
 14. Click **Run** to run the map.
 15. After running the map, the **Results** panel appears. Click on inputs or outputs to view the data, trace or audit for the specific input and output.
-

Running flows in the Designer

When you develop a flow in the Flow Designer you can run the flow directly from the Designer either in the Design Server, or on a different runtime environment.

If you want to run in a different runtime if, for example, if its environment has been configured with client software to access a particular resource by using a Link adapter.

- [**Preparing to run a flow from the Designer**](#)
Make these preparations in order to run your flow from the Designer.
 - [**Running the flow**](#)
After you specify the server, variables and input data, using the Run properties dialog, you can run the flow.
 - [**Viewing logs**](#)
The log for an individual node can be viewed by right-clicking on the node in the canvas and selecting View Log from the menu.
 - [**Viewing flow terminal data**](#)
The input and output data of the flow can be viewed by clicking on the node containing the flow terminal and then clicking on the input or output terminal icon in the structure.
 - [**Displaying link data**](#)
The data that was sent down a link in the flow can be viewed using either of these methods:
 - [**Including Maps and Files that are Dependencies of a Flow**](#)
-

Preparing to run a flow from the Designer

Make these preparations in order to run your flow from the Designer.

1. Save the flow.
2. Click the Run icon on the toolbar located in the upper right of your display. The Run Properties dialog opens. You can use this dialog to specify Server, Variables and Input Data.

- **Server section options**
The server drop-down in the Run Properties dialog allows you to specify these options:
 - **Variables section options**
The Variables section lists in table form the flow variables that are defined in the flow.
 - **Input Data section options**
If the flow has any input terminals, then the Input Data section provides a way to provide input to the flow terminal.
-

Server section options

The server drop-down in the Run Properties dialog allows you to specify these options:

- Run on Design Server – Set this toggle to either run the flow directly on the Design Server, or to deploy it to a runtime server.
 - Server – If you are deploying the flow to a runtime server, select the server from the drop down list. The servers in this list are defined in the Deployment screen. Only servers that are defined as Local or Web type will be listed.
-

Variables section options

The Variables section lists in table form the flow variables that are defined in the flow.

The name of each variable along with its value are listed.

The values of the variables can be specified in this table. Additional variables can be added, or rows can be deleted from the table. The flow will be run with the set of variables defined in this table.

Input Data section options

If the flow has any input terminals, then the Input Data section provides a way to provide input to the flow terminal.

The data can be provided from a file or pasted directly into the dialog. The options available are:

- Select File – A file uploaded into the project can be selected as input to the API.
 - Upload a File – A file can be selected from the local machine or dragged into the panel. Once added in this way the file will then appear in the project's file list.
 - Paste Data – Input data can be entered or pasted into a text box
-

Running the flow

After you specify the server, variables and input data, using the Run properties dialog, you can run the flow.

1. Click on the Run button to run the flow.

This action prepares and packages the flow and any flows or maps that are used within the flow. If running on a remote server, the package will be deployed to the server. The flow is then run in debug mode which allows the data passed between links in the flow to be saved so that it can be viewed.

2. After the flow run is complete, indicators display on each node of the flow to indicate whether the node succeeded, failed or was not invoked.
A green tick indicator means that the node ran successfully, a red cross indicator means that the node failed (or a downstream node failed) and no indicator means that the node did not run (because no data was provided to its input). Hovering over the node status icon shows a status message for the node run.
-

Viewing logs

The log for an individual node can be viewed by right-clicking on the node in the canvas and selecting View Log from the menu.

This displays the log in a panel in the UI that shows the node status and log messages from the node. If the node is a map, the map audit is shown.

Viewing flow terminal data

The input and output data of the flow can be viewed by clicking on the node containing the flow terminal and then clicking on the input or output terminal icon in the structure.

- To view the data that was sent to a flow, click on the node that contains the flow input terminal This is always the first node in the flow, then click on the input terminal in the structure.
- To view the data that the flow produced on an output flow terminal, click on the node that contains the flow output terminal, then click on the appropriate output terminal in the structure.

The input or output data is displayed in a new browser window.

Displaying link data

The data that was sent down a link in the flow can be viewed using either of these methods:

- Right-click on the link to display the context menu, and then select View Data from the menu.
- Click on a node in the canvas, right-click on a link terminal in the structure, and then select View Data from the menu.
The link data is displayed in a new browser window.

Including Maps and Files that are Dependencies of a Flow

When executing a flow from the design server, it is possible that the flow relies on internal dependencies. These dependencies need to be explicitly defined through specially formatted statements within the flow description.

In cases where a map node executes a map using the RUN function to call other maps, you must reference these additional maps in the flow description using the following format:

```
@packagemap=[map folder]/mapname
```

Furthermore, if a map node uses a map or run map that necessitates a specific file on its input card, you can incorporate the required file into the deployment process using the following notation:

```
@packagefile=<file path>
```

Both files and maps can be grouped into a package, and then you can reference this package using:

```
@package=<package name>
```

Note: It is important to observe that all "**@package*=" statements**, when added to the flow description, must each be on a new line. These measures ensure that all essential dependencies are correctly included and referenced when executing the flow from the design server.

Introduction to Design Studio

- [Design Studio overview](#)

The Design Studio provides the means for developing event-driven application-to-application (A2A) integration, business-to-business (B2B) integration, and consumer-to-business (C2B) application integration.

- [Design Studio configuration](#)

The Design Studio components are the design-time applications that are used to define, manage, and test maps and systems in a development environment.

- [Design Studio customization](#)

The Design Studio application toolbars and shortcut keys can be customized to fit your needs. Use the configuration options in each designer application to set preferences for your working environment.

- [Map Designer](#)

Map Designer is the application in the Design Studio that is used to specify data transformation logic in the form of map rules.

- [Database Interface Designer](#)

- [Integration Flow Designer](#)

The Integration Flow Designer is the graphical process-modeling element of the Design Studio. Use the Integration Flow Designer as a modeling component for defining and managing business processes.

- [Design Studio reference](#)

There are some restrictions that apply when using the Design Studio components to design your data transformation process. Certain words and symbols have a specific meaning when used within the Design Studio applications.

Design Studio overview

The Design Studio provides the means for developing event-driven application-to-application (A2A) integration, business-to-business (B2B) integration, and consumer-to-business (C2B) application integration.

The Design Studio is the design component of IBM Transformation Extender used to model and test your e-business solutions. Use the Design Studio to:

- Define your data.
- Create maps for your data.
- Manage the systems of maps that you create.
- Develop and test maps and systems in the Microsoft Windows environment.

The Design Studio is installed on and operates on Windows platforms. The client components of the Design Studio include the Type Designer, Map Designer, Integration Flow Designer, and Database Interface Designer.

Example files are located in the *install_dir/examples/dsgnstud* directory.

Design Studio configuration

The Design Studio components are the design-time applications that are used to define, manage, and test maps and systems in a development environment.

While the Design Studio components are primarily used during the design phase of your transformation project, the Launcher and Command Server are used to execute maps and systems in a runtime environment.

- [Design Studio design tasks](#)
There are tasks that you must complete when using the Design Studio.
- [Design Studio runtime tasks](#)
At runtime, the Design Studio overwrites map and card settings.

Design Studio design tasks

There are tasks that you must complete when using the Design Studio.

Design-time tasks include:

- Creating/generating type trees.
- Modifying type properties.
- Configuring the map settings, including SourceRule settings in input cards and TargetRule settings in output cards. (These configuration settings are compiled into maps.)

Systems can be designed using the Integration Flow Designer before, during, or after the type tree and map design phase.

These design-time tasks are performed on the client components of the Design Studio. Changing these settings at the time of execution are runtime tasks.

Use a platform-specific Launcher or Command Server to execute maps and systems in a runtime environment.

Design Studio runtime tasks

At runtime, the Design Studio overwrites map and card settings.

Map settings and card settings can be overwritten in two ways:

- Modifying the execution settings of a system component in the Integration Flow Designer.
- At execution time with execution commands.

Execution commands override the map settings or card settings of the compiled map when that map is run. See the Execution Commands documentation for a complete description of the override commands.

- [Execution settings](#)

Execution settings

Execution settings in the Integration Flow Designer are either Launcher or Command Server settings that are generated for the system. Command Server settings are used when the system is to run by means of the Command Server. Launcher settings are used when the system is to run by means of the Launcher.

Choose execution settings based on the server on which the system is to run. These execution settings overwrite the map and card settings, but do not alter the map source file (.mms) or compiled map file (.mmc).

Design Studio customization

The Design Studio application toolbars and shortcut keys can be customized to fit your needs. Use the configuration options in each designer application to set preferences for your working environment.

- [Customizing Design Studio toolbars](#)
In Design Studio, you can customize the standard toolbar or create your own. You can also delete and rearrange tools within a toolbar.
- [Customizing Design Studio shortcut keys](#)

Customizing Design Studio toolbars

In Design Studio, you can customize the standard toolbar or create your own. You can also delete and rearrange tools within a toolbar.

About this task

You can customize the standard toolbar or create your own. You can also delete and rearrange tools within a toolbar.

- **Delete a tool from a toolbar.** Press the **Alt** key and drag the tool off the toolbar.
- **Rearrange tools on a toolbar.** Hold down the **Alt** key and drag any tool, placing it in the toolbar where desired.
- **Add tools to a toolbar.** Select Toolbar(s) from the **View** menu. When the **Customize** dialog box appears, click the Commands tab, drag the tools to the toolbar as desired. You can also delete and rearrange tools; however, you do not need to use the **Alt** key in the **Customize** dialog box.
- **Make a floating toolbar.** Drag a toolbar and place it inside or outside the Type Designer window.

- **Cancel all changes made to a toolbar.** In the **Customize** dialog box, click **Reset**.
- **Customize the appearance of the tools.** You can select options for showing tool tips, viewing the tools as flat (**Cool Look** check box enabled) or raised, and specifying small or large tools on the toolbar (**Large Buttons** check box enabled).

To reset a toolbar to its default configuration

Procedure

1. From the **View** menu, choose **Toolbar(s)**.
2. From the **Toolbars** tab, select the toolbar you want to reset and click **Reset**.

To create a new toolbar

Procedure

1. From the **View** menu, choose **Toolbar(s)**.
2. On the **Customize** dialog box, click **New**.
3. Enter a name for the toolbar (for example, **Editing**) and click **OK**.
4. Select the new toolbar in the **Customize** dialog box.
5. The new toolbar appears on your screen.
6. Select the **Commands** tab.
7. Drag each tool that you want into the new toolbar.
8. Click **OK** to save the toolbar when you are finished.

Results

For example, you can drag the **Cut**, **Copy**, and **Paste** tools to the new "Editing" floating toolbar.

Arrange your toolbars by dragging them into positions.

To move a toolbar

Procedure

Drag the toolbar to the new location inside or outside the Type Designer window.

Results

When positioning a toolbar inside the Type Designer window and docking is not desired, press the **Ctrl** key to prevent docking.

Customizing Design Studio shortcut keys

About this task

To customizing the trace for IBM Sterling Transformation Extender Rest execution, complete the following steps:

Procedure

1. From the **Tools** menu, choose **Short Cuts**
The Shortcut Keys dialog box is displayed.
2. In the **Select a macro list** in the **Shortcut Keys** dialog box, select a command to which you want to assign a shortcut key.
3. Click **Create Shortcut**.
A message displays that indicates whether the shortcut is currently assigned.
4. After you finish assigning shortcuts, click **OK**.

Map Designer

Map Designer is the application in the Design Studio that is used to specify data transformation logic in the form of map rules.

The Map Designer uses the definition of data objects created in the Type Designer as inputs and outputs and provides the functionality to specify rules for the transformation and routing of data.

Maps are analyzed, compiled, and tested using the Map Designer. Each map defines input and output specifications. Map rules entered in the Map Designer operate on input data objects and build output data objects.

- [Map Designer overview](#)
To use the Map Designer, you must already have created or generated the type tree(s) that define your data. The Map Designer uses the data object definitions that are stored in those type trees.
- [Map Designer Eclipse overview](#)
In IBM Transformation Extender Design Studio release 8.2 and later, the Map Designer is an Eclipse-based program. The Eclipse platform provides a common user interface for the Map Designer called a *perspective*.

- [Map deployment overview](#)

You build and run maps using the Map Designer on a Windows platform. You can also use the Map Designer to build a map to be run on another platform.

Map Designer overview

To use the Map Designer, you must already have created or generated the type tree(s) that define your data. The Map Designer uses the data object definitions that are stored in those type trees.

After defining data objects and their properties in the Type Maker, define the map in the Map Designer to specify the source of the input and the target of the output. Sources and targets are specified with map cards. Each map can have none or more input cards, and one or more output cards.

The Map Designer is used to:

- Create maps to specify the logic necessary to transform the input data to the desired output data.
- Identify the source and data objects of the input data.
- Identify the target and data objects of the output data.
- Specify and build the output data according to the map rules.
- Provide information about data validation by generating trace files.
- View the run results of the map execution.

Map Designer Eclipse overview

In IBM Transformation Extender Design Studio release 8.2 and later, the Map Designer is an Eclipse-based program. The Eclipse platform provides a common user interface for the Map Designer called a *perspective*.

For information about using the Eclipse Workbench and about keyboard navigation, see the *Workbench User Guide* in the Eclipse Help.

Map deployment overview

You build and run maps using the Map Designer on a Windows platform. You can also use the Map Designer to build a map to be run on another platform.

Each map can be built for a specific platform, and then executed on that platform to perform the transformation of the data.

Database Interface Designer

The Database Interface Designer is used to import metadata about queries, tables, and stored procedures for data stored in relational databases. Use the Database Interface Designer to identify characteristics of those objects to meet mapping and execution requirements such as update keys and database triggers.

Use the Database Interface Designer to:

- Specify the databases to use for a source or target.
- Define query statements.
- Automatically generate type trees for queries or tables.
- **Database Interface Designer unique key combinations**
There are some unique key combinations available in the Database Interface Designer in addition to the standard Microsoft Windows navigation keys.
- **Database Interface Designer configuration**
The Database Interface Designer must be configured prior to use.
- **Database Interface Designer adapters**
The IBM Transformation Extender database adapters provide the connectivity to the RDBMS for retrieving input data and routing output data. Database adapters provide the option of using a driver to connect to the platform of your choice so you can automatically create type trees for database queries and tables.

Database Interface Designer unique key combinations

There are some unique key combinations available in the Database Interface Designer in addition to the standard Microsoft Windows navigation keys.

For enhanced accessibility, the following key combinations are available in the Database Interface Designer.

Navigation description	Key combination
To navigate to each pane of the Database Differences window	Shift+down arrow
To navigate to each tab in the Database Differences window	F8
To set the focus on the first item on the left tree view in the Database Differences window	Shift+F7
To select the first selection button in the Set Table Update Key Columns window	Ctrl+A
To select the second selection button in the Set Table Update Key Columns window	Ctrl+D
To select the third selection button in the Set Table Update Key Columns window	Ctrl+B
To select the fourth selection button in the Set Table Update Key Columns window	Ctrl+L
To select the fifth selection button in the Set Table Update Key Columns window	Ctrl+E

Database Interface Designer configuration

The Database Interface Designer must be configured prior to use.

After defining database query files and database table connectivity in the Database Interface Designer, define your map in the Map Designer. In map cards, you can specify an input source as either a query or a stored procedure and an output target as either a table or a stored procedure.

The Database Interface Designer is used to support business processes that read from and write to databases under control of a Relational Database Management System (RDBMS). The Database Interface Designer provides runtime database connectivity to the Transformation Server. This connectivity is supported with database-specific adapters or open database connectivity (ODBC). ODBC is a programming interface that enables programs to access data in database management systems that use structured query language (SQL) as a data access standard.

The Database Interface Designer:

- Provides connectivity to existing resources.
- Integrates multiple resources.
- Provides transactional control when resources support transactions.
- Combines transformation capabilities and resource connectivity.

All database access is transactional for resources that support transactions.

Database Interface Designer adapters

The IBM Transformation Extender database adapters provide the connectivity to the RDBMS for retrieving input data and routing output data. Database adapters provide the option of using a driver to connect to the platform of your choice so you can automatically create type trees for database queries and tables.

The Windows-based database adapters are installed with the Database Interface Designer as part of the Design Studio. You can also install the database adapters on additional systems to provide remote database connectivity.

Note: Non-Windows database adapters do not require the Database Interface Designer if you plan to only use **mtsmaker** without a database/query file (.mdq) to generate each type tree. For information about using **mtsmaker**, see the Resource Adapters documentation.

Integration Flow Designer

The Integration Flow Designer is the graphical process-modeling element of the Design Studio. Use the Integration Flow Designer as a modeling component for defining and managing business processes.

The Integration Flow Designer manages collections of maps at design time. Other product components, such as the Command Server or Launcher, manage the execution of maps at run time.

Collections of maps are called systems. Systems can be used as a focal point for interfacing to other development components and for preparing maps for execution.

Maps are created in the Map Designer. The systems created and managed in the Integration Flow Designer can be defined with existing maps, or they can be designed with maps that have not yet been created (pseudo map components). Systems can also be designed with subsystems.

Use the Integration Flow Designer to:

- Define interactions among maps and systems of maps.
- Specify integration workflow.
- Display data flow relationships among system components.
- Validate the logical consistency of workflows.
- Prepare systems to run.
- Verify component relationships.
- Generate process control information.
- Experiment at design time.
- Coordinate multiple data sources and targets.
- Coordinate events.
- Model a variety of what-if scenarios.
- Visualize the scenario results.
- Create and manage collections of logically related maps.
- Specify system deployment servers: Use one of three predefined server definitions or create server definitions that describe the servers in your enterprise.

The graphical representation of systems in the Integration Flow Designer helps you to visualize system designs, what-if scenarios, and system execution behavior. The interface provides easy management of system definition diagrams that allow concentration on the system design. Error reduction is achieved by automation of repetitive tasks and the use of system hierarchy.

- **[Integration Flow Designer unique key combinations](#)**

There are some unique key combinations available in the Integration Flow Designer in addition to the standard Microsoft Windows navigation keys.

- **[Integration Flow Designer components](#)**

The Integration Flow Designer enables you to overwrite settings compiled into maps.

Integration Flow Designer unique key combinations

There are some unique key combinations available in the Integration Flow Designer in addition to the standard Microsoft Windows navigation keys.

For enhanced accessibility, the following key combinations are available in the Integration Flow Designer.

Navigation description	Key combination
To move the focus to the Navigator	Shift+F7
To navigate to each pane of the System File Differences window	Shift+down arrow
To navigate to each tab in the System File Differences window	F8

Tip: In the More Find/Replace window, press Alt+i twice to move the focus to the Find All button.

Integration Flow Designer components

The Integration Flow Designer enables you to overwrite settings compiled into maps.

Map components of a system provide:

- Visual indication of audit log generation.
- Identification of the source or target type: message, database, file, or application.
- Ability to override map and card settings compiled into a map.
- A method of recording your system design with linked documents.

A map component is an Integration Flow Designer object that represents an executable map and displays its relationship with other components in the same system.

Systems can contain one or more map components. To keep the system manageable, you may create smaller systems and then include them as a subsystem of a larger system. Map components and system components are executable units.

Execution settings specify how the map or subsystem executes. The execution settings for system components are maintained in the Integration Flow Designer and are eventually used on designated servers to execute the system components. Each component in a system has separate execution settings. The referenced files (maps or subsystems) are static and are not modified directly with this information.

A map component that references an executable map has a default set of sources and targets defined within the referenced map. Map settings and card settings for sources and targets are defined in the Map Designer, and may be overwritten in the Integration Flow Designer.

The sources and targets defined in the map are the interface points of the map component used within a system. The sources and targets of executable map components are displayed in the main window of the Integration Flow Designer.

In the Integration Flow Designer, the expanded view of a map component distinguishes its sources from its targets.

Design Studio reference

There are some restrictions that apply when using the Design Studio components to design your data transformation process. Certain words and symbols have a specific meaning when used within the Design Studio applications.

- [Design Studio quotation mark usage](#)
Design Studio allows you to use quotation marks.
- [Design Studio reserved words and symbols](#)
There are some words and symbols that are restricted in Design Studio.
- [Design Studio hex, decimal, and symbol values](#)
Design Studio allows you to use various hex, decimal, and symbol values.

Design Studio quotation mark usage

Design Studio allows you to use quotation marks.

Although there are platform-specific limitations regarding the usage of double and single quotation marks, when adding a text string into any map rule, component rule, or command line, use the following characters for quotation marks.

Symbol	ASCII Character	Text Reference in Documentation
'	039	Single quotation mark
"	034	Double quotation mark
''	039 and 039	Two single quotation marks (for some non-Windows environments)

Using any other characters can produce unexpected results.

Design Studio reserved words and symbols

There are some words and symbols that are restricted in Design Studio.

Reserved words and symbols have a special meaning and may be used only in the manner described in this documentation. The following is the list of reserved words and symbols, and their intended use.

Reserved Word or Symbol

Intended Use	
\$	Shorthand representation of the object name in a component rule or a map rule.
:	Used to separate components.
::	Shorthand representation of a unique component or partition path. For example, A::B refers to the unique path from the type A to the type B .
<>	Separates a partition from its partitioned type
,	Used to separate arguments of functions and maps
""	Used to enclose literal text strings.
()	Used for a component range, to enclose function or map arguments, and to block sub-expressions in a rule
@	Separates a comment from a component.
[]	Denotes an indexed member of a series.
{}	Used to enclose a list of literals.
Names of Functions	
	Names of functions are reserved words. Names of functions can be used as type names, but cannot be used as map names. For a complete list of functions, see the Functions and Expressions documentation.
Operators	
	Operators are reserved. For a complete list of operators, see the Functions and Expressions documentation.
ANY	Represents one or more types in a component name. ANY cannot be used in component rules or map rules.
COMPONENT	Refers to any component preceding the current one in the same component list. Used in the Type Designer to refer to preceding components of the same component list. This reserved word can only follow the reserved word IN. For example:
	COUNT (Record IN COMPONENT)
FALSE	Represents the logical false.
IN	Separates a component from an object that contains it. For example, LineItem IN File refers to all occurrences of the type LineItem in the type File .
LAST	Special index value that refers to the previous data object of a particular series.
NONE	Stands for the null value or non-existence of data content. Can be used in a component rule to test the presence of data. Can be used in a map rule to generate no occurrences of an output.
<space>	Separates type names. (This is an actual space.) The names of individual types in component names are separated by spaces. Component names themselves are separated by colons. For example, C:A B is the C component of the A subtype of B ; C:A B:X is the C component of the A B component of type X ; and so forth.
<symbol>	Symbol abbreviations for entering non-printable characters. For example, <CR>.
TRUE	Represents the logical true.
WHERE	Can be used in the LOOKUP or EXTRACT function in place of the comma (,) between arguments.

Design Studio hex, decimal, and symbol values

Design Studio allows you to use various hex, decimal, and symbol values.

The following table lists characters, hexadecimal values, decimal values, and symbols.

In this list, some characters in the character name are capitalized to indicate the common abbreviation for the character. For example, the <STX> abbreviation represents the **S**tart of **T**e**x**t character.

The hexadecimal values are provided as a reference. For any literal value specified in the Type Designer Properties window, hexadecimal values must be enclosed in double angle brackets. For example, <<0A>> represents the hexadecimal value for a line feed character.

The decimal values are provided as a reference. Decimal values can be used with the SYMBOL function. For any literal value specified in the Type Designer Properties window, decimal values must be enclosed in single angle brackets. For example, <10> represents the decimal value for a line feed character.

Symbols can be inserted into map rules and component rules. You can insert them from the Symbols dialog box or by typing the hexadecimal or decimal value.

Character	Hex Value	Decimal Value	Symbol
NewLine	-	-	<NL>
WhiteSpace	-	-	<WSP>
KanjiSpace (WideSpace)	-	-	<KSP>
0-9	-	-	<DIGIT/>
a-z and A-Z	-	-	<LETTER/>
A-Z	-	-	<LETTERUPPER/>
a-z	-	-	<LETTERLOWER/>
NULL	00	0	<NULL>
StartOfHeading	01	1	<SOH>
StartofTeXt	02	2	<STX>
EndofTeXt	03	3	<ETX>
EndOfTrans.	04	4	<EOT>
ENQuiry	05	5	<ENQ>
ACKnowlege	06	6	<ACK>
BELL	07	7	<BELL>
BackSpace	08	8	<BS>
HorizTab	09	9	<HT>
LineFeed	0A	10	<LF>
VerticalTab	0B	11	<VT>
FormFeed	0C	12	<FF>
CarriageReturn	0D	13	<CR>
ShiftOut	0E	14	<SO>
ShiftIn	0F	15	<SI>
DataLinkEscape	10	16	<DLE>
DeviceControl1	11	17	<DC1>
DeviceControl2	12	18	<DC2>
DeviceControl3	13	19	<DC3>
DeviceControl4	14	20	<DC4>
NegativeAck	15	21	<NAK>
SYNchron.Idle	16	22	<SYNI>
EndTransBlock	17	23	<ETB>
CANcel	18	24	<CAN>
EndofMedium	19	25	
SUBstitute	1A	26	<SUB>
ESCAPE	1B	27	<ESC>
FileSeparator	1C	28	<FS>
GroupSeparator	1D	29	<GS>
RecordSep.	1E	30	<RS>
UnitSeparator	1F	31	<US>
SPace	20	32	<SP>
!	21	33	-
"	22	34	-
#	23	35	-
\$	24	36	-
%	25	37	-
&	26	38	-
'	27	39	-
(28	40	-
)	29	41	-
*	2A	42	-
+	2B	43	-
,	2C	44	-
-	2D	45	-
.	2E	46	-
/	2F	47	-
0	30	48	-
1	31	49	-
2	32	50	-
3	33	51	-
4	34	52	-
5	35	53	-
6	36	54	-
7	37	55	-
8	38	56	-
9	39	57	-
:	3A	58	-

Character	Hex Value	Decimal Value	Symbol
;	3B	59	-
<	3C	60	-
=	3D	61	-
>	3E	62	-
?	3F	63	-
@	40	64	-
A	41	65	-
B	42	66	-
C	43	67	-
D	44	68	-
E	45	69	-
F	46	70	-
G	47	71	-
H	48	72	-
I	49	73	-
J	4A	74	-
K	4B	75	-
L	4C	76	-
M	4D	77	-
N	4E	78	-
O	4F	79	-
P	50	80	-
Q	51	81	-
R	52	82	-
S	53	83	-
T	54	84	-
U	55	85	-
V	56	86	-
W	57	87	-
X	58	88	-
Y	59	89	-
Z	5A	90	-
[5B	91	-
\	5C	92	-
]	5D	93	-
^	5E	94	-
_	5F	95	-
.	60	96	-
a	61	97	-
b	62	98	-
c	63	99	-
d	64	100	-
e	65	101	-
f	66	102	-
g	67	103	-
h	68	104	-
i	69	105	-
J	6A	106	-
k	6B	107	-
l	6C	108	-
m	6D	109	-
n	6E	110	-
o	6F	111	-
p	70	112	-
q	71	113	-
r	72	114	-
s	73	115	-
t	74	116	-
u	75	117	-
v	76	118	-
w	77	119	-
x	78	120	-
y	79	121	-
z	7A	122	-
{	7B	123	-
	7C	124	-
}	7D	125	-

Character	Hex Value	Decimal Value	Symbol
~	7E	126	-
DElete	7F	127	-

Type Designer

Getting started

The IBM Transformation Extender Design Studio is an Eclipse-based design environment that integrates Type Designer and Map Designer functionality.

The Eclipse platform provides a common user interface for the Type Designer and Map Designer called a "perspective". First-time Eclipse users must review the Workbench User Guide before using the Eclipse-based Design Studio.

The *Workbench User Guide* is located in the help.

1. Open the Design Studio.
 2. Select Help > Help Contents. The help opens for the Eclipse Platform.
 3. Under Contents, select Workbench User Guide.
- [Starting the Type Designer](#)

Starting the Type Designer

About this task

On Windows operating systems, you can start the Type Designer from the Programs menu.

Procedure

1. Select Transformation Extender > Design Studio.
 2. If you are prompted to choose a metadata directory, you can either accept the default or enter a different directory. The metadata directory is a designated workspace where your type tree files will be stored after importing them.
The first time you open the Type Designer, the Eclipse SDK Welcome page is displayed.
 3. From the Welcome page, choose Workbench (*Go to the workbench*).
The Workbench opens into the Transformation Extender Development perspective.
You must be in the Transformation Extender Development perspective before you can create a type tree.
- [Opening an existing type tree](#)
Use this procedure to open an existing type tree.

Related tasks

- [Opening or changing to the Transformation Extender Development perspective](#)

Opening an existing type tree

Use this procedure to open an existing type tree.

About this task

To open an existing type tree, either double-click the type tree .mtt file in any directory listing, or select the type tree .mtt file from the Extender Navigator view in the Design Server. Select the option under Transformation Extender preferences to change the default behavior to always open the Transformation Extender Development perspective when you double-click an existing type tree (.mtt) file outside the Design Server.

Procedure

To open an existing type tree:

1. Determine if the type tree file is in your current project.
If so, make sure the Extender Navigator view (workspace) is open and complete steps **a** and **b**.
 - a. Expand the project's subtree to show the desired file.
 - b. Double-click the file.
The type tree opens in an editor window.

When the type tree file is not in the current project, it must be imported.
2. Click File > Import.
The Import window opens.

3. Expand General.
4. Click File System and click Next.
5. Click Browse to locate the file directory.
6. Choose the directory that contains the type tree and click OK.
The source and destination directories and files appear pictorially in the Import window.
Tip: To select only specific file extensions for import, use the Filter Types option.
After you make your import selections, you must browse for a folder into which to import the type tree.
7. At the Into folder field, click Browse.
The Import into Folder window opens. This window displays the folders in the current project.
8. Select a folder and click OK.
9. Under Options, make additional selections if needed.
 - Overwrite existing resources without warning.
 - Create a complete folder structure.
 - Create selected folders only.
10. Click Finish.
The files you selected are imported into the project.

Related reference

- [Customizing the Transformation Extender Development perspective](#)
-

Introduction to the Type Designer

Use the Type Designer to define, modify, and view type trees. A type tree describes the syntax, structure, and semantics of your data. The syntax of data refers to its format including tags, delimiters, terminators, and other characters that separate or identify sections of data. The structure of data refers to its composition including repeating substructures and nested groupings. The semantics of data refer to the meaning of the data including rules for data values, relationships among parts of a large data object, and error detection and recovery.

- [About type trees](#)
 - [Subtypes](#)
 - [Type tree hierarchy](#)
-

About type trees

A type tree (.mtt) defines the entire contents of at least one input that you intend to map or one output you intend to map. A type tree is the mechanism for defining each element of your data. Similar to a data dictionary, a type tree contains a collection of type definitions.

Because data definitions are defined in a type tree, you should be familiar with the specifications that define your data before attempting to create one.

A data file is a simple example. The file is made up of records and each record is made up of fields. In this case, there are three kinds of data objects: a file, a record, and a field.

In a file of records, think of the data in terms of the three data objects. For example, one type defines the entire file; another type defines the entire record contained in that file. Other types in the same type tree define the data fields of the record.

A type tree has three datatype classifications: group, category, and item. The tree has a root type and other types are connected to the tree through branches of the tree. The root type is the base type from which all other types stem, representing the data objects of all types in the tree. "Root" is the default name when creating a new type tree, however, you can modify any type name.

When viewing a type tree from the Extender Navigator, you can see the different kinds of data objects, but you cannot see the layout or composition of the data. For example, by looking at the tree, you cannot tell that a record consists of fields.

Types within a type tree are listed in alphabetical order by default. To change the order of appearance for new types, modify the root type properties.

IBM Transformation Extender supports large metadata. It supports more than 65,536 types and 65,536 components in a single type tree. To reduce the number of types that maps that use large type trees must process, trim these type trees so that they contain only the required types.

Subtypes

Subtypes are created for a number of reasons like identifying distinct data object. Another reason is that specific types of an object may have different properties such as different date formats.

Think of subtypes as different "flavors". Ice cream flavors can be chocolate, strawberry, or vanilla. To create a type tree to represent ice cream as data, the different flavors of ice cream could be subtypes of the type **IceCream**.

The type **IceCream** is generic and describes any kind of ice cream. The type **Chocolate**, a subtype of **IceCream**, represents a certain kind of ice cream: chocolate.

A file of purchase order data may have two different kinds of records: header and detail. The type tree representing this data might have a **Record** type with **Detail** and **Header** as subtypes of **Record**.

Type tree hierarchy

The types in a tree are arranged in a hierarchy. If a type is subordinate to another type, it is called a *subtype*. The type on the branch stemming above a specific type is called its *supertype*.

Subtypes have more specific properties. For example, two different kinds of fields, **Name** and **Date**, may be defined as subtypes of **Field**.

The item type **Field** describes any field. The item types **Date** and **Name** are more specific kinds of fields. In a classification hierarchy such as this, the deeper the subtype is in the type tree, the more specific the data characteristics.

Example scenario

Imagine that a type tree represents your house. There is a type for the entire house (**House**). A type for each room (**Bathroom**, **Bedroom**, and so on), and types for the different furnishings in these rooms (**Bed**, **Chair**, and so on). Each type represents a complete object. A house is made up of rooms. Inside these rooms are beds, chairs, couches, and so on. You know this, but you cannot see it by looking at the classification hierarchy view of the type tree. You must open one of the types in the group view to see its components.

Type Designer basics

- [Type Designer files](#)
Type trees have the .mtt file name extension.
- [Type Designer icons](#)
This topic displays the icons used in the Type Designer and provides a description of each of them.
- [Opening or changing to the Transformation Extender Development perspective](#)
This topic describes how you open or change to the Transformation Extender Development perspective.
- [Customizing the Transformation Extender Development perspective](#)
Use the Transformation Extender preferences to configure the default settings for the Transformation Extender Development perspective.
- [Customizing the Type Designer environment](#)
Use the Type Designer preferences to configure the default type tree settings for the Transformation Extender Development perspective.
- [Search options](#)
The Design Studio supports standard search options.
- [Bookmarks and Tasks](#)
Use bookmarks and tasks to annotate type trees and map source files in the Design Server.

Type Designer files

Type trees have the .mtt file name extension.

If the **Backup on save** option is enabled in the Type Designer preferences, the backup type trees are automatically created when the type tree is saved. The backup type trees have the .bak file name extension. The backup type trees are automatically saved in the same directory as the type trees.

When a type tree is analyzed, a type tree analysis message file is automatically created in the same directory as the type tree. The Type Designer has the following file name extensions:

.mtt	Type tree file
.dbe	Type tree analysis message file
.bak	Backup type tree file

Related concepts

- [Customizing the Type Designer environment](#)

Type Designer icons

This topic displays the icons used in the Type Designer and provides a description of each of them.

Icon	Description
◆	Item type
◆	Partitioned item type
⌚	Sequence group type
⌚	Choice group type
ParallelGroup	Partitioned group
ParallelGroup	Unordered group
ParallelGroup	Category type

Opening or changing to the Transformation Extender Development perspective

This topic describes how you open or change to the Transformation Extender Development perspective.

Procedure

To open or change to the Transformation Extender Development perspective:

1. Select Window > Open Perspective > Other.
2. Select Transformation Extender Development.

The Transformation Extender Development environment opens.

The default environment (perspective) consists of the following views:

- Extender Navigator view
- Outline view
- Composition view
- Properties view

What to do next

Tip: You can also initiate this task using the Open Perspective button: 

This perspective is saved upon closing the application. The next time you open the Type Designer, you will arrive at this point—the Transformation Extender Development perspective.

The tasks described in the Type Designer documentation assume that you are beginning from the Transformation Extender Development perspective.

Customizing the Transformation Extender Development perspective

Use the Transformation Extender preferences to configure the default settings for the Transformation Extender Development perspective.

To access user preferences, click Window > Preferences > Transformation Extender.

Customize the Transformation Extender Development perspective by making your selections for the following options. After you save the selections, you must reopen the map for the Design Server to use the changed default settings.

Always open Transformation Extender perspective when opening a mtt or mms file.

Every time you open a type tree (.mtt) or map source (.mms) file, the Transformation Extender Development perspective opens.

In the Transformation Extender perspective, always save the type tree after analysis.

Every time you analyze a type tree in the Transformation Extender Development perspective, the Design Server saves it.

Show Schemas as Native Icons

Design Server displays the native icons for schema types.

Suppress Build Warnings

If warnings occur during the map build process, the process ignores the warnings and compiles the map.

In the Content Assist pane, select the components on which to enable content assistance, as well as the option to enable automatic content completion. When you enable types, maps, or functions, Design Server provides you with hover help in the Rule editor, which includes information about the components that you selected. While you are entering a map rule in the Rule editor, you can view the hover help for a type so that you can access information about those selected components in one place.

Enable Types

Enable content assistance on types.

When you hover your mouse pointer over a type in a map rule in the Rule editor, the Design Server displays the hover help, which contains information including the name of the type, the type tree (.mtt) value that is in the card, and the location of the type tree file.

Enable Maps

Enable content assistance on maps.

When you hover your mouse pointer over a map in a map rule in the Rule editor, the Design Server displays the hover help, which contains information including the name of the map, the input and output that the map in the rule uses, and details about the input and output cards.

Enable Functions

Enable content assistance on functions.

When you hover your mouse pointer over a function in a map rule in the Rule editor, the Design Server displays the hover help, which contains information including the definition and syntax of the function.

Enable AutoComplete

Enable automatic content completion.

When you are entering a map rule in the Rule editor, and you type any one of the following symbols: a single quotation mark, a double quotation mark, an opening parenthesis, or an opening bracket, Design Server automatically completes the open symbol with its corresponding closing symbol.

To display settings and dialogs that you previously selected to hide, click Clear, which is next to Clear all 'do not show again' settings and show all hidden dialogs again in the WTX Dialogs pane.

Customizing the Type Designer environment

Use the Type Designer preferences to configure the default type tree settings for the Transformation Extender Development perspective.

The Type Designer preferences include customization options for saving, comparing, entering text, and setting the default root name.

To access user preferences:

1. Select Window > Preferences > Transformation Extender > Type Tree.

Auto-Save

Specifies the time interval in minutes for automatically saving open type trees. (The default setting is 0 minutes.)

Backup on save

Creates a backup copy of the type tree using file extension .bak when a type tree file is saved. (This option is enabled by default.)

Commit changes with

Method for committing changes, such as when entering restrictions in component rules.

For example, when Enter is the chosen method, you would press Enter to enter restriction text in a component rule. As a result, the text is entered in the rule cell and the next component cell is selected.

- Enter & Tab

This is the default setting. Either Enter or Tab can be used to commit the changes.

- Enter

Use the Enter key to commit changes.

Tip: You can create a hard return in any component rule or edit area by pressing Ctrl+Enter.

- Tab

Use the Tab key to commit changes.

Tip: When Tab is the chosen method, you can create a hard return in any rule or edit area by pressing Tab.

Rule differences

- Ignore white spaces and tabs

Enable this option to ignore white spaces and tabs when comparing component rules.

Default Root

A default type name for the root type when creating new type trees in the Type Designer. (The default name is Root.)

Type tree differences

Specifies node colors when comparing type trees.

- New Node Color (The default setting is blue.)
- Different Node Color (The default setting is red.)

- [Confirmation message preferences](#)

Using the Confirmations preferences, you can enable or disable the appearance of confirmation messages. Confirmation messages are typically displayed before completing an action such as a deletion.

- [Group view preferences](#)

Use the Group View preferences to customize the appearance of the group view.

- [Font preferences](#)

Use the font preferences to change the font types that are used in the Type Designer editors and views.

Confirmation message preferences

Using the Confirmations preferences, you can enable or disable the appearance of confirmation messages. Confirmation messages are typically displayed before completing an action such as a deletion.

Confirmation messages can be displayed for the following actions:

Propagating attributes

A confirmation message is displayed prior to propagating attributes.

Changing type class

A confirmation message is displayed prior to changing the class of any type.

Deleting type(s)

A confirmation message is displayed prior to deleting a type.

Adding new type

A confirmation message is displayed prior to adding a type.

Moving type(s)

A confirmation message is displayed prior to moving a type.

Reordering type(s)

A confirmation message is displayed prior to reordering a type.

Changing Group subclass

A confirmation message is displayed prior to changing a group subclass.

Disable the option if you want the action to take place without confirmation.

Group view preferences

Use the Group View preferences to customize the appearance of the group view.

- **Show range:** When enabled, the range assigned to each component is displayed next to the component name.
- **Show component number:** When enabled, a sequential number is displayed preceding the component name.

- **Use ellipses:** Use this option to control the appearance of object names in the group view.
 - When this option is enabled, the abbreviated short object name displays in the component rule.
 - When this option is disabled, the full name of the type is displayed in the component rule.

Rule Color Specifications

You can choose specific colors for the following elements pertaining to component rules:

- Syntax
- Number
- Type Name
- Map Name
- Reserved
- Literal
- Comment
- Background

Font preferences

Use the font preferences to change the font types that are used in the Type Designer editors and views.

You can change the font type for the following areas:

- Item view
- Type tree editor
- Group view
- Analysis view
- Category view
- Properties view

Search options

The Design Studio supports standard search options.

Search and replace

- Use the search operation to search through a single map source file or type tree file, or multiple map source files or type tree files, within search scope parameters that you specify, such as workspace, working set, and so forth.
- Use the replace operation to first do the search, and then replace text in the search results.

Find and replace

- Use the find operation to search for matches on text within a single map source file or type tree file that you have open in the editor.
 - Use the replace operation to replace text in the matched find results. This option is only available for searches within a type tree file.
 - You can also do these operations using search and replace.
- **[Search and replace](#)**
This topic describes the features of search and replace.
 - **[Find and replace](#)**
This topic describes the features of find and replace.

Search and replace

This topic describes the features of search and replace.

You have the option to do a search, which does a search operation only, or a replace, which does both a search operation and a replace operation. The search option searches the entire contents of a single map source file or multiple map source files, or a single type tree file or multiple type tree files based on the criteria you selected. The replace option does the search, and replaces the values in the search string with a replacement value for the search result components that you select to be included in the replace operation.

Features of the search and replace options:

- Run through the Transformation Extender Search tab in the Search window navigated from the Search menu on the menu bar.
- Operates on an entire map source or type tree file or files. You can set the scope of the **Search** operation to do the search within the workspace, a specific working set, or a specific directory. For map source files, you can also set the scope to do the search in a current Map editor.
- Within a map source file or files, searches on:
 - All type tree names
 - All card names
 - All output objects
 - All map names
 - All card type names
 - All map rules
 - All reference files

For example, you can search through map cards within a map source file, and replace file names ending with .txt with .dat. You can replace all card types named Record Set Data with Record Data. You can find card names that include the word Test.

- Within a type tree file or files, searches on:
 - Type properties
 - Component rules
 - Restrictions
 - Type names

For example, you can search through all type trees in your workspace, and replace those type names named month with mm.

- **Search For**

Search For: is used to set the entity on which search and replace operates.

- **Limit To**

Limit To: is used to set the limitations within what is being searched by the search and replace operations.

- **Scope**

Scope: is used to set the range within which search and replace operates.

- **Using search and replace**

There are different criteria that you can set when you use search and replace in the Design Studio.

Search For

Search For: is used to set the entity on which search and replace operates.

The following list describes the search entity options you can choose:

- Property
 - Searches type properties
- Components
 - Searches components
- Restrictions
 - Searches various symbols.
- Type Name
 - Searches type names

Related tasks

- [Using search and replace](#)
-

Limit To

Limit To: is used to set the limitations within what is being searched by the search and replace operations.

The following list describes the search limitation options you can choose:

- Property
 - For type properties, you can select one of the following search limitation options:
 - Release
 - Implied default value
 - Special value(None)
 - Delimiter
 - Max Bytes
 - Special value(zero)
 - Empty
 - Min Bytes
 - Initiator
 - This is the default setting.
 - Date Time
 - Max Chars
 - Terminator
 - Pad
 - Min Chars

- Components

For components, there is one search limitation:

- Components

- Restrictions

For restrictions, there is one search limitation:

- Value

- Type Name

For type names, there is one search limitation:

- Name

Related tasks

- [Using search and replace](#)
-

Scope

Scope: is used to set the range within which search and replace operates.

The following list describes the search scope options you can choose:

Workspace

Searches for the search entity you selected, within the entire Eclipse workspace, which is where your projects are stored on your local drive

Working set:

Searches for the search entity you selected, within an entire Eclipse work set, which you create containing a subset of resources in the workspace

Directory:

Searches for the search entity you selected, within an entire folder

Current editor:

Option is only available when you select Map Source Files under Search In:

Searches for the search entity you selected, within the map source file that is open in the Map editor

Selection options:

- All Maps (default)
- Current Map
- Current and Reference Maps

Related tasks

- [Using search and replace](#)
-

Using search and replace

There are different criteria that you can set when you use search and replace in the Design Studio.

About this task

To use search and replace:

Procedure

1. Click **Search > Search** on the menu bar, or press **Ctrl+H**.
The Search window opens.
2. Click the Transformation Extender Search tab.
This is the view through which you make your search and replace selections.
3. Enter text or wildcard characters in the Search string field.
Case sensitive and Whole word are optional fields that you can select.
4. Under **Search In:**, click **Type Tree Files**.
5. Under **Search For:**, click one of the search entities.
6. Under **Limit To:**, click one of the search limitation options.
There are several search limitation options when you select **Property** under **Search For:**. There is only one search limitation option for each of the other **Search For:** options.
7. Under **Scope:**, click one of the scope entities.
8. Click **Search** to do a search operation only, or **Replace** to do both a search and a replace operation.
9. If you clicked **Replace**, the Type Designer window opens and displays the following message: Changes made using replace can not be undone. Do you want to continue with replace? Respond to the message by clicking **OK**.
10. While the operation is running, it displays the Search Query window containing the **Operation in progress** message. If it is searching through a large number of resources, the window remains open. Click **Run in Background** so that you can continue to access the Design Server graphical user interface (GUI).
You can monitor the operation's progress by opening the Progress view, and viewing the progress indicator. You can also cancel the operation in this view.
The operation completes and displays any matches in the lower pane under the **Search** tab.
11. Click a type tree file listed in the search results to see the details of the matches.

Double-click any type tree name in the list to locate it in the Type Tree editor.

12. If you clicked **Replace** to do both a search and a replace operation, do the following additional steps:

- a. Select the check box next to each type tree name in the search results details to include in the replace operation.
There are two options in the Search view that you can use if there are many search results to include (Select All), or exclude (Deselect All) from the replace operation.
 - Select All - automatically selects all of the type trees
 - Deselect All - automatically deselects all of the type trees, which were previously selected
- b. Enter the replacement text in the **With:** field.
The Replace: field contains the text you entered in the Search string field of the Search window.
- c. Click **Replace**.

The operation replaces the text in the Replace: field with the text in the With: field in every search result you selected to be included in the operation. It cannot replace text if you selected Output under the Search For: pane. When the operation completes, the Type Designer window displays a message about how many replacements were made.

d. Click OK.

Related reference

- [Search For](#)
 - [Scope](#)
 - [Limit To](#)
-

Find and replace

This topic describes the features of find and replace.

You have the option to do a find, which does a find or a find next operation only, a replace, which does a replace operation, a replace/find, which does both a replace operation and a find next operation, and a replace all, which does a replace operation for all found text. The available options are dependent on the objects you are searching.

Features of the find and replace options:

- Run through the Find/Replace window navigated from the Edit menu on the menu bar, or from the Find/Replace option on the context menu of an object.
- Operates on selected objects in a map source or type tree file.
- Within a map source file, searches on:
 - Object names in a map card
 - Map rules in a card
 - Map elements in the Organizer folder, which is in the Outline view

Only the Find option is available for searches within a map source file.

- Within a type tree file, searches on:
 - Text characters associated with types
 - Text characters associated with components
 - Text characters associated with restrictions

All options are available for searches on types within a type tree file.

Only the Find and the ReplaceAll options are available for searches on components within a type tree file.

- [Using the find and replace operation](#)

There are different criteria that you can set when you use find and replace in the Design Studio.

Using the find and replace operation

There are different criteria that you can set when you use find and replace in the Design Studio.

About this task

To use find and replace:

Procedure

1. Select the area in which you want to search. For example, select the card to be searched.
2. Click **Edit > Find/Replace**.
3. In the Find field, enter the text you want to find.
4. To customize your search, select additional options in the window pertaining to direction, scope, case-sensitivity, whole words, and so forth.
5. Click Find to begin the search.
When search criteria is found, it is highlighted within the area that you selected in Step 1.
6. At this point, you can select from the following options:
 - To skip this instance and find the next, click Find.
 - To cancel the search, click Close.

Bookmarks and Tasks

Use bookmarks and tasks to annotate type trees and map source files in the Design Server.

A bookmark is a persistent marker that is visible in the Bookmarks view. A task is a persistent marker that is visible in the Tasks view. With a bookmark, you can keep track of an element of a map or type tree and return to that element quickly and easily during development. With a task, you can track the progress of your development work in type trees and maps.

- [Adding a bookmark](#)
- [Removing a bookmark](#)
- [Adding a task](#)
- [Removing a task](#)

Adding a bookmark

About this task

Use Add Bookmark to add a bookmark to an element in a map or type tree.

To add a bookmark to an element:

Procedure

1. Right-click an element in a map or type tree and click Add Bookmark.
The Add Bookmark window opens with a default value in the Enter Bookmark description field.
2. To change the default value, enter a value in the Enter Bookmark description field.
3. Click OK.

Results

The bookmark is added to the element. You can view the bookmark in the Bookmarks view. Click Window > Show view > Other, and in the Show View window, click Bookmarks under General.

Removing a bookmark

About this task

Use Remove Bookmark to remove a bookmark from an element in a map or type tree.

To remove a bookmark from an element:

Procedure

Right-click the element with the bookmark and click Remove Bookmark.

Results

The bookmark is removed.

Adding a task

About this task

Use Add Task to add a task to an element in a map or type tree.

To add a task to an element:

Procedure

1. Right-click an element in a map or type tree and click Add Task.
The Properties window opens with default values in the fields.
2. To change the default value that is in the Description field, enter a value.
3. To change the task priority, select a priority from the Priority list.
4. If you completed the task, select the Completed check box.
5. Click OK.

Results

The task is added to the element. You can view the task in the Tasks view. Click Window > Show view > Other, and in the Show View window, click Tasks under General.

Removing a task

About this task

Use Remove Task to remove a task from an element in a map or type tree.

To remove a task from an element:

Procedure

Right-click the element with the task and click Remove Task.

Results

The task is removed.

Working with type trees

- [Creating type trees](#)
- [Viewing type trees side by side](#)
- [Viewing subtypes](#)
- [Selecting subtypes](#)
- [To change the placement order of subtypes](#)
- [Defining data](#)
- [To add a component to the target tree](#)
- [To view a specific component/item restriction](#)
- [Type tree differences](#)
- [Exporting a type tree](#)

Creating type trees

The following list outlines the process for creating type trees:

- Identify the data objects in your data and define each piece of data that you intend to map.
- Create types for each data object in your data.
- Define the properties of each type.
- Create component lists.
- Define component rules, if needed.
- Define item restrictions, if needed.
- Analyze and save the tree.
- [To create a type tree](#)
- [To add types to the type tree](#)

To create a type tree

Before you begin

Remember: You must have a project created before you can create a new type tree.

About this task

Important: Be familiar with the specifications that define your data before creating a type tree.

Procedure

1. Select **File > New > Type Tree** when "type tree" is present in the shortcut menu. Otherwise, select **File > New > Other > Transformation Extender > Type Tree**.
2. Click Next.
3. In the first field, accept the default project folder or enter a different project folder name.
4. In the File name field, enter a file name and extension for the type tree. For example, *tree_1.mtt*.
5. Click Finish.
The new type tree is displayed under the project folder in the Extender Navigator. In the main window, a tab has been created for the new type tree from where you can begin adding types that define your data to the "Root" type.

To add types to the type tree

About this task

Types are always created as a subtype of the currently selected type. A new type tree has a single root type. All types are added as subtypes of the root type and then as subtypes of other types.

Procedure

1. Select the type under which you want to add a type. For example, select "Root".
 2. Choose one of the following methods to add a type:
 - Press Insert.
 - Right-click and choose Add from the context menu.The program will prompt for confirmation unless that setting has been disabled in the user preferences.
3. If prompted, click Yes to confirm that you want to add a type.
 4. Enter a name for the type.
 5. Open the properties for the type (Type > Properties).
 6. Begin defining the type properties, such as **Name**, **Class**, **Description**, and so forth.
 7. After defining the properties, save changes.

What to do next

Continue adding types that define your data.

Viewing type trees side by side

About this task

To view two type trees side by side:

Procedure

1. Open both type trees for viewing.
2. Click on the title bar of one tree.
3. Hold down the mouse button and drag down and to the left, until the gray outline of a box and a black arrow appear.
4. Release the mouse button.

Results

The two trees are visible side by side.

Viewing subtypes

About this task

To view all subtypes:

Procedure

From the type tree editor, right-click on the highest level of the type you want to expand and select Expand All Subtypes from the context menu. (For example, select the root type if you want to expand the entire tree.)
All subtypes of the selected type are displayed.

What to do next

To collapse all subtypes:

1. From the type tree editor, right-click on the highest level of the type that has subtypes to collapse and select Collapse All Subtypes from the context menu. The subtypes of the selected type are collapsed.

Selecting subtypes

About this task

To select all subtypes:

Procedure

From the type tree editor, right-click on the parent level of the type that has child types you want to select and choose Select All Subtypes from the context menu. The child subtypes of the selected parent type are selected. The grandchildren, or, types that are more than one level down from the parent, are not included in the selection.

To change the placement order of subtypes

About this task

By default, when types are added to a type tree, they are placed in alphabetical or *ascending* order. In addition to ascending order, you can change the default placement order of new types to be descending, first in the list, or last in the list. To change the placement order of types for a type tree, modify the properties of the root type.

Procedure

1. Open the properties of the root type.
 2. Go to the Order Subtypes property and make a selection from the drop-down list: Ascending, Descending, Add First, or Add Last.
 3. Save changes.
-

Defining data

For optimum data transformation, the entire contents of your data must be defined. Using the Type Designer, you can define input data so that each data object of the source data is identified. You can define the output data according to your output specifications.

How specifically you define the data is up to you. For example, if there is a large section of data that you want to process quickly, define it loosely as a chunk of text.

The Type Designer does not define data content as source data or target data. The purpose of the Type Designer is to define your data. Use the Map Designer to define input and output definitions for the data.

- [Objects, types, and classes](#)
 - [To define a literal or variable value](#)
-

Objects, types, and classes

Type trees consist of objects, types, and classes.

Objects

A data object is a complete unit that exists in your input or is built on output. A data object can be simple (such as a date) or complex (such as a purchase order).

A data object is some portion of data in a data stream that can be recognized as belonging to a specific type. When you create types, identify all of the objects that make up the data: input objects and output objects.

Types

A type defines a set of data objects that have the same characteristics. For example, the type "Date" can be defined as representing data objects in the form MM-DD-YY. The type "CustomerRecord" can be defined as representing data objects, each of which consists of a **Company**, **Address**, and **Phone** data object.

Classes

A type is classified according to whether or not it consists of other objects. Each type in a type tree must be defined in one of three classes: item, group, or category.

- **Item type**
An item type represents a simple data object that does not consist of other objects.
 - **Group type**
A group type represents a complex data object that consists of other objects. For example, "FullName" is a group type that contains the components: FirstName, MiddleInitial, and LastName.
 - **Category type**
A category type is used for inheritance and for organizing other types in a type tree. For example, you might have a category named "OrderField" to organize the different kinds of order fields in your type tree.
-

To define a literal or variable value

Procedure

1. Open the Properties for a group or category.
 2. Expand the options under Group Subclass.
 3. Expand the options under Format.
 4. In the drop-down menu next to Component Syntax, select Delimited.
 5. Expand the options under Component Syntax.
 6. In the drop-down menu next to Delimiter, select Literal or Variable.
-

To add a component to the target tree

Procedure

1. In the type tree editor, double-click the type.
The Component view opens. The Component view consists of an edit area and two columns: Component and Rule.
 2. From the type tree editor, select a component and drag it into a cell in the Component column.
The component has been added.
-

To view a specific component/item restriction

About this task

Procedure

From the type tree editor, double-click the item.
The item view opens and the item name is displayed on the tab.

Results

Restrictions are displayed in the cells (rows) of the Include column.

Related tasks

- [Using search and replace](#)
-

Type tree differences

Before you begin

The type tree differences feature allows you to compare two type tree files.

Two type trees are considered different when:

- Any of the types are different.
- Any of the types that exist in one type tree does not exist in the second type tree.
- When the order of the subtypes within a subtype is different.

A type is different when:

- Any of the properties are different.
- Any of the components are different.
- Any of the restrictions are different.

A restriction is different when:

- The restriction value is different.
- The description is different.

About this task

To compare type trees:

Procedure

1. In the Extender Navigator view or the Navigator view, press Ctrl and select two type tree (.mtt) files.
You can compare type trees with different names.
2. Right-click one of the selected type tree files, and click Compare With... > Compare Type Trees.
IBM Integration Platform opens the type trees to do the compare operation.
If the type tree files are large and do not open immediately, the Comparing Type Trees... window opens, and displays the Operation in progress... message.
The progress of the comparison continues to display until the operation completes.
3. Click Run in Background so that you can continue working while the Compare Type Trees operation runs.
You can monitor the Compare Type Trees operation by clicking the Progress tab, and viewing the progress indicator.

Results

After the Compare Type Trees operation completes, a window containing the results opens. You can resize and position the window to view all differences.

Exporting a type tree

About this task

You can export a type tree or a portion of a type tree to a type tree script file. A type tree script file (.mts) is a document file containing a script of commands in XML format. An .mts file can be used as input to the Type Tree Maker to create a type tree or portion of a type tree. See the Type Tree Maker documentation for more information.

To export a type tree or portion of a type tree:

Procedure

1. Select the type you want to export. (Select the root if you want to export the entire type tree.)
2. Right-click and choose Export from the context menu.
The Export type tree to file window opens.
3. Select a directory and enter a name for the new .mts file. (The default name is the name of the selected type followed by an .mts extension.)
4. Click Save.

Results

The subtree of the selected type is exported. If you select a type other than the root, it is assumed that you are adding to an existing tree and the document file will have a <OPENTREE> command. If you select the root type, it is assumed that you are generating a new tree and the document file has a <NEWTREE> command.

An example of a type tree of an export script and an example map that converts an exported restriction list to a cross-reference table is included with the Design Studio examples (*install_dir\examples\dsgnstud\ttmaker*).

Managing types

- [Selecting subtypes](#)
- [Drag-and-drop operations in the Type Designer](#)
- [Moving and copying objects](#)
- [Reordering objects](#)
- [Merging types](#)
- [Renaming types](#)
- [Generating a type definition report](#)
- [Testing part of a type tree](#)

You can test a type tree by generating test data for an item or a group. Type Designer creates source and compiled maps that generate and validate the test data. The test-data-generation map automatically runs and produces a text file with the sample data. After that, the test-data-validation map automatically runs on the sample data file.

Selecting subtypes

About this task

To select all subtypes:

Procedure

From the type tree editor, right-click on the parent level of the type that has child types you want to select and choose Select All Subtypes from the context menu. The child subtypes of the selected parent type are selected. The grandchildren, or, types that are more than one level down from the parent, are not included in the selection.

Drag-and-drop operations in the Type Designer

About this task

The following list is a summary of available drag-and-drop operations:

- To move an object, drag it to the destination.
- To copy an object, press **Ctrl** and drag it to the destination.
- To merge a type, press **Shift** and drag the type to the destination tree.
- To add a component name to a component rule, press **Alt** and drag the component to the component rule bar.
- To specify a type as a comment type or a variable delimiter, initiator, terminator, or release character in the Properties view, press **Alt** and drag the type from the tree to the Properties view.

Related concepts

- [Components must be in the same type tree](#)

Moving and copying objects

About this task

You can move and copy the following objects within the Type Designer by using the drag-and-drop method:

- Types (within the same tree)
When you drag a type from one tree to another, it is copied to the other tree because it is copied from one type tree file to another.
- Components (from one view to another)
- Restrictions (from one view to another)
Duplicate restrictions are not permitted.

When you move or copy a type using the drag-and-drop method, the entire subtree of the type is also moved or copied. The properties, components, component rules, and restrictions are also copied.

If a type cannot be a subtype of the type you are dragging it to, you will not be able to drag it. For example, you cannot move an item under a group.

Note: You cannot place a type under another type if it cannot be a valid subtype.

- [Moving a type](#)
- [Copying a type](#)
- [Type names](#)

Moving a type

About this task

To move a type, you can use the drag-and-drop method or use the following procedure.

To move a type:

Procedure

1. In the type tree editor, right-click the type you want to move and select Move from the context menu.
The Move dialog opens and the type name is present in the From field.
2. While the dialog is still open, return to the type tree editor and select the "destination" type. The destination type name appears in the To field of the Move dialog
3. Click Move.

Copying a type

About this task

In addition to using the drag-and-drop method, you can use the context menu to copy a type.

To copy a type:

Procedure

1. In the type tree editor, right-click on the type and select Copy from the context menu.
The Copy Type dialog opens.
2. Go back to the type tree editor and select the target type—the type under which you want to copy the given type.
The target type name appears in the To field of the Copy Type dialog.
3. In the As field, accept the default original name of the type or modify it if needed.
4. The Copy Sub-tree option is enabled by default. Disable the option if you do not want the subtree of the type copied to the target type.
5. Click Copy.
The type is copied to the target type that you selected.
6. Click Close to close the dialog.

Type names

When you move or copy a type, if that type is referenced by another type (for example, if it is used as a component), the reference is updated to reflect the new relative type name. In addition, if a type you move or copy references other types, these references are automatically updated. Referenced type names include component names, syntax item names, and comment type names.

When you move a syntax item that is referenced as a number separator, initiator, terminator, delimiter, or release character, its referenced name changes.

When you move a type that references other types, the references are automatically updated.

Reordering objects

You can reorder the following objects within the Type Designer by using the drag-and-drop method:

- Subtypes - to reorder subtypes, press **Ctrl + Shift** while dragging a subtype to the desired location. Subtypes may be reordered only when the type whose subtypes you are reordering has the **Add First** or **Add Last** setting for the **Order subtypes** property.
- Components - to reorder components, drag the component(s) to the desired location in the component list.
- Restrictions - to reorder restrictions, drag the restriction(s) to the desired location in the restriction list.
- [Reordering existing subtypes](#)

Reordering existing subtypes

About this task

If the **Order Subtypes** option is set to **Add First** or **Add Last**, you can arrange the subtypes in any order.

To reorder subtypes:

Procedure

1. Select the type.
2. Open the Properties view to confirm that the **Order subtypes** property is **Add First** or **Add Last**.
3. Right click on the type and choose **Reorder subtypes** from the context menu.
The Reorder subtypes dialog is displayed.
4. You can arrange the order by selecting a type and pressing the Up or Down buttons. You can also move a type by dragging it to a position.
5. Click Close to close the dialog.
6. Save changes.

Merging types

You can merge types from one tree to another. Merging a type copies that type and all types referenced by that type to another tree. The referenced types include any components, comment types, and any syntax items used as a number separator, initiator, terminator, release character, or delimiter. You can merge the type itself or the type with its subtree.

- [Merging a type](#)
- [Supertypes](#)
- [Existing types](#)
- [Invalid types](#)

Merging a type

Before you begin

Before using the **Merge** command, perform the following tasks:

- Analyze the type tree you are merging from.
- Analyze the type tree you are merging to.
- Ensure that the root of the type tree you are merging to has the same root name as the tree you are merging from.

About this task

To merge a type from one tree to another:

Procedure

1. Select the type you want to merge into another tree.
2. Right-click on the type and choose **Merge** from the context menu.
The Merge dialog is displayed.
3. When applicable, enable the Merge subtree check box to merge the entire subtree.
4. Open the destination type tree if it is not already open.
5. Select the destination tree.
6. Click Merge.
7. Click Close.

Supertypes

When you merge a type from one tree to another, the supertypes of that type are created. In the example above, notice that **Detail**'s supertype, **Record**, is included in the copied types.

Existing types

The **Merge** command copies a type only if it does not already exist in the destination tree. If the type already exists in the destination tree, it is not copied.

Invalid types

The **Merge** command will only merge a type if it can exist in the new tree. For example, if you try to merge an item into a tree where it would exist as a group, the type is not merged.

Types referred to by ANY in a component are not copied in a merge.

Renaming types

About this task

Remember: When you rename a type, it might move to a different position on the same level in the tree if types on that level are in alphabetical order.
To rename a type:

Procedure

1. Open the properties for a type.
2. From the Properties view, locate Property \gg Name.
3. Click in the Name field cell and enter a new name.
4. Save changes.

Related tasks

- [Using search and replace](#)

Generating a type definition report

About this task

Use the Generate Type Definition Report menu option to generate a type definition report file.

Generate a type definition report HTML file to view information about the type through a web browser, or an HTML editor. The generated type definition report file contains information, including properties, restrictions, components, and where the type is used, for the selected type, depending on the options that you select. It also includes the subtree if you select that option.

Procedure

To generate a type definition report file:

1. In the Type Tree editor of the Design Server, right-click a type in the type tree, and click Generate Type Definition Report.
The Generate Report window opens.
 2. Click Browse and specify in the Save As window, the file name, and path in which you want to generate the file.
If you choose to generate the report to an existing output file, click Yes at the prompt that gives you the option to replace the file.
 - a. To change the default current location in which to save the type definition report file, select a different location in the Save in list.
 - b. To change the default type definition report file name, enter File name.
- After you click Save, the path and file name are displayed in the Output file name field of the Generate Report window.
3. Under Type Definition Options, click the options that represent the information you want the report to include.
 4. Click Generate.
The Generate Type Definition Report window opens and displays the progress of the generate type definition report operation. The window remains open and the progress of the operation continues to display until the operation completes. There is an option available for the operation to run in the background. The reason is so that you can continue to access the Design Server graphical user interface (GUI). If the operation is long-running, you might want to select this option.
The operation writes the type definition to the output file that you specified based on the options that you selected.

Results

The type definition report file is saved in the specified location and opens in the Internal Web Browser editor that is in the Transformation Extender Development perspective. You can view the file output from within the Design Server or any external tool. You can also view the .html source file from the Design Server or numerous HTML editors.

Testing part of a type tree

You can test a type tree by generating test data for an item or a group. Type Designer creates source and compiled maps that generate and validate the test data. The test-data-generation map automatically runs and produces a text file with the sample data. After that, the test-data-validation map automatically runs on the sample data file.

For choices and partitions, Type Designer fills the first choice and first partition with data in the generation map. For sequences, Type Designer generates one index. Type Designer uses implied default-value and item-value restrictions if they are available; otherwise, it generates sample values for different data types.

- [Generating test data](#)
- [Invalid input message from test data generation](#)

The message One or more inputs was invalid displays when test data generates successfully but the validation map flags some data as not valid. This message can occur when there are data type constraints, component rules, or restrictions that the sample data does not account for. Trace the validation map and inspect the generation map rules to find the data types that are not generating correctly.

Generating test data

Procedure

1. Open the type tree and right-click on an item or group.
2. Click Generate Sample Data.
3. Change the sample data options, or accept the defaults. Optionally, you can set defaults for these options in Window > Preferences > Transformation Extender > Type Tree > Sample Data.

Create optional types

When this option is not selected, a type with the size **Min=0** and **Max=1** generates a map rule of **None**.

Fill types with data

When this option is not selected, all types generate a map rule of **None**.

Results

Type Designer generates the following files in the SampleData folder relative to the source type tree:

- A source map named *selected_type_name.mms*
- A compiled map named *selected_type_name.mmc*
- A source map named *Validation_selected_type_name.mms*
- A compiled map named *Validation_selected_type_name.mmc*
- A log file of the generation session named *SampleDataGenerator.log*
- A test data file named *selected_type_name.txt*

Invalid input message from test data generation

The message One or more inputs was invalid displays when test data generates successfully but the validation map flags some data as not valid. This message can occur when there are data type constraints, component rules, or restrictions that the sample data does not account for. Trace the validation map and inspect the generation map rules to find the data types that are not generating correctly.

Type properties

The properties of a type define the characteristics of the data objects of that type.

For *item* types, properties define whether that item is text, a number, a date and time, or a syntax value. Properties include such characteristics as size, pad characters, and justification.

For *group* types, the properties are related to the format of that group. The format of a group may be explicit or implicit. In addition, type properties include syntax objects that appear at the beginning or end of the object, as well as release characters.

- [Defining type properties](#)
- [Basic type properties](#)
- [Item properties](#)
- [Group properties](#)
- [XML properties in the type tree](#)

Defining type properties

About this task

You can define specific properties for each type. Type properties generally consist of the following:

- Definition of a type's **Class**, **Group** or **Item** properties
- Type **Initiator**, **Terminator**, and **Release** character
- A list of where a type is used in the definitions of other types

Related concepts

- [Basic type properties](#)

To access properties of a type

Procedure

Select a type and click the Properties button on the toolbar: 

The Properties view opens.

Tip: You can also access the Properties view from the Window menu. Select Window > Show View > Other > General > Properties.

Results

Now, when you select a type in the type tree editor, the corresponding type properties are displayed in the Properties view.

To define the properties of a type

Procedure

1. In the Properties view, enter all relevant information for the type. For example, the Name, Class, Description, and so forth. All information is entered in the Value column. To view a drop-down list of options (when available), you must click in the field (in the Value column) and the down arrow is displayed.
2. To define a certain property for the selected type, enter or select the property value in the **Value** column.
When needed, expand each property to view all associated values. For example, to define the type initiator as a literal value, for **Initiator**, select Literal from the drop-down list and then expand the **Initiator** property to define the literal initiator value.
3. Save changes.

Results

Continue defining additional type properties as needed.

Basic type properties

The following type properties are common to categories, groups, and items:

- [Name](#)
- [Class](#)
- [Description](#)
- [Intent](#)
- [Partitioned](#)
- [Order Subtypes](#)
- [Initiator](#)
- [Terminator](#)
- [Release Characters](#)
- [Empty](#)
- [National Language](#)
- [Document Type](#)
- [Where Used](#)

Item-specific properties are discussed in more detail in "[Item Properties](#)". Group-specific properties are discussed in more detail in "[Group Properties](#)".

- [Name](#)
- [Class](#)
- [Description](#)
- [Intent](#)
- [Partitioned](#)
- [Order subtypes](#)
- [Initiator](#)
- [Terminator](#)
- [Release characters](#)
- [Empty](#)
- [Document Type](#)
- [Where used](#)
- [Symmetric swapping](#)
- [Orientation](#)
- [Shaping](#)

Related concepts

- [XML properties in the type tree](#)

Related tasks

- [Defining type properties](#)

Name

The name of the type should be as descriptive as possible and reflect the information that the type represents.

A type name must conform to the following guidelines:

- A type name cannot be more than 256 characters long.
- A type name cannot be a reserved word or contain a reserved symbol. For a list of reserved words, refer to Design Studio Introduction documentation.
- A type name cannot contain only digits and/or periods.
- A type name may contain numbers and special characters.
- A type name cannot contain spaces. Use an underscore instead.
- A type name can contain the following:
 - Letters
 - Digits
 - ASCII characters 128-255
 - Special characters: #, ~, %, _, ? or \
 - Double-byte characters (Japan edition only)
- A type name cannot have the same name as another type on the same level in the same type tree.
A type name can be up to 256 characters in length, but use a short type name if possible. The full path name of each type is used in map rules in the Map Designer.

Class

The class of a type describes whether the type represents a simple data object (item), a complex data object (group), or whether it is used to organize types (category). Define the class of the selected type by selecting the class from the drop-down list in the **Value** column.

Category

Categories organize types that have common properties. Each category has a set of item properties and a set of group properties. Subtypes of the category inherit these properties.

A category does not define data objects in detail and it does not represent data to be used in a map. You never map a category type, so a category type does not have components. A category type may be a component of a partitioned group, after a component with the identifier attribute. A category cannot be a component of a non-partitioned group.

Item

The item type represents a simple data object that does not consist of any objects. An item does not have components.

Group

The group type represents a complex data object that consists of components.

To change a type's class, select the type and choose the desired class for the **Class** property in the Properties window.

If you change a type to an item, all the types in its subtree become items. If you change a type to a group, all the types in its subtree become groups. This is because the type tree has a classification hierarchy and all the nested subtypes of an item must be items. Similarly, all the nested subtypes of a group must be groups.

Description

Use this property to record a brief description of the type. The description entered is for informational purposes only. It is not used for identifying data objects at runtime.

It is good practice to add a meaningful description of the type when you are defining it.

Intent

Indicates whether the type is a general type or an XML type.

The type can only be an XML type if the type tree was created using the XML DTD Importer or the XML Schema Importer.

Refer to "[XML Properties in the Type Tree](#)" for more information about type properties created from XML.

Partitioned

If the data of this type can be divided into mutually exclusive subtypes, it can be partitioned. For information about partitioning, see "[Partitioning](#)".

- Yes
Partitioning is enabled for this type.
No
Partitioning is not enabled for this type.
-

Order subtypes

Choose the method in which the subtypes of this type will be added or viewed in the type tree.

- Ascending
Add and view subtypes of this type in alphabetic or numeric order.
Descending
Add and view subtypes of this type in reverse alphabetic or numeric order.
Add First
Add subtypes of this type to the top of the type list.
Add Last
Add subtypes to the bottom of the type list.

To manually reorder the subtypes in a type tree, the **Order Subtypes** property must be either **Add First** or **Add Last**.

Initiator

An initiator is a syntax object that appears at the beginning of a data object. Defining an initiator for a type specifies that when data of that type appears, the initiator appears at the beginning of the data object. The initiator becomes part of the data type definition.

- None
There is no initiator.
Literal
The initiator is literal. Expand the **Initiator** property to enter the literal initiator value and data language of the initiator value.
Variable
Allow for possible values. Expand the **Initiator** property to define the variable terminator **Default**, **Item**, and **Find** properties.

If each record begins with an asterisk *, define the * as a literal initiator of the record type.

The following data represents the classes in a college English department. An asterisk * is displayed at the beginning of each **ClassRecord**.

For information about other symbols used in the Value field of the Initiator property, see the Type Tree Importer documentation.

Terminator

A terminator is a syntax object that appears at the end of a data object. The terminator becomes part of the data type definition.

- None
There are no terminators.
Literal
A constant value. Expand the **Terminator** property to define the literal terminator **Value**.
Variable
Allow for possible values. Expand the **Terminator** property to define the variable terminator **Default**, **Item**, and **Find** properties.

For example, a carriage return/linefeed (CR/LF) at the end of a record is the record's terminator.

Generally, if the data ends with a given syntax object, you should define a literal terminator. For example, it is very common to define a CR/LF as a terminator of a record when you know that the record always ends with a CR/LF, regardless of where the record appears.

Release characters

A release character is a one-byte character in your data indicating that the character(s) following it should be interpreted as data, not as a syntax object. The release character is not treated as data, but the data that follows it is treated as actual data.

- [Building release characters for output data](#)
 - [Guidelines for using release characters](#)
 - [Release character example](#)
-

Building release characters for output data

If a release character is defined for a type, a release character is inserted for each occurrence of a syntax object in the data of any item contained in that type.

Guidelines for using release characters

Guidelines for using release characters include the following:

- Release characters apply to character data only, *not* binary data.
- Characters defined as pad characters are not released.
- The maximum size of an item does *not* include the release characters.

Release character example

The group type **Record** type has a literal delimiter of , and a release character of ?. The group type **Record** has three item components. Data for the record looks like the following:

Miller?, MD,Harkin Hospital,1996

The ? releases the comma after **Miller**.

In the first field, the actual data value is **Miller, MD**. Because the comma appears as part of the data, it is necessary to have the release character ?, which indicates that the , following it is data, *not* a delimiter. This data would be interpreted as the following:

Data for component #1: **Miller, MD**

Data for component #2: **Harkin Hospital**

Data for component #3: **1996**

A release character can apply to a delimiter, a terminator, or even the release character itself.

If a release character appears in the data and it is not followed by a syntax item, the release character is ignored.

Empty

The **Empty** property provides alternative type syntax for groups or items when they have no data content.

When the **Empty** property is specified for a type and there is no data content, the **Empty** syntax is displayed. For example, this can be used for XML data that contains either start and end tags or an empty tag.

You can use the **Empty** type property instead of syntax object items to potentially improve the type tree runtime processing time (during data validation).

The following options are available for the **Empty** property.

None

Default setting. Select None if you do not have an initiator, terminator, or release character.

Literal

A constant value. When **Literal** is selected, the **Value**, **Ignore Case**, **Required**, and **National language** sub-fields become available. You can use only one literal to indicate a zero-length data item.

- [Empty type property example](#)

Empty type property example

Consider an XML element called **Comment**. Presuming that this element has a simple content of an arbitrary length, it can be represented in a type tree as a text item, with the initiator value <**Comment**> and the terminator value </**Comment**>.

This item can then be used to validate the following data:

<**Comment**>Some comment...</**Comment**>

The resulting value for the item will be:

Some comment...

As another example, the following data is also successfully validated:

<**Comment**></**Comment**>

The resulting value for the item will be a zero-length string.

The problem occurs with the following data: <**Comment**/>

From the XML perspective, this data is equivalent to the <**Comment**></**Comment**> data shown above, as it represents the **Empty** element **Comment**. However, from the type tree perspective, this data is invalid because it does not start with <**Comment**> and does not end with </**Comment**> as required by the initiator and terminator item property values.

This is where the **Empty** property is useful. By defining the **Empty** property for the item to have a value of <**Comment**/>, it will be possible to validate the data <**Comment**/>.

Even if the data does not match the syntax described by the initiator and terminator properties, it will be validated because it will match the **Empty** property value. It will be treated as an *empty item*, that is, the resulting value for the item will be a zero-length string, similar to the example shown previously.

Document Type

The **Document Type** setting is a component of Document Verification. A requirement of using Document Verification is to set the **Document Type** properties of the associated type tree object.

To use the Document Verification option for XML documents, you must do the following:

- Set the **Document Type** property to **XML**.
- Set the Document Type Metadata value to **Schema** or **DTD**.
- Specify the **Location** of the DTD or Schema file.

Default

This is the default value, which indicates a non-XML document type.

XML

Indicates an XML document type. When this value is set to **XML** and the appropriate **DocumentVerification** map settings are in place, the XML document will be validated by an external program in addition to the standard data validation process.

See the Map Designer documentation for information about using the Document Verification option.

Where used

Where Used shows how and where the type is used within other types in the tree. For example, the **Where Used** property shows if the type is used as a delimiter, component, and so on.

Symmetric swapping

When the input and output objects have different symmetric swapping characteristics, the Symmetric Swapping property is considered "on" for the related mapping rule.

Examples

Original text	Resulting text
abcF (E) Dghi	abcF (E) Dghi
Input Object No swapping	
Output Object Swapping	
Original text	Resulting text
abcF (E) Dghi	abcF (E) Dghi
Input Object Swapping	
Output Object No swapping	
Original text	Resulting text
abcF (E) Dghi	abcF) E (Dghi
Input Object No swapping	
Output Object No swapping	
Original text	Resulting text
abcF (E) Dghi	abcF) E (Dghi
Input Object Swapping	
Output Object Swapping	

Orientation

Orientation is to change characters from one direction to a different direction.

Example

A simple orientation example is to define English characters "abc" as a bidirectional object with left-to-right (LTR) orientation. If you moved the object to a right-to-left (RTL) object, the output should be "cba". The following example demonstrates a string of characters that change in orientation.

Original text	Resulting text
ABC DEF Ä ÄíÄé Input left-to-right (LTR) Output right-to-left (RTL)	ÄÄíÄé FED CBA

Shaping

Text shaping is the concept of combining or separating characters where possible. The Arabic usage is with ligatures (such as the LAM, ALEF, and LAM-ALEF characters). For example, in a non-shaped word, both LAM and ALEF characters would appear separately. In a shaped word, the two characters would be replaced with a single LAM-ALEF character.

Numeric shaping is the idea that normal regular numerals (0–9) can be shaped to be Arabic-Indic

Examples

- Ü,, = LAM
- i° = ALEF
- Ü°i»» = LAM ALEF

Original text	Resulting text
abcii»»def Input Shaped Output Unshaped	abcÜ,,i°def
abcÜ,,i°def Input Unshaped Output Shaped	abci»»def

Item properties

Item types define data for a selected type. Item types are divided into the following subclasses: **Number**, **Text**, **Date & Time**, and **Syntax**. Each item subclass has a specific set of properties.

- [Item subclass](#)
- [Number item subclass properties](#)
- [Text item subclass properties](#)
- [Date & Time item subclass properties](#)
- [Syntax item subclass properties](#)

Item subclass

The subclass of an item represents the characteristics of the data. Both category and item types have the **Item Subclass** property.

To define the subclass of an item type:

1. Access the **Properties** for the selected category or item type.
2. For the **Item Subclass** property, select a value that defines the data for the type:

Number

Any number excluding active syntax objects. The interpretation can be either character or binary. See "[Number Item Subclass properties](#)".

Text

Any character excluding active syntax objects. The interpretation can be either character or binary. See "[Text Item Subclass properties](#)".

Date & Time

A valid date and time format based on the data. The interpretation can be either character or binary. See "[Date & Time Item Subclass properties](#)".

Syntax

Syntax objects are used as separators between portions of data. A restricted set of values used to define a dynamic delimiter, initiator, terminator, or release character. The interpretation is character. A syntax object cannot be longer than 120 bytes. Syntax objects, used as syntax, cannot be mapped. See "[Syntax](#)"

[Item Subclass properties](#).

For text and syntax objects, character size constraints can be in bytes or characters. Numbers are considered in digits and date-time formats are sized by the format, such as CCYYMMDD.

Number item subclass properties

Number item subclass properties can be of **Character** or **Binary** value. When you choose **Number** as the **Item Subclass** value, you must choose an **Interpret as** value: **Binary** or **Character**.

The **Interpret as** value defines how the data should be interpreted. Items with a subclass of **Number**, **Text**, and **Date & Time** can be interpreted as either character or binary. Items with a subclass of **Syntax** can be interpreted as character only.

- [Interpret as binary](#)
- [Length \(bytes\)](#)
- [Byte order](#)
- [Interpret as character](#)
- [Size \(digits\)](#)
- [Separators](#)
- [Pad](#)
- [Restrictions](#)
- [National language](#)
- [NONE and Zero](#)
- [Places](#)

Interpret as binary

Binary text items have content size and pad properties. Binary data is required to be sized or of a fixed size.

Binary number items interpret the data as a binary number or as a byte stream.

Items with an item subclass of **Text** can be interpreted as character or binary.

When **Interpret as** is set to **Binary**, the binary presentation options are Integer, Float, Packed, or BCD.

- [Binary integer presentation](#)
- [Binary float presentation](#)
- [Binary packed presentation](#)
- [Binary BCD presentation](#)

Binary integer presentation

The **Integer** value is a whole number.

Length(bytes)

Can have a length of **1**, **2**, **4**, or **8** bytes

Byte order

See "[Byte order](#)".

Sign

See "[Sign](#)".

Binary float presentation

The **Float** value is a number with decimals in a location as needed.

Length(bytes)

Can have a length of **4**, **8**, or **10** bytes

Binary packed presentation

The **Packed** value is a number that has size, decimal places, and sign properties.

The binary packed properties include Length, Implied places, and Sign.

Length

Can have a length of **1 - 16** bytes

Implied places

Can be from **0 - 31**

Sign

Can be **Trailing-** or **Trailing+**

For **Packed** numbers, the **Implied places** cannot exceed the **Length**.

Binary BCD presentation

The **BCD** value is a binary coded decimal number that has size.

Length (bytes)

Use the **Length (bytes)** property to select the number of bytes that equals the length of the item.

The **Length(bytes)** property is an option when the **Number** item subclass property is interpreted with a **Binary** value.

Byte order

Use the **Byte order** property to define the way that bytes in the data are ordered by selecting one of the following values from the Byte order drop-down list.

Value	Description
Big Endian	The most significant byte has the lowest address. (Usually systems such as IBM, HP, and Solaris.)
Little Endian	The least significant byte has the lowest address. (Usually systems such as Intel and VAX.)
Native	The order is dictated by the platform.

Byte order is an option when the **Number** item subclass is interpreted with a **Binary** value.

Interpret as character

Character number items interpret the data as symbolic data. Symbolic data has the same meaning on different computers. For example, a comma is symbolic because it has the same meaning, regardless of the computer type.

Character text items can have content size and pad properties.

When the **Interpret as** value is defined as **Character**, the **Release** property is specific to the **Character** value and is *not* available as an option for a **Binary** value. For more information, see "[Release characters](#)".

When **Interpret as** is set to **Character**, integer, decimal, and zoned presentation options are available.

- [Character integer presentation](#)
- [Character decimal presentation](#)
- [Zoned character presentation](#)
- [Places > implied](#)

Character integer presentation

The **Integer** value is a whole number.

Further information on the **Integer** value is found in "[Integer Separators](#)".

Character decimal presentation

The **Decimal** value is a number that contains decimals.

More information about the **Decimal** value is found in section "[Decimal Separators](#)".

Zoned character presentation

The **Zoned** value is a number that has a size, decimal place, pad, and sign properties.

Use the "Size (digits)" property to specify the size of the zoned number. The size of a zoned number can also be specified in terms of minimum and maximum. The minimum size must be less than or equal to the maximum size.

When the content size of a character-zoned number is used, the last digit and the sign are combined into the same digit but the content size does *not* include the initiator, terminator, release characters, or pad characters.

When the content size of a character-zoned number is important (for example, when the **SIZE** function is used on the character zoned number), the content size includes the sign but does not include the initiator, terminator, release characters, or pad characters.

Use the "[Sign](#)" property to define whether the zoned number will be signed. The default value is **No**.

The size of a zoned number is specified in terms of the minimum and maximum number of digits. If the size is specified with the **Min** and **Max** properties, the sign is *not* included.

Example

Zoned Number Not Signed	Signed Zoned Number
1230	123{
1231	123A
(Length = 4 digits)	(Length = 4 digits)

Places > implied

Numbers have implied decimal places. The number of decimal places cannot exceed the length specification. The list of available decimal places varies with the length selected. In the **Value** column, enter a total number of decimal places (not to exceed 31).

Size (digits)

Use the Size (digits) property to specify the minimum and maximum size of a number item.

Min

The minimum number of digits of the data object.
The default value is 0.

Max

The maximum number of digits of the data object.

If there is no maximum size, a **Max** value is not required.

- [Excluded from min and max size](#)
 - [Size example](#)
-

Excluded from min and max size

For character number items, there are certain objects excluded from the **Min** or **Max** count. For each **Presentation** option, the following table lists what is excluded from the minimum or maximum **Size (content)** count.

Presentation:	Integer	Decimal	Zoned
	initiators	initiators	initiators
	pad characters	pad characters	pad characters
	release characters	release characters	release characters
	separators	separators	terminators
	symbols	symbols	
	terminators	terminators	

Size example

In the following ASCII number there are a total of 12 characters:

+1234567.999

However, because the initiator (+) and separator (.) are not counted, the number would validate for a maximum size of 10 (**Max=10**).

Separators

Integer and decimal number items can have a separator for thousands and fractional places. If you opt to have a separator, choose **Yes** and sub-options will appear that enable you to choose the format and syntax.

The **Zoned** value of the **Presentation** property does not have separators.

- [Integer separators](#)
- [Separator > format](#)
- [1000's syntax > value](#)

- [1000's syntax \(literal\) > value](#)
 - [1000's syntax \(variable\) > default](#)
 - [1000's syntax \(variable\) > item](#)
 - [1000's syntax \(variable\) > find](#)
 - [Decimal separators](#)
 - [Separators > format](#)
 - [Separators > 1000's syntax](#)
 - [Separators > fraction syntax](#)
 - [Sign](#)
-

Integer separators

For integer numbers, the separator is the thousands separator.

Integer character numbers have separator properties of:

- Format
- 1000's Syntax
- Value

These properties specify the placement and value of the thousands separator.

Separator > format

Use the Separators Format property to specify the placement and value of the thousands separator.

To define the separator format, select one of the separator formats from the drop-down list in the **Value** column.

#[.]####

The thousands separator is not required on input, but if present, it must be in the proper location. The separator is not built on output.

#.####

The thousands separator is required on input in the proper location. The separator is built on output.

1000's syntax > value

Define the thousands separator as either literal or variable. Select one of the following from the drop-down list in the **Value** column:

Literal

The thousands syntax is a constant literal specified in the **1000's Syntax > Value** column.

Variable

The thousands syntax allows for variable values. Use the **1000's Syntax > Default**, **Item**, and **Find** properties to define the variable fraction separator.

1000's syntax (literal) > value

For the literal thousands separator, define the literal separator by typing the literal separator character in the **Value** column or by clicking the browse button to display the Symbols dialog box from which you can insert any non-printable value.

A separator can be from one to 120 bytes in length. A separator cannot start or end with a digit and cannot be the ? character. The ? character is a reserved character that can be used as a wildcard to represent any single valid character.

1000's syntax (variable) > default

Use the **1000's Syntax > Default** property to define the default variable separator value for the thousands place. The default literal separator value for thousands syntax is a comma.

Enter the default variable separator character or click the browse button to display the Symbols dialog box in which you can insert any non-printable value.

1000's syntax (variable) > item

For integer numbers with a variable thousands separator, the **Separator** property is **Yes**, and the **1000's Syntax** is **Variable**, you can specify an item type for the variable thousands separator.

You can select the desired syntax item from the drop-down list. Or, with the focus on the **Item** property (in the **Value** column), press **Alt** and drag the default separator type item from the type tree editor into the **Value** column.

An item type with a class of **Syntax** must exist in the type tree to be a valid **1000's Syntax Item**.

1000's syntax (variable) > find

For integer numbers with a variable thousands separator, the **Separator** property is **Yes** and the **1000's Syntax** is **Variable**. The value of the separator can be the current value or the value of the separator can be determined each time an occurrence of that type is found.

Yes

Determine the value of the separator each time an occurrence of that type is found. After the value of that separator is found, that particular value is used until it is reset either by another Find or by the occurrence of that separator as a component.

No

The system uses the value to which the separator item is currently set or, if not set, it uses the default value.

Decimal separators

For decimal numbers, the separator is the thousands separator and the fractional separator.

Decimal character numbers have separator properties of:

- Format
- 1000's Syntax
- Fraction syntax
- Value

These properties specify the placement and value of the fractional separator.

The item is assumed to have an implicit number of decimal places. To specify the number of implied decimal places, use the **Item Subclass Places** property to specify the number of implied decimal places.

Separators > format

Use the Separators Format property to specify the placement and value of the fraction separator.

To define the separator format, select one of the separator formats from the drop-down list in the **Value** column.

#####[.##]

The fractional separator is present if there are fractional digits. There is no thousands separator.

#####,##

The fractional separator is always present, even if there are no fractional digits. There is no thousands separator.

#[,###][.##]

The fractional separator is present if there are fractional digits. There is an optional thousands separator.

#[,##][,##]

The fractional separator is always present. There is an optional thousands separator.

,###,##

The fractional separator is always present. There is always a thousands separator if there are more than three whole number digits.

Separators > 1000's syntax

Use the Separators 1000's Syntax to define the thousands syntax. Select one of the following from the drop-down list in the **Value** column:

Literal

The thousands syntax is a constant literal specified in the **1000's Syntax > Value** column.

Variable

The thousands syntax allows for variable values. Use the **1000's Syntax > Default**, **Item**, and **Find** properties to define the variable fraction separator.

See the following topics for additional information:

- ["1000's Syntax \(Literal\) > Value"](#)
- ["1000's Syntax \(Variable\) > Default"](#)
- ["1000's Syntax \(Variable\) > Item"](#)
- ["1000's Syntax \(Variable\) > Find"](#)

Separtors > fraction syntax

Use the **Fraction syntax** property to define the fraction syntax as either a constant literal value or a variable value.

You can define the **Fraction syntax** as literal or variable.

Literal

A constant value.

Use the Fraction syntax > Value property to select a literal fraction separator from the **Symbols** dialog box.

Variable

Allow for variable fraction separator values. The following options are available to define the variable fraction separator:

- **Default** - Use this property to define the default variable separator value for fractions. The default literal separator value for fractions is a period. Enter the default variable separator character or click the browse button to display the Symbols dialog box in which you can insert any non-printable value.
- **Item** - For decimal numbers with a variable fractional separator, the **Separator** property is **Yes** and the **Fraction syntax** is **Variable**, you can specify an item type for the variable fraction separator.
Note: An item type with a class of **Syntax** must exist in the type tree to be a valid **Fraction syntax Item**.
- **Find** - For decimal numbers with a variable fraction syntax, the **Separator** property is **Yes** and the **Fraction Syntax** is **Variable**, the value of the separator can be either the current value or the value of the separator can be determined each time an occurrence of that type is found. Select an option:
 - Yes - Determines the value of the separator each time an occurrence of that type is found. After the value of that separator is found, that particular value is used until it is reset by another find or by the occurrence of that separator as a component.
 - No - The system uses either the value that the separator item is set to currently, or, if it is not set, uses the default value.

Sign

Use the Item Subclass Sign property to define whether the number is signed for either the **Integer** or **Decimal** value. A sign is a symbol that identifies a number as being either positive or negative. A positive sign is plus (+); a negative sign is negative (-).

Yes

The number is signed. Expand the **Sign** property to define the sign values. If the **Sign** property is **Yes**, a minimum of at least one sign value (**If number is +**, **If number is -**, or **If number is 0**) must be specified and at least one value must be required on input.

No

The number is not signed. This is the default setting.

When you select Yes, you can further define the sign values. The following sub-options are available:

Leading-

The sign precedes the number when it is negative only. For input, a sign is required for negative data, but is optional for positive data.

Trailing-

The sign follows the number when it is negative only. For input, a sign is required for negative data, but is optional for positive data.

Leading+

The sign always precedes the number. A sign is required for input and output data.

Trailing+

The sign always follows the number. A sign is required for input and output data.

Custom

Defaults to **Leading-** but can be changed.

Sign values may be specified for the following:

- **If Number is +** (positive numbers)
- **If Number is -** (negative numbers)
- **If Number is 0** (value of zero)

For each value (**If number is +**, **If number is -**, or **If number is 0**), specify the following:

Leading sign

Enter the symbol to be placed before a number.

Trailing sign

Enter the symbol to be placed after a number.

Required on input

If the number has a sign, at least one value (**If number is +**, **If number is -**, or **If number is 0**) must be required on input. Select an option from the drop-down list:

Yes

The sign is mandatory on input.

No

The sign is optional on input.

Pad

Pad implementation determines how the pad value is sized during validation and how the pad value is sized when written as output.

When the data value to be mapped to the target item is smaller than the minimum length of that item, pad characters can be used to pad the data to that minimum length. Input data can contain both content and pad characters. Output data is built according to the pad definitions of the types. Bytes are the default measurement for sizing, however, both byte and character sizing options are available.

Yes

Enables the **Pad** option. Allows the item to contain both content and pad characters.

No

All data is assumed to be content on input; no pad characters are built on output.

- [To specify a pad character](#)
- [Pad > value](#)
- [Pad > Padded to](#)
- [Padded to > length](#)
- [Padded to > SizedAs](#)
- [Padded to > CountsTowardMinContent](#)
- [Padded to > Allow Excess Trailing Pads](#)

- [Pad > justify](#)
 - [Pad > apply.pad](#)
 - [Pad > fill](#)
-

To specify a pad character

Procedure

1. Select a type and open the properties.
 2. In the Properties view, expand the options under Item Subclass.
 3. Set the Pad property to Yes.
 4. Expand the Pad properties.
 5. Click the browse button for the Pad._>.Value property to view a list of symbols.
 6. Select a symbol, and press Insert.
 7. Click OK.
- The new pad character appears in the Pad field.
-

Pad > value

Use the Pad Value property to define a 1-character pad character. The default pad value is 0.

Type the pad character symbol between angled brackets < > in the **Value** field or click the browse button to display the Symbols dialog box to insert any non-printable value.

For example, to enter a **FormFeed** value for the pad character, in the **Value** field enter <FF> or click the browse button to select the **FF (FormFeed)** symbol from the Symbols dialog box.

Pad > Padded to

Use the Pad._>.Padded to property to define whether the data item is padded to a fixed size or to the minimum content size defined for the data object.

Fixed size

The data object is padded to a fixed size (in bytes or characters) that you specify in the Length field.

For any item padded to a fixed size, the item must have a value specified for the Size (content)._>.Max property and the Padded to._>.Length value must be greater than or equal to the Size (content)._>.Max value.

Min Content

The data is padded to the value specified as the minimum size (in the Size (content)._>.Min field).

You can further specify whether to count pad characters toward the length using the CountsTowardMinContent subproperty.

Padded to > length

To pad to a fixed size, you must specify the length.

Input data can contain both content and pad characters but when an item is built for output, the item is padded to the number of bytes or characters specified in this field.

The Padded to._>.Length value must be greater than or equal to the Size._>.Max value.

You can specify the method of measurement (bytes or characters) using the SizedAs subproperty.

Padded to > SizedAs

Use the Padded to._>.SizedAs property to specify that padding is measured in either bytes or characters. The default setting is Bytes.

Trace file and audit log lengths are always reported in bytes.

Table 1. Pad implementation example (bytes compared to characters)

Scenario for empty field:	Result:
Data language is UTF-8	Two À pad characters using 4 bytes.
Pad. _{>} .Value = À (hexadecimal representation = 0xC3 0x80)	
Pad. _{>} .Padded to = Fixed Size	
Padded to. _{>} .Length = 5	
Padded to. _{>} .SizedAs = Bytes	

Scenario for empty field:	Result:
Data language is UTF-8	Five \AA pad characters totaling 10 UTF-8 bytes.
Pad > Value = \AA (hexadecimal representation = 0xC3 0x80)	
Pad > Padded to = Fixed Size	
Padded to > Length = 5	
Padded to > SizedAs = Characters	

Padded to > CountsTowardMinContent

When you select Padded to > Min Content, the **CountsTowardMinContent** field is displayed. Use this setting to specify if pad characters should count toward the length of an object when determining if it meets its minimum content length.

Yes

Pad characters are included in the count for the length of an object.

No

Pad characters are not counted toward the length of an object.

You can use the **Propagate** function here to propagate the present **CountsTowardMinContent** value to the subtypes of the present type, if present.

If you are using the trace option, there might be cases where in the trace file the content length appears to be different than the input length. A trace file lists the input length (the length used to validate an object), which includes the input length of each object plus the pad characters. In the case of numbers, signs and separators are also counted in the length.

CountsTowardMinContent > AcceptAllPads

The **AcceptAllPads** property is applicable to data objects that contain only pad characters. Use this property when you need an object that contains only pad characters (no content) to either pass or fail validation depending on whether it is a mandatory or an optional type, regardless of the minimum size requirement.

Yes

When an object contains only pad characters, the minimum size requirement is ignored and the object passes size validation. The object then passes or fails type validation depending on whether it is a mandatory or an optional type.

No

(Default setting) Pad characters do not count toward the minimum size requirement, therefore an object that contains only pad characters and that does not meet the minimum size (content) requirement fails size validation.

Example

The results of the following examples are based on these values:

Size (content)

5

Pad

Yes

Pad > Padded to

Min Content

Padded to > CountsTowardMinContent

No

Default Behavior: When **AcceptAllPads** is set to **No**, the following results occur:

Input Data (X = pad character)

Valid?

ABCDE

Yes

ABC

No

ABCXX

No

XXXXX

No

When **AcceptAllPads** is set to **Yes**, the following results occur:

Input Data (X = pad character)

Valid?

ABCDE

Yes

ABC

No

ABCXX

No

XXXXX

Yes (when type is optional)

No (when type is mandatory)

X

Yes (when type is optional)
No (when type is mandatory)

Padded to > Allow Excess Trailing Pads

When you select Padded to \geq Min Content, the Allow Excess Trailing Pads property is displayed. Use this setting to flag an element as not valid when the following conditions apply:

- The element is padded to a minimum size.
 - The number of pad characters exceeds the number that is required to achieve the minimum size.
-

Pad > justify

Use the **Justify** property to specify whether the data is padded to the left or right.

Left

The data will be on the left and will be padded (if necessary) on the right.

Right

The data will be on the right and will be padded (if necessary) on the left.

Use the **TRIMLEFT** and **TRIMRIGHT** functions to exclude pad characters from the justified side.

In the following example, xxxxxxxxxxx represent 10 spaces:

For an input of 1234567891xxxxxxxx with a right justified pad, the rule =SIZE(input) returns 20. For the same input, the rule =SIZE(TRIMRIGHT(input)) would return 10 (removing the padding on the right).

Pad > apply pad

Use the **Apply pad** property to specify when to apply the pad character. Choose a value from the drop-down list.

Fixed Group

Apply the pad characters only when the item appears in a fixed group.

Any context

Apply the pad characters when the item appears in any context.

Each item in a fixed group must be padded to a fixed size or have the same value for the minimum and maximum content size.

For example, suppose the item **Name** has a space pad character with this value:

Mary<sp><sp>

If the **Apply pad** property is **Fixed Group**, the two spaces at the end are treated as pad characters only when the item appears in a fixed group. If the **Apply pad** property is **Any context**, then the spaces are always treated as pad characters.

Each item in a fixed group must be padded to a fixed size or have the same value for the minimum and maximum content size.

Pad > fill

For padded signed numbers, this option determines placement of pad characters. Choose a value from the drop-down list.

After sign

Pad characters are placed after the sign.

Before sign

Pad characters are placed before the sign.

Restrictions

Restrictions of an item are the valid values of that item. When the **Interpret as** value is defined as **Character**, the **Restrictions** property is specific to the **Character** value and is not available as an option for a **Binary** value.

Depending on other **Item Subclass** settings, the **Restrictions** property can be set to **Value**, **Character**, or **Range**.

- [Restrictions > ignore case](#)
 - [Restrictions > rule](#)
-

Restrictions > ignore case

By default, restrictions are case-sensitive. To enable or disable the **Ignore case** property of the **Restriction**, choose from one of the following options:

Yes

Ignore case of restrictions in item type properties.

No

Do not ignore case-sensitive restrictions.

For example, if **Ignore Case = Yes** and the restriction **Value** is **ft**, the data values **ft**, **fT**, **Ft**, and **FT** are all valid.

Restrictions > rule

This setting indicates whether you are going to include or exclude the criteria (as "valid" or "invalid") specified for the restrictions.

Include

Includes the restriction values as valid data.

Exclude

Excludes the restriction values as invalid data.

National language

The National language default value is Western. For initiator, terminator, and release character **Literal** values, you can optionally specify a "[Data language](#)".

When the **Interpret as** value is defined as **Character**, the **National language** property is specific to the **Character** value and is not available as an option for a **Binary** value.

- [National language > data language](#)

National language > data language

Use the National language > Data language property to define the data language or character set of this character text item.

- [Supported code pages](#)

Supported code pages

IBM Transformation Extender supports the following code pages:

Arabic
ASCII (deprecated)
Big5
Big5-HKSCS (IBM)
BOCU-1
CESU-8
CII Kanji (deprecated)
Cyrillic
EBCDIC (deprecated)
ebcdic-ar
ebcdic-cp-ar1
ebcdic-cp-ar2
ebcdic-cp-be/ch
EBCDIC-CP-DK/NO
ebcdic-cp-es
ebcdic-cp-fi/se/sv
ebcdic-cp-fr
ebcdic-cp-gb
ebcdic-cp-he
ebcdic-cp-it
ebcdic-cp-roece/yu
ebcdic-de
ebcdic-he
ebcdic-is
EBCDIC-JP-kana
ebcdic-xml-us
EUC (deprecated)
EUC-CN
EUC-JP
euc-jp-2007
EUC-TW
euc-tw-2014
GB_2312-80
gb18030

GBK
Greek8
Hebrew
hp-roman8
HZ-GB-2312
IBM Kanji (deprecated)
ibm-037 (ebcdic-cp-us/ca/wt/nl)
ibm-1006
ibm-1025
ibm-1026
ibm-1047
ibm-1047-s390
ibm-1097
ibm-1098
ibm-1112
ibm-1122
ibm-1123
ibm-1124
ibm-1125
ibm-1129
ibm-1130
ibm-1131
ibm-1132
ibm-1133
ibm-1137
ibm-1140 (ebcdic-us-37+euro)
ibm-1140-s390
ibm-1141 (ebcdic-de-273+euro)
ibm-1141-s390
ibm-1142 (ebcdic-dk/no-277+euro)
ibm-1142-s390
ibm-1143 (ebcdic-fi/se-278+euro)
ibm-1143-s390
ibm-1144 (ebcdic-it-280+euro)
ibm-1144-s390
ibm-1145 (ebcdic-es-284+euro)
ibm-1145-s390
ibm-1146 (ebcdic-gb-285+euro)
ibm-1146-s390
ibm-1147 (ebcdic-fr-297+euro)
ibm-1147-s390
ibm-1148 (ebcdic-international+euro)
ibm-1148-s390
ibm-1149 (ebcdic-is-871+euro)
ibm-1149-s390
ibm-1153
ibm-1153-s390
ibm-1154
ibm-1155
ibm-1156
ibm-1157
ibm-1158
ibm-1160
ibm-1162
ibm-1164
ibm-1250
ibm-1251
ibm-1252
ibm-1253
ibm-1254
ibm-1255
ibm-1256
ibm-1257
ibm-1258
ibm-12712-s390
ibm-1276 (Adobe Standard Encoding)
ibm-1363 (korean)
ibm-1363_P110-1997
ibm-1364
ibm-1371
ibm-1373_P100-2002
ibm-1386_P100-2001
ibm-1388
ibm-1390
ibm-1399
ibm-16684
ibm-16804-s390
ibm-33722_P120-1999
ibm-37-s390
ibm-437

ibm-4517
ibm-4899
ibm-4909
ibm-4971
ibm-5123
ibm-5346
ibm-5347
ibm-5348
ibm-5349
ibm-5350
ibm-5351
ibm-5352
ibm-5353
ibm-5354
ibm-5471_P100-2006
ibm-720
ibm-737
ibm-775
ibm-803
ibm-813
ibm-8482
ibm-850
ibm-851
ibm-852
ibm-855
ibm-856
ibm-857
ibm-858
ibm-860
ibm-861
ibm-862
ibm-863
ibm-864
ibm-865
ibm-866
ibm-867
ibm-868
ibm-869
ibm-874
ibm-875
ibm-901
ibm-902
ibm-9067
ibm-916
ibm-921
ibm-922
ibm-930
ibm-933
ibm-935
ibm-937
ibm-939
ibm-9447
ibm-9448
ibm-9449
ibm-949_P110-1999
ibm-949_P11A-1999
ibm-950_P110-1999
ibm-954_P101-2000
ibm-971_P100-1995
ibm-eucKR
IBM-Thai
IMAP-mailbox-name
ISO_2022,locale=ja,version=0
ISO_2022,locale=ja,version=1 (JIS)
ISO_2022,locale=ja,version=2
ISO_2022,locale=ja,version=3 (JIS7)
ISO_2022,locale=ja,version=4 (JIS8)
ISO_2022,locale=ko,version=0
ISO_2022,locale=ko,version=1
ISO_2022,locale=zh,version=0
ISO_2022,locale=zh,version=1
ISO_2022,locale=zh,version=2
JIS (deprecated)
KOI8-R
KOI8-U
Latin-9
Latin1
Latin1 (deprecated)
Latin2
Latin3

Latin4
Latin5
Latin6
Latin8
LMBCS-1
macintosh
MS_Kanji
Native
SCSU
Shift_JIS
shift_jis78
SJIS (deprecated)
Thai8
UNICODE Big Endian (deprecated)
UNICODE Little Endian (deprecated)
US-ASCII
UTF-16
UTF-16 Big Endian
UTF-16 Little Endian
UTF-16 Opposite Endian
UTF-16 Platform Endian
UTF-16,version=1
UTF-16,version=2
UTF-16BE,version=1
UTF-16LE,version=1
UTF-32
UTF-32 Big Endian
UTF-32 Little Endian
UTF-32 Opposite Endian
UTF-32 Platform Endian
UTF-7
UTF-8
UTF-8 (deprecated)
Windows 874
Windows 949 (korean)
x-iscii-be
x-iscii-de
x-iscii-gu
x-iscii-ka
x-iscii-ma
x-iscii-or
x-iscii-pa
x-iscii-ta
x-iscii-te
x-mac-ce
x-mac-cyrillic
x-mac-greek
x-mac-turkish
x11-compound-text

NONE and Zero

When the Item Subclass has a Number value and the Interpret as property has a Character value, the NONE and Zero properties are available. Expand the NONE or Zero property to specify an override data value to define the Special Value and Required on input properties.

- [**NONE > Special Value and Zero > Special Value**](#)
Enter a value in the Special Value property if the item is NONE or Zero.
 - [**NONE > Required on input and Zero > Required on input**](#)
Select a value from the Required on input property list.
-

NONE > Special Value and Zero > Special Value

Enter a value in the Special Value property if the item is NONE or Zero.

For example, if you specify * in the Special Value property for NONE, and a number object has the value *, it is interpreted as NONE.

If you enter a value in the Special Value property for NONE or Zero, enter the exact characters to be validated in the input data and built in the output data.

The maximum element length that is allowed for the Special Value property is 3000 bytes.

NONE > Required on input and Zero > Required on input

Select a value from the Required on input property list.

Yes

If the data object is NONE or Zero, the map uses the value in the Special Value property when it validates the item in the input.

No

Do not use the value in the Special Value property. No special values or input requirements are defined when data is NONE or Zero.

When the map builds the item in the output, the item might be either the value in the Special Value property or the default value for NONE or Zero. For example, if an item contains all pad characters and no actual data, the default value for NONE is used when that item is built in the output.

Places

The **Places** property allows you to specify the number of implied decimal places.

For item types defined with an **Item Subclass of Number**, a **Decimal** presentation, and a **Separator**, the **Places** property allows you to specify the minimum and maximum number of decimal places and whole number places. The minimum number of places must be less than or equal to the maximum number of places.

If the decimal number has no separator, use the **Places** property to specify the number of implied decimal places.

Text item subclass properties

Text item subclass properties can be interpreted with a **Character** or **Binary** value. Depending on the value, character or binary, the following are text item subclass properties:

- ["Interpret as Binary"](#) or ["Interpret as Character"](#)
 - ["Size \(content\)"](#)
 - ["Pad"](#)
 - ["Restrictions"](#)
 - ["National language"](#)
 - [Interpret as binary](#)
 - [Interpret as character](#)
 - [Size \(content\)](#)
 - [Pad](#)
 - [Restrictions](#)
 - [National language](#)
 - [NONE](#)
 - [Bidirectional](#)
-

Interpret as binary

Binary text items have content size and pad properties. Binary data is required to be sized or of a fixed size.

Binary number items interpret the data as a binary number or as a byte stream.

Items with an item subclass of **Text** can be interpreted as character or binary.

When **Interpret as** is set to **Binary**, the binary presentation options are Integer, Float, Packed, or BCD.

- [Binary integer presentation](#)
 - [Binary float presentation](#)
 - [Binary packed presentation](#)
 - [Binary BCD presentation](#)
-

Interpret as character

Character number items interpret the data as symbolic data. Symbolic data has the same meaning on different computers. For example, a comma is symbolic because it has the same meaning, regardless of the computer type.

Character text items can have content size and pad properties.

When the **Interpret as** value is defined as **Character**, the **Release** property is specific to the **Character** value and is *not* available as an option for a **Binary** value. For more information, see ["Release characters"](#).

When **Interpret as** is set to **Character**, integer, decimal, and zoned presentation options are available.

- [Character integer presentation](#)
- [Character decimal presentation](#)
- [Zoned character presentation](#)
- [Places > implied](#)

Size (content)

Each text or syntax object can be defined with the minimum and maximum sizes in bytes and/or characters. The product determines size for text or syntax objects according to the following rules.

- When the size of an object is in bytes only, size and validation are calculated according to the size limitations set for bytes.
- When the size of an object is in characters only, size and validation are calculated according to the size limitations set for characters.
- When an object has size values defined for both bytes and characters, the object size is determined by characters (based on any character size limitations). Bytes are used to verify that the size of the data did not exceed any byte constraints that were set.

When the content size of a text or syntax item is specified in bytes, initiators, terminators, release characters, and pad characters are excluded from the count.

Min

The minimum size of the data object. You can enter values for bytes and/or characters.
The default value is 0. The largest value allowed is 65535.

Max

The maximum size of the data object. You can enter values for bytes and/or characters.
The default value is 0. The largest value allowed is 65535.
When there is no maximum content size limitation, this value is not required.

During type tree analysis, checks are performed on byte and/or character Min and Max values according to the following methods:

- When values are present for both Min_Bytes and Min_Characters, the Min_Bytes value must be greater than or equal to MinCharacters.
- When values are present for both MaxBytes and MaxCharacters, MaxBytes must be greater than or equal to MaxCharacters.
- When no size limitations are present, no verification takes place.

Remember:

- Some character sets use two or more bytes per character.
- The trace file and audit log lengths are reported in bytes.

Pad

Pad implementation determines how the pad value is sized during validation and how the pad value is sized when written as output.

When the data value to be mapped to the target item is smaller than the minimum length of that item, pad characters can be used to pad the data to that minimum length. Input data can contain both content and pad characters. Output data is built according to the pad definitions of the types. Bytes are the default measurement for sizing, however, both byte and character sizing options are available.

Yes

Enables the **Pad** option. Allows the item to contain both content and pad characters.

No

All data is assumed to be content on input; no pad characters are built on output.

- [To specify a pad character](#)
- [Pad > value](#)
- [Pad > Padded to](#)
- [Padded to > length](#)
- [Padded to > SizedAs](#)
- [Padded to > CountsTowardMinContent](#)
- [Padded to > Allow Excess Trailing Pads](#)
- [Pad > justify](#)
- [Pad > apply.pad](#)
- [Pad > fill](#)

Restrictions

Restrictions of an item are the valid values of that item. When the **Interpret as** value is defined as **Character**, the **Restrictions** property is specific to the **Character** value and is not available as an option for a **Binary** value.

Depending on other **Item Subclass** settings, the **Restrictions** property can be set to **Value**, **Character**, or **Range**.

- [Restrictions > ignore case](#)
- [Restrictions > rule](#)

National language

The National language default value is Western. For initiator, terminator, and release character **Literal** values, you can optionally specify a "[Data language](#)".

When the **Interpret as** value is defined as **Character**, the **National language** property is specific to the **Character** value and is not available as an option for a **Binary** value.

- [National language > data language](#)
-

NONE

When the Item Subclass has a Text value and the Interpret as property has a Character or Binary value, the NONE property is available. Expand the NONE property to specify an override data value to define the Special Value and Required on input properties.

- [NONE > Special Value](#)

Enter a value in the Special Value property if the item is NONE.

- [NONE > Required on input](#)

Select a value from the Required on input property list.

NONE > Special Value

Enter a value in the Special Value property if the item is NONE.

For example, if you specify * in the Special Value property for NONE, and a text object has the value *, it is interpreted as NONE.

If you enter a value in the Special Value property for NONE, enter the exact characters to be validated in the input data and built in the output data.

The maximum element length that is allowed for the Special Value property is 3000 bytes.

NONE > Required on input

Select a value from the Required on input property list.

Yes

If the data object is NONE, the map uses the value in the Special Value property when it validates the item in the input.

No

Do not use the value in the Special Value property. No special values or input requirements are defined when data is NONE.

When the map builds the item in the output, the item might be either the value in the Special Value property, or the default value for NONE. For example, if an item contains all pad characters and no actual data, the default value for NONE is used when that item is built in the output.

Bidirectional

Use the Item subclass > Bidirectional property to define the data as bidirectional.

Certain languages that are read from right-to-left often contain numeric or other phrases that are read from left-to-right. This type of data is called "bidirectional" because it changes direction in the middle of a line.

The default setting is **No**.

To enable this setting, select **Yes**. When you enable this setting, additional options are available to further define the bidirectional data.

Text items

The Bidirectional subproperties for text item types provide options for symmetric swapping, ordering, orientation, and shaping.

Symmetric Swapping

Use this property to maintain the orientation of directional pairs of characters (such as parentheses, greater than/less than symbols, brackets, braces).

Ordering Scheme

This property pertains to the order of the data as stored in memory.

Logical, the default setting, indicates that the data is stored in memory from left-to-right, regardless of the orientation of the text. Using the **Visual** ordering scheme, the data is read and stored with the start character in the right-most position both visually and in memory.

Orientation

Orientation pertains to the direction of the text as read visually. The default direction is left-to-right (LTR), meaning the text object is read starting from the left. The direction can be right-to-left (RTL), meaning the object is read starting from the right. When you select RTL, the behavior of some text functions (such as LEFT, RIGHT, MID) can change.

Use the Contextual LTR and Contextual RTL settings when the orientation should be taken from the context of the data because the data contains "strong" characters that are either orientation left or orientation right. When no strong characters are present in the data, the orientation will be based on this setting.

For example, when you set the orientation to Contextual LTR and no strong characters are encountered (meaning the data is orientation-neutral), the data orientation will be considered left-to-right.

Text Shaping

Use this property to produce output in a different glyph (shape or bit pattern). The default setting is off, or **Unshaped**.

To enable text shaping, select **Shaped**. The output will be shaped according to the code page of the target object.

Character number items

For character number item types, you can additionally define a numeric shaping characteristic.

Numeric Shaping

Use this property to indicate whether the output of numeric values will have the shapes that are used in English (Arabic digits 0123456789) or the national numerical shapes.

The default setting is **Regular**, which means Arabic digits (0123456789). Otherwise, select **Arabic-Indic** (أ و و و و و و و و).

The Bidirectional property is not applicable to binary numbers.

Date & Time item subclass properties

The Date & Time properties provide the flexibility to define multiple combinations of date-time formats.

Date & Time items can be interpreted as either binary or character. (See sections "["Interpret as Binary"](#)" and "["Interpret as Character"](#)").

For binary **Date & Time** items, the **Presentation** property has two different values: **Packed** and **BCD**.

Packed

The **Packed** value is a number that has size, decimal places, and sign properties.

BCD

The **BCD** value is a binary coded decimal number that has size.

Other **Item Subclass** properties for **Date & Time** are displayed depending on the **Interpret as** setting (**Binary** or **Character**).

- [Date](#)
- [Time](#)
- [Format](#)
- [Time zones](#)
- [Time zone format string for XML](#)
- [Optional time segments of the time format string](#)
- [Date and time format examples](#)
- [NONE and Zero](#)

Date

For binary **Date & Time** items, use the **Date** property to define whether the date format is enabled.

Yes

Date format is enabled for this type. Expand the **Format** property to select the date format.

No

Date is not defined for this type.

- [Date > format](#)

Date > format

Select the date format that the data interpretation will be based on. For binary **Date & Time** items, the **Date > Format** property has four values.

CCYYDDD and **YYDDD** Julian date formats are supported.

- CCYYMMDD
- YYMMDD
- CCYYDDD
- YYDDD

Time

For binary **Date & Time** items, use the **Time** property to define whether the time format is enabled. Select one of the following options:

Yes

Time format is enabled for this type. Expand the **Format** property to select the time format.

No

Time is not defined for this type.

- [Time > format](#)

Time > format

Select the time format that the data interpretation will be based on. For binary **Date & Time** items, the **Time Format** property has several values. The following choices are available from the drop-down list:

- HH24MMSS
 - HH24MM
 - HH24:MM:SS
 - HH24:MM
 - Custom
-

Format

You can define a date-time format from within the type properties, except for native schema XML.

The output format for the native schema XML date/time field is always:

```
{CCYY-MM-DD}T{HH24:MM:SS.3-3+/-ZZ:ZZ}
```

To specify a different output format, use Xerces XML instead.

To define the Date & Time format:

1. Open the type properties.
2. In the Item Subclass **Format** property, click the browse button.
The Date Time dialog box is displayed.
3. Make your format selections and click OK.
You can use alphabetical characters as separators. In the example,
2001-04-02T10:32:59-0500, **T** is the separator.

Separators are limited to 60 characters.

Supported date formats:

- CCYYMMDD
- YYMMDD
- MMDDCCYY
- MMDDYY
- CCYYDDD
- YYDDD
- DDMMCCYY
- DDMMYY

Note: The DDMMCCYY and DDMMYY formats did not exist prior to version 6.5 of the Design Studio. If you had defined either one of these formats prior to 6.5 as a custom format, it will be recognized in 6.5 as the respective new format, instead of the previously defined custom format.

- [Use of format elements](#)
 - [Custom date format](#)
 - [Custom Time Format](#)
-

Use of format elements

If you select the same format element more than once for the same item, that item may not be validated separately. If you specify **MON** twice, the day of the month is not validated separately for both months. For example, if the date format element **MON** is used twice for a single item, with the following format string:

MON D-MON D

Suppose the data is this:

Feb 28-Mar 31

28 is not validated for the month of February.

Some combinations of reserved words are invalid. A separator must follow reserved words representing a variable number of digits (**D** and **M** for date) if data follows.

For example:

- D/M/CCYY is valid.
- CCYYM/D is valid.
- DMCCYY is invalid.

See the Design Studio Introduction documentation for a list of reserve words and symbols.

Custom date format

After choosing **Custom** from the date format drop-down list, click browse. The Date Format dialog box is displayed.

You can only use non-alphanumeric characters (excluding the {, } and [,] non-alphanumeric characters) as separators in the custom Date Format dialog box.

The following table provides examples of date formats.

Date Format	Description	Example
CCYY	4-digit Century + Year	1999
YY	2-digit Year (00-99)	99
MM	2-digit Month (01-12)	12
M	1- or 2-digit Month (1-12)	8
MON	3-character Month (Jan to Dec)	JAN
MONTH	Full name of Month	January
DDD	3-digit Day of year (001-366)	32
DD	2-digit Day of Month (01-31)	31
D	1- or 2-digit Day of Month (1-31)	7
DY	3-character Day of Week (Sun-Sat)	Fri
DAY	Full Name of Day of Week (Sunday-Saturday)	Friday
WW	1- or 2-digit Week of Year	13
Qn	Quarter of Year (Q1-Q4)	Q2
Custom	User defined custom date format	

After you define and save a custom format, the custom format string is displayed in the Properties window. For example, the custom date format **CCYYMMDD**, and the custom time format **HH24MMSS** display as:

{CCYYMMDD}{HH24MMSS}

Custom Time Format

After selecting **Custom** from the time format list, click browse. The Time Format window is displayed.

You can only use non-alphanumeric characters as separators, excluding the {, } and [,] non-alphanumeric characters, in the Time Format window.

The following table provides time format examples.

Time Format	Description	Example
HH24	2-digit hour in 24 hour format (00-23)	23
H24	1- or 2-digit hour in 24 hour format (0-23)	11
HH12	2-digit hour in 12 hour format (00-12)	08
H12	1- or 2-digit hour in 12 hour format (0-12)	8
MM	2-digit minute (00-59)	09
M	1- or 2-digit minute (0-59)	9
SS	2-digit second (00-59)	05
S	1- or 2-digit second (0-59)	5
AM/PM	Meridian (AM/PM)	AM
ZZZ	A time zone abbreviation. See " Time Zones " for a list of supported time zones.	EST
+/-ZZZZ	Hours and minutes before or after the Greenwich Mean Time (GMT). GMT is now referred to as Coordinated Universal Time (UTC).	+0500
+/-ZZ:ZZ	4-digit time where the format is a 2-digit hour and 2-digit minute, separated by a colon.	+05:00
+/-ZZ[;ZZ]	4-digit time where the format is a 2-digit hour and an optional 2-digit minute, separated by colon.	+05:00
+/-ZZ[ZZ]	4-digit time where the format is a 2-digit hour and an optional 2-digit minute.	+0500
TZD	4-digit time where the format is a 2-digit hour and 2-digit minute, separated by a colon. Z on output, if value is +/-00:00.	+05:00

After you define and save a custom format, the custom format string is displayed in the Properties window. For example, the custom date format CCYYMMDD, and the custom time format HH24MMSS display as:

{CCYYMMDD}{HH24MMSS}

Time zones

The following table lists the supported time zones.

Time Zone	Abbreviation	Hours before Greenwich Mean Time	Hours ahead of Greenwich Mean Time
Greenwich Mean Time (Zulu)	GMT	0	0
West African Time	WAT	-1	+23
Azores Time	AT	-2	+22
No name; Brasilia Time	###	-3	+21
Atlantic Standard Time	AST	-4	+20
Eastern Standard Time	EST	-5	+19
Central Standard Time	CST	-6	+18
Mountain Pacific Time	MST	-7	+17
Pacific Standard Time	PST	-8	+16
Yukon Standard Time	YST	-9	+15
Hawaii Standard Time	HST	-10	+14
Nome Time	NT	-11	+13
New Zealand Time	NZT	-12	+12
No name; no location	###	-13	+11

Time Zone	Abbreviation	Hours before Greenwich Mean Time	Hours ahead of Greenwich Mean Time
Guam Standard Time	GST	-14	+10
Japan Standard Time	JST	-15	+9
China Coast Time	CCT	-16	+8
West Australia Time	WAT	-17	+7
Zulu + 6 (Russia zone 6)	ZP6	-18	+6
Zulu + 5 (Russia zone 5)	ZP5	-19	+5
Zulu + 4 (Russia zone 4)	ZP4	-20	+4
Baghdad Time	BT	-21	+3
Eastern European Time	EET	-22	+2
Central European Time	CET	-23	+1

Time zone format string for XML

Note: The output format for the native schema XML date/time field is always:

{CCYY-MM-DD}T{HH24:MM:SS.3-3+/-ZZ:ZZ}

To specify a different output format, use Xerces XML.

The following time zone strings support the specification of a single character Z to indicate Coordinated Universal Time (UTC), as described by the World Wide Web Consortium (www.w3c.org) and the ISO 8601 standard:

- +/-ZZZZ
- +/-ZZ:ZZ
- TZD

When the +/-ZZ:ZZ or +/-ZZZZ format string is specified, the data validation process works in the following manner:

- Valid data that corresponds to this format string will contain either a character literal Z (representing UTC) or a zone in the appropriate format: +/-ZZ:ZZ or +/-ZZZZ, as specified.
- If a character literal Z is present, it shall be interpreted, and treated at mapping time, as UTC, which is equivalent to +00:00 or +0000.

When the TZD format string is specified, the data validation and output generation processes work in the following manner:

- Valid data that corresponds to this format string will contain either a character literal Z, representing UTC, or a zone, in the +/-ZZ:ZZ format.
- If a character literal Z is present, it shall be interpreted, and treated at mapping time, as UTC, which is equivalent to +00:00.
- If the value is +/-00:00, it generates Z in the output. Otherwise, it generates the output in +/-ZZ:ZZ format.

The following time zone strings support the omission of the minute portion of the difference from UTC:

- +/-ZZ[:ZZ]
- +/-ZZ[ZZ]

Optional time segments of the time format string

This section discusses how you can specify a portion of a time-format string to be optional. For example, you can specify that either the *time zone* portion or the *hours, minutes, seconds, fractional seconds, Meridian* portion is optional.

This flexibility is also applicable to time-format strings used in functions or in type tree scripts imported by the Type Tree Maker.

Only the time portion **or** the zone portion can be optional-not both.

To specify a portion of the time-format string as optional:

1. The following procedure assumes that the **Item Subclass** property of your type tree is **Date & Time**.
2. From within the type tree properties, navigate to Item Subclass-> Format.
3. Click the browse button to open the Date Time dialog.
4. From a Format field that is set to **Time**, go to the field directly below it and choose Custom from the drop-down menu.
5. Next to the Format field with **Custom** selected, click the browse button. The Time Format dialog is displayed.
6. Choose from the following tasks:
 - To specify the *hours, minutes, seconds, fractional seconds, and Meridian*-portion of the time format string as optional, enable the check box on the far left side of the dialog.
 - To specify the *time zone* portion of the time format string as optional, enable the check box.
 Only one segment of the time format string can be specified as optional.

Date and time format examples

The following list includes examples of popular standard date and time formats:

Format

Description

X12 EDI

Fractional seconds format: **HH24MM[SS[0-2]]**.

SWIFT

Time Zone Designator or UTC Designator format: **HH24MSS [TZD]**.

TimeStamp format: **CCYY-MM-DDTHH:MM[SS[.0-6]] [TZD]**, which is a combination of date and time.

HL7

Time format: **HH24MM[SS[.0-6]] [+/-ZZZZ]**.

TimeStamp format: **CCYYMMDDHHMM[SS[.0-6]] [+/-ZZZZ]**, which is a combination of date and time.

SAP

Time format: **HHMMSS**.

ODBC

Time format: **HH:MM:SS [.0-9]**, which is the form most commonly used in **ODBC** mapping. The fractional part is optional and not often used.

NONE and Zero

When the Item Subclass has a Date & Time value and the Interpret as property has a Character value, the NONE and Zero properties are available. Expand the NONE or Zero property to specify an override data value to define the Special Value and Required on input properties.

- [NONE > Special Value and Zero > Special Value](#)

Enter a value in the Special Value property if the item is NONE or Zero.

- [NONE > Required on input and Zero > Required on input](#)

Select a value from the Required on input property list.

NONE > Special Value and Zero > Special Value

Enter a value in the Special Value property if the item is NONE or Zero.

For example, if you specify * in the Special Value property for NONE, and a data and time object has the value *, it is interpreted as NONE.

If you enter a value in the Special Value property for NONE or Zero, enter the exact characters to be validated in the input data and built in the output data.

The maximum element length that is allowed for the Special Value property is 3000 bytes.

NONE > Required on input and Zero > Required on input

Select a value from the Required on input property list.

Yes

If the data object is NONE or Zero, the map uses the value in the Special Value property when it validates the item in the input.

No

Do not use the value in the Special Value property. No special values or input requirements are defined when data is NONE or Zero.

When the map builds the item in the output, the item might be either the value in the Special Value property or the default value for NONE or Zero. For example, if an item contains all pad characters and no actual data, the default value for NONE is used when that item is built in the output.

Syntax item subclass properties

Syntax objects are characters that precede, separate, or follow a particular data object. Item types with an **Item Subclass of Syntax** are defined and used to specify delimiters, initiators, terminators, and release characters. Syntax objects with variable values are defined as the syntax object's **Variable** property. Syntax objects appearing as actual data to be mapped as output data are defined as components of a type.

- [Syntax objects with variable values](#)
- [Example of variable syntax object as an item type](#)
- [Syntax objects as data](#)
- [Example of syntax objects as components of a type](#)
- [Delimiter > find](#)

Syntax objects with variable values

About this task

Using a syntax item to specify delimiters, initiators, terminators, and release characters is required when the value of syntax objects (delimiters, initiators, terminators, and release characters) may vary.

The restrictions of syntax items define the variable literal values.

To define a variable syntax object:

Procedure

1. Create an item type with an **Item Subclass** of **Syntax** to represent the syntax object.
2. Define the restrictions for the syntax item type.
3. For the item or group type with the variable syntax object, for the **Variable** delimiter, initiator, terminator, and release character, select the item from the **Item** drop-down list in the Properties view.
4. For the **Find** property, select Yes.
Only item types defined with an **Item Subclass** of **Syntax** appear in the drop-down list.

Related tasks

- [Defining item restrictions](#)
-

Example of variable syntax object as an item type

In this example, the group type **Header** has variable delimiter values of: , * + ~

1. Create an item type with the **Item Subclass** of **Syntax** and the name **Delimiter**. (The name can be any valid type name, but naming this item **Delimiter** is useful).
2. To define the item restrictions, double-click the **Delimiter** type. The item view is displayed.
3. Define the restrictions in the **Include** column.
4. In the Properties window, select Syntax for the **Item Subclass** property.
5. Open the properties for the delimited group type **Header**.
6. For the **Syntax** property, choose **Delimited**.
7. For the **Delimiter** property, choose **Variable**.
8. Expand the **Delimiter** property.
9. Define the Delimiter._{Default} literal value.
10. For the variable Delimiter Item property, choose the item type **Delimiter** from the drop-down list.
11. Indicate that the variable delimiter should be determined for each occurrence of the object by selecting Yes for the Delimiter._{Find} property.

Syntax objects as data

The value of a syntax object (delimiters, initiators, terminators, and release characters) may appear as actual data that can be mapped. Syntax objects that appear as actual data are defined as components of a type.

Example of syntax objects as components of a type

The following is an example of defining a syntax object as a component of a group type. The following data represents a message received from multiple departments.

Each message is made up of segments. Each department uses different segment delimiters and terminators. The message format specifies that the delimiter and terminator are the first two bytes of the message, followed by the actual data.

To define syntax objects for the delimiter and terminator:

1. Define two separate syntax objects with an **Item Subclass** of **Syntax**, one for the message delimiter and one for the message terminator. Appropriate type names might be **SegmentDelimiter** and **SegmentTerminator**.
2. Define the possible message delimiter values as restrictions of the **SegmentDelimiter** item type. Define the possible message terminator values as restrictions of the **SegmentTerminator** item type.
3. Define these item types as the first two components of the group type **Message**.
4. Expand the **Delimiter** property.
5. For the variable Delimiter._{Item} property, select SegmentDelimiter from the drop-down list.
6. For the variable Terminator._{Item} property, select SegmentTerminator from the drop-down list.

During the data validation process, the component **SegmentDelimiter** appears in the data with a value of *. The group type **Segment** has a variable delimiter specified as the item type **SegmentDelimiter**, with the value *. Therefore, it is understood that the segment has * as the delimiter.

When the value of a syntax object appears as actual data, it can be mapped as data. For example, source data may use different delimiters from several different sources. Acknowledgments of this data must be sent back to the source using the delimiter used in the original data. This data can be mapped from the input to the output if these syntax objects are defined as components within the data.

Delimiter > find

When syntax objects are **Variable**, the **Find** property must be defined. The **Find** property determines whether the value of syntax object is set on each occurrence or whether the current setting (or default value) is used.

When the **Initiator**, **Terminator**, **Release**, or **Delimiter** property value is **Variable**, select one of the following from the drop-down list in the **Value** column:

Yes

Determine the value of the syntax object each time an occurrence of that type is found. After the value of that syntax object is found, that particular value is used until it is reset by another **Find** or by the occurrence of that syntax item as a component.

INPUT: The value of the object in the input data is determined by the location in the data stream and the restrictions of the syntax item.

OUTPUT: The default value is used for building the syntax object in the output data.

No

The variable syntax object is defined as the current value or, if it is not set, as the default value.

INPUT: If the syntax object is not encountered in the data stream, the value of the syntax object is the default value.

OUTPUT: If the value of the syntax item has *not* been previously set, the default value is used.

Group properties

Group properties include the group's **Subclass** and **Format**, which describes how to distinguish one component of that group from another component of that group.

- [Group subclass](#)
- [Sequence group formats](#)
- [Explicit format](#)
- [Implicit format](#)
- [Specifying a delimiter](#)
- [Defining exclude characters](#)

During map run time, the data validation process uses the Exclude Character List group property to define a set of characters that are not allowed to exist in the data stream as data content.

Group subclass

Group types have a subclass of **Sequence**, **Choice**, or **Unordered**.

Property	Description
Sequence	A partially-ordered or sequenced group of data objects. Each component of a Sequence group is validated sequentially.
Choice	Choice groups provide the ability to define a selection from a set of components like a multiple-choice question on a test. A Choice group is valid when the data matches one of the components of the choice group. Validation of a Choice group is attempted in the order of the components until a single component is validated. If the Choice group has an initiator, the initiator is validated first.
Unordered	An unordered group has one or more components. Unordered groups can only have an Implicit format property, with the same syntax options as sequence groups: None or Delimited .

- [Properties of group subclasses](#)
- [Choice group components](#)
- [Unordered group components](#)

Properties of group subclasses

The distinction between Sequence and Choice group subclasses is that Choice groups have no **Partition** or **Format** properties.

Type syntax properties such as **Initiator**, **Terminator**, **Release**, and **Empty** can be applied to **Choice** groups. If a release character is defined, by default it applies to the **Terminator**.

Choice group components

A Choice group can have both items and groups as components. A partitioned Sequence group can only have group subtypes.

Components of a Choice group must be distinguishable from each other. The components of a Choice group cannot have a component range other than (1:1). Only one component of a Choice group built in the output data.

A Choice group data type is only one of the group components. For example, the data type **Record** is a group type with a **Group Subclass** of **Choice**. The group type **Record** has three components: **Order**, **Invoice**, and **Sales**. The data validation of **Record** will be only one of the components: **Order**, **Invoice**, or **Sales**. For this reason, the components of a Choice group must have a component range of (1:1).

Unordered group components

An unordered group has one or more components that can appear in the data stream in any order. They allow many SWIFT and FIX message types, for example, to be defined in a more natural way.

Unordered groups have no partitioned property. They have **Implicit** format properties with the same syntax options as sequence groups: **None** and **Delimited**.

When a group is defined as **Unordered**, any component can appear in the data stream. A component can be an item or a group.

Unordered group components have a range property. For example, if the unordered group, **A**, has the following component list:

B(1:S)
C

D (S)

then **A** must have one **C**, at least one **B**, and possibly some **D**s. They could appear in any order. For example, data for **A** could have the pattern: **CDBBDDD** or **BBBDDCDB**.

Component rules of an unordered group cannot reference other components of the same group. They can only reference the component to which the rule refers and the objects contained in that component.

Sequence group formats

A sequence group has either an **Explicit** or **Implicit** format.

Explicit

The explicit format relies on syntax to separate components. Each component can be identified by its position or by a delimiter in the data. Delimiters appear for missing components.

Implicit

The implicit format relies on the properties of the component types. The format is not fixed. If delimiters separate components, they do not appear for missing components.

For example, if each component of a fixed group has a fixed size, the component is distinguished from the next component by its position in the data. Or, a group may have delimiters that appear for missing components. In these cases, the format is apparent; the group has an explicit format.

If a group does not have an explicit format, it has an implicit format. An implicit format relies on the properties of the component types. In this example, the components make some pattern in the data and it is possible to distinguish between them, but the format is not fixed and if delimiters separate components, they do not appear for missing components.

When deciding what format a group has, it may help to ask first whether it is clear where one component ends and another begins. Generally, a group has an explicit format if the position of each component in the data stream is always the same or if a delimiter always marks the place for each component.

Explicit format

To specify that a group has an explicit format, choose **Explicit** for the **Format** property. Select a setting for the **Track** property, and choose the group's syntax: **Fixed** or **Delimited**.

- [Track](#)
- [Fixed syntax](#)
- [Explicit delimited syntax](#)

Track

The **Track** property indicates whether the system should track only the components that have content or all components, including those that do not have content. The settings are **Content** and **Places**.

For example, if a group **StudyGroup** has the component **Name(s)**, and Places is specified for the Track property, any empty occurrences of **Name** are tracked.

Suppose **Name** has an * initiator and a space pad character. This is the data for **StudyGroup**:

*Carolyn * *Stuart *Margaret

Notice that the second **Name** is missing.

If Places is specified for the Track property, and you want to map the third **Name**, the empty **Name** would be counted as an occurrence, so the third **Name** would be Stuart. However, if Content is specified for the Track property, the empty **Name** would not be counted as an occurrence because it does not have content and the third **Name** would be Margaret.

Fixed syntax

If a group data object is always the same size (in bytes), it has a fixed syntax. For example, a record that is always 160 bytes has a fixed syntax.

Each component of a fixed group must be fixed. If you break down a fixed group, it ultimately consists of items that are fixed. Each is padded to a fixed size or its minimum and maximum content size are equal. Do not specify the size of a fixed group. The size is automatically calculated based on the size of the group's components.

- [Guidelines for defining a fixed group](#)

Guidelines for defining a fixed group

- Each component must be a group with a fixed syntax or a fixed item. The item is padded to a fixed size or its minimum and maximum content size are equal.
- Each component must have a specified range maximum. The maximum cannot be s.
- If a component range minimum is not equal to the maximum, content is not required for optional component occurrences. For example, if a component is the item **ShippingAddress** (0:1) and **ShippingAddress** has a minimum content size of two characters, data may either contain: 1) all pad characters, or 2) if content is in the

data stream, there must be a minimum of two characters for a **ShippingAddress**. If a component is a group, no content is required for any of the items contained in an optional occurrence of that component.

Explicit delimited syntax

An explicit-format group with a delimited syntax is one whose components are separated by a delimiter and the delimiter appears as a placeholder even when a component has no content.

About delimiters and explicit-format groups:

- The only time a delimiter can be missing is when all components following the delimiter are optional and there is no data for the optional components.
- A delimiter is a character or series of characters that separates data objects.
- A delimiter cannot be longer than 500 bytes.
- The delimiter of a group appears inside the group, separating its components. When a group is delimited, that indicates something about the components of the group. The delimiter inside a group is delimiting the components.

Example

The group **Employee** has an explicit-delimited format because a comma delimiter appears between **Employee's** components, the items that make up the **Employee** object. In addition, the delimiter is displayed when a component is missing and there is data for components following it.

The components of **Employee** are the items **ID#**, **Name**, **Department**, **Address**, and **Age**.

- [Delimiter](#)

Delimiter

Use the **Delimiter** property to specify the value and location of the delimiter. For specific information on specifying the delimiter, see "[Specifying a Delimiter](#)".

In an explicit-delimited group, if the delimiter is whitespace <WSP>, the delimiter is interpreted as one byte long; each <WSP> character is interpreted as another delimiter. In the output, a <WSP> delimiter is one space.

The limit for <WSP> is 512 bytes.

Implicit format

In a group with an implicit format, the components are distinguishable not by delimiter or position, but by their pattern; something in the definition of the component types themselves. The group has no syntax property that distinguishes one component from another.

For example, the type **File** consists of **Record**(s). <CR><LF> appears at the end of each **Record**. It has been defined as the terminator of **Record**. **File**, however, has no syntax of its own. The **Record** terminator distinguishes one **Record** from another.

To specify that a group has an implicit format, choose **Implicit** for the **Format** property. Optionally, define a comment type, and choose the group's syntax: **Delimited** or **None**.

- [Floating component](#)
- [Implicit whitespace syntax](#)
- [Implicit delimited syntax](#)
- [No syntax](#)
- [Distinguishable components of an implicit group](#)

Floating component

The floating component represents an object that may appear after any component of the group.

An implicit group can have a floating component; an explicit group cannot. If the group is prefix or infix delimited, the floating component is displayed before the delimiter. If the group is postfix delimited, the floating component is displayed after the delimiter.

A floating component can be an optional component that may appear after any other component. However, it is not included in the component list because it does not appear at a specific location.

If a group has a floating component, a component must be distinguishable from a floating component. For example, components and floating component could start with different initiators.

Note: When a floating component appears in the input data, it is validated during mapping. If there are floating components in your output data, define them as actual components of the output.

A floating component can appear after the initiator (when the type has an initiator), after each component, or both. For EDI and other existing floating components, trees will be converted as "after each component".

A floating component can be specified for implicit sequence group, choice group, and unordered group definitions.

A floating component provides additional flexibility to support XML data. With the floating component property, XML element groups would have a floating component defined as an unordered group of XML comments, XML processing instructions, and/or white space.

The XML DTD Importer, for example, uses the floating component property for XML elements whose content contains other elements. For such XML element definitions, a floating component can be a choice of XML comments and/or XML processing instructions.

To define a type as a floating component:

1. In the Properties window expand the **Format** property.
2. With current focus in the Floating Component Value field, define the value for the floating component by pressing **Alt** and dragging the item from the tree into the Floating Component Value field.

Implicit whitespace syntax

Select WhiteSpace for the **Component Syntax** value when white spaces are not allowed in the data. When you select the **WhiteSpace** option, define the **Build As** and **Character Set** properties.

Note: Helpful for XML data.

- [Build as](#)
- [Character set](#)

Build as

Enter the characters that will replace white spaces.

The Build As property is only available when the Component Syntax property is defined as WhiteSpace.

Character set

Select a character set for the component from the drop-down list.

The default value is Native, where the literal values use the native character set of the computer.

[Supported character sets \(code pages\)](#)

Implicit delimited syntax

If a delimiter separates the components of a group, but the delimiter does not appear when a component is missing, the group has an implicit format with a delimited syntax.

In certain data, the delimiter may not be a placeholder.

- [Delimiter](#)

Delimiter

Use the **Delimiter** property to specify the value and location of the delimiter. For specific information on specifying the delimiter, see "[Specifying a Delimiter](#)".

In an implicit-delimited group, when the delimiter is whitespace <WSP>, all contiguous <WSP> characters are treated as one delimiter. In the output, a <WSP> delimiter is built as one space.

The limit for <WSP> is 512 bytes.

No syntax

To specify a group with an implicit format that has no syntax, choose **None** for the **Component Syntax** value.

Distinguishable components of an implicit group

Each component of an implicit group needs to be recognizable. If either of two different components appears at the same place in a data stream, there must be a distinguishable difference between one component and the other.

Sometimes components of an implicit group may be distinguished because there is something in the data that distinguishes them.

After the first item **Line** of the **Form**, the next data object could be another item **Line**. Or, it could be a **Trailer Line**. When looking at a particular **Line**, there is something in the data that identifies that **Line** either as a **Header Line**, an item **Line**, or a **Trailer Line**. The type **Line** has been partitioned to distinguish between the different kinds of **Lines**.

Related concepts

- [Partitioning](#)

Related information

- [Distinguishable objects](#)
-

Specifying a delimiter

For the **Delimiter** property, specify whether the delimiter is a literal or a variable value.

- [Literal](#)
 - [Variable](#)
 - [Location](#)
 - [Delimiter value appears as data](#)
 - [Allow Excess Trailing Delimiters](#)
-

Literal

If the delimiter is a constant value, enter the delimiter value in the **Value** field. To enter a non-printable value, click the browse button and select a value from the Symbols dialog box.

- [National language](#)
 - [Data language](#)
-

National language

The National language default value is Western. For initiator, terminator, and release character **Literal** values, you can optionally specify a "[Data language](#)".

When the **Interpret as** value is defined as **Character**, the **National language** property is specific to the **Character** value and is not available as an option for a **Binary** value.

- [National language > data language](#)
-

Data language

Use the National language...> Data language property to define the data language or character set.

[Supported code pages](#)

Variable

Sometimes you do not know what delimiter is used in the data source, especially if you receive that data from an outside source. However, you know all of the possible values that the delimiter could be. You can create an item to represent this delimiter and specify all of the possible values as restrictions of that item. The value of the delimiter in the data is found.

To specify an item as a Variable Delimiter:

1. In the Properties view for the delimited group, click in the Delimiter...>Item value field.
 2. Press Alt and drag the item from the type tree editor into the **Item** field.
-

Location

The **Location** property specifies the location of the delimiter with respect to the components. The options are prefix (the delimiter appears before the component), postfix (the delimiter appears after the component), and infix (the delimiter appears between components).

The following table explains each option.

Property	Description	Example
----------	-------------	---------

Property	Description	Example
Prefix	Before each component. Before each member of a component in a series.	*a*a*b*c
Postfix	After each component. After each member of a component in a series.	a*a*b*c*
Infix	Between components. Between members of a component in a series.	a*a*b*c

Delimiter value appears as data

Delimiters do not appear as part of the actual data. For example, if the delimiter is specified as a comma, the comma in the following text item would be considered a delimiter, rather than part of the data itself:

Tom Smith, Jr.

If the data does contain the delimiter value, there are ways to format the data so that both the data and the delimiter are distinguishable. For example, text items can be enclosed in quotation marks. Or, a release character may be used. For information about release characters, see ["Release Characters"](#).

Allow Excess Trailing Delimiters

The type tree validation process allows data that contains excess trailing delimiters on sequence groups.

You can change the default setting for the Allow Excess Trailing Delimiters property to No so that the type tree validation process, instead, fails data that contains excess trailing delimiters.

The type tree validation process interprets a delimited sequence group that contains optional components at the end of its sequence, as having excess trailing delimiters.

For example, GROUPA contains the following components: COMPA (mandatory), COMPB (optional), COMPC (optional), and COMPD (optional). The group is infix-delimited by a tilde. The following data stream examples are valid for GROUPA:

```
A~B~C~D
A~B~C~
A~B~~
A~~~
```

For this example, you want the type tree validation process to disallow all of the data stream examples that contain unnecessary or excess trailing delimiters. To disallow the excess trailing delimiters, you can change the setting for the Allow Excess Trailing Delimiters property to No. The result is that the validation process fails all of the examples except the first one.

Defining exclude characters

During map run time, the data validation process uses the Exclude Character List group property to define a set of characters that are not allowed to exist in the data stream as data content.

The data validation process allows these excluded characters to exist in the data stream as syntax, such as delimiters, if they are not being used as data content.

The group exclude characters might be derived from previously parsed values in the data stream or might be defined as literal values.

When you define exclude characters at the group level, the data validation process handles these specified characters as if they were in-scope data syntax.

For example, you have a type tree which defines two groups, GROUPA and GROUPB. The tree defines the asterisk (*) character as a group delimiter for GROUPA. It defines the at-sign (@) character as a group delimiter for GROUPB. The tree also defines the FILE group, which contains both GROUPA records and GROUPB records.

The data stream contains content corresponding to the FILE group, which is composed of a series of both GROUPA and GROUPB records.

When a map begins to run, it parses and validates this data stream. Since the asterisk is the delimiter of GROUPA, the data content of GROUPA records is only allowed to contain asterisk characters that are escaped. If the data content contains non-escaped asterisk characters, those GROUPA records fail data validation. Similarly, since the at-sign is the delimiter of GROUPB, the data content of GROUPB records is not allowed to contain non-escaped at-sign characters. If it does, those GROUPB records fail data validation.

The type tree defines the delimiters, or markup, for the GROUPA and GROUPB records. Asterisk characters are considered to be in-scope markup while the data validation process handles GROUPA records. At-sign characters are considered to be in-scope markup while the data validation process handles GROUPB records.

Conversely, GROUPA data content is allowed to contain at-sign characters, as the GROUPB at-sign delimiter does not apply to GROUPA records. Therefore, at-sign characters are considered out-of-scope markup for GROUPA records. GROUPB data content is allowed to contain asterisk characters, as the GROUPA asterisk delimiter does not apply to GROUPB records. Therefore, asterisk characters are considered out-of-scope markup for GROUPB records.

You might require the data validation process to exclude asterisk and at-sign characters in all of the data content for both GROUPA and GROUPB records, except when those characters are used as delimiters.

If so, use the Exclude Character List property to specify the asterisk and at-sign literal values at the FILE group level. When the validation process evaluates the data, it throws an exception if there is an asterisk (*) or an at-sign (@) character in the data content within the FILE group. The asterisk and at-sign characters are still valid if they are used as group delimiters; they are invalid if they are used in data content.

Performance tip: The Exclude Character List property causes the map validation process to use more resources as it searches for each of the exclude group characters in every data element under the group on which it is defined.

Open the Exclude Character List window by clicking the Value field on the Exclude Character List property, and then clicking the button that appears at the end of the field. Specify the characters in the list as literal characters or syntax items. You can use the Add Literal, Add Syntax Item, Edit, Remove, and Remove All functions to add literal characters and syntax items to the list, edit them, and remove them from the list.

- [Adding literal](#)
- [Adding syntax type](#)
- [Escaping an excluded character in the data content](#)

Select Use release character when you configure the exclude character list to enable the release character to escape excluded characters. When enabled, a release character that precedes an excluded character in data content escapes the excluded character, and the data content passes validation. Likewise, Transformation Extender precedes an excluded character with the release character when building the output stream.

Adding literal

About this task

Use this function to add literal exclude characters to the exclude character list.

Procedure

To add literal exclude characters to the exclude character list:

1. Open the Exclude Character List window by clicking the Value field on the Exclude Character List property, and then clicking the button that appears at the end of the field.
2. In the Exclude Characters List window, click Add literal.
The Add literal window opens.
3. Supply the literal values in the following ways:
 - Type simple text directly in the Value field.
This simple text can be any text or any of the IBM Integration Platform symbols from the list of symbols.
 - Insert IBM Integration Platform symbols that you select from the list of symbols. After you select a symbol, click Insert.
Design Server inserts the literal you selected from exclude characters list into the Value field.
4. Continue to type text and insert IBM Integration Platform symbols in the Value field, as required.
Be sure to place the cursor where you need to type or have Design Server insert the symbol. You can add, insert, and replace new characters.
5. Click OK.
The Add literal window closes and the Exclude Characters List window is redisplayed.
6. Click OK.
The Exclude Characters List window closes and the Properties view is redisplayed.

Results

You can see the Literal property that you added, along with the literal value or values that you supplied, in a new row under the Exclude Character List property. Every time you use the Add Literal function, the added Literal property appears in a new row in the Exclude Character List property on the group. You can edit a Literal property in the list by using the Add Literal, Edit, and Remove functions in the Exclude Characters List window.

Adding syntax type

About this task

Use this function to add syntax type exclude characters to the exclude character list.

Procedure

To add syntax type exclude characters to the exclude character list:

1. Open the Exclude Character List window by clicking the Value field on the Exclude Character List property, and then clicking the button that appears at the end of the field.
2. In the Exclude Characters List window, click Add syntax type.
The Add syntax type window opens.
3. Select syntax type item.
4. Click OK.
The Add syntax type window closes and the Exclude Characters List window is redisplayed.
5. Click OK.
The Exclude Characters List window closes and the Properties view is redisplayed.

Results

You can see the Syntax Item property that you added, along with the syntax item value or values that you supplied, in a new row under the Exclude Character List property. Every time you use the Add syntax type function, the added Syntax Item property appears in a new row in the Exclude Character List property on the group. You can edit a Syntax Item property in the list by using the Add syntax type, Edit, and Remove functions in the Exclude Characters List window.

Escaping an excluded character in the data content

Select Use release character when you configure the exclude character list to enable the release character to escape excluded characters. When enabled, a release character that precedes an excluded character in data content escapes the excluded character, and the data content passes validation. Likewise, Transformation Extender precedes an excluded character with the release character when building the output stream.

Related tasks

- [Adding literal](#)
 - [Adding syntax type](#)
-

XML properties in the type tree

When you open a type tree in the Type Designer after you imported it from a DTD or XML Schema, the XML-specific properties are displayed as read-only fields.

For each type in the resulting type tree, the Intent property is either General (indicating non-XML) or XML. All category types that are created during the import process are considered non-XML properties. All group and item types that are created during the import process are considered XML properties.

Any types that you manually add to the XML type tree are considered non-XML and have General intent. In such a case where there are both XML and non-XML types present (such as EDI data), the appropriate method of validation is determined automatically.

See the Type Tree Maker documentation for information about creating type trees by importing XML Schemas and DTDs.

See ["Utilities for XML"](#) for information about tools that can assist you with upgrading your XML type trees and maps to this version of IBM Transformation Extender.

- [Support for XML constructs](#)
 - [XML schema datatypes](#)
 - [Identity constraints](#)
 - [Namespaces](#)
 - [Element type declarations](#)
 - [Element and attribute wildcards](#)
-

Related concepts

- [Basic type properties](#)
-

Support for XML constructs

IBM Transformation Extender validation supports the following XML constructs.

Character Data

During validation, character data is mapped by the XML parser to IBM Transformation Extender types. This data includes both parsed character data (PCDATA) and unparsed character data (CDATA).

Comments and Processor Instructions

XML comments and processing instructions (PI) are mapped to floating components in type trees.

Namespaces

The XML Schema importer supports the specification of arbitrary prefixes for namespaces declared in the input grammar.

XSDL Hints

The xsi:schemaLocation and xsi:noNamespaceSchemaLocation attributes are collectively known as XML Schema Definition Language (XSDL) hints. These attributes specify the location of the XML Schema(s) that the XML parser uses for validation.

An XML Schema allows either or both of these attributes to display within any element tag. But the XML parser respects the location values only when the XSDL hint is specified in the root element of the schema.

The Doc group type of type trees that are created with the XML Schema or XML DTD importer contains the location of the Xerces DTD or Xerces XML Schema that is used for Xerces XML validation. The Intent->Validate As->Location property can be modified. However, use the Schema->Type->Metadata card setting in the map to change the location of the DTD or Schema if needed.

Empty Elements

Only one set of initiators and terminators for the Empty element is required for validation.

Nillable Elements

A nillable element can have one of three states: absent, present with content, or present with nil content. Both the DTD and XML Schema allow the definition of optional elements in the instance document. Additionally, the XML Schema allows nillable elements where the content can be empty when it contains an xsi:nil attribute with a value of "true", despite the fact the element's content is mandatory.

The XML Schema Importer can create the content of a nillable element within a group with a range of (0:1), which makes the element content optional.

Mixed Content

When parsing elements with a mixed content model (containing both character data and child elements), the DTD and XML Schema importers generate text items within the mixed content (instead of character sequences).

Regular Expressions

An XML Schema allows the restriction of the values of simple types that are based on regular expressions or pattern facets. During XML validation, the parser enforces the pattern facet. The XML Schema Importer fills the appropriate type properties with pattern facets encountered in the input schema, which are viewable in the Type Designer.

XML schema datatypes

This section describes the type tree constructs that correspond to XML Schema datatypes.

The following table lists some common XML Schema datatypes with their corresponding type property values in the Type Designer.

For detailed information about XML attributes and elements as type properties in the type tree, refer to the Type Tree Importer documentation.

XML Schema Datatype	Type Property	Value
simpleType	Intent > Validate As	XML_SIMPLETYPE
complexType	Intent > Validate As	XML_COMPLEXTYPE
element	Intent > Validate As	XML_ELEMENT
attribute	Intent > Validate As	XML_ATTRIBUTE
group	Intent > Validate As	XML_GROUP

- [Simple types](#)
- [Complex types](#)
- [Elements](#)
- [Attributes](#)
- [Groups and substitution groups](#)

Simple types

Elements that have assigned simple types have character data content but not child elements or attributes.

Simple type elements are represented in the Type Designer type properties with the value XML_SIMPLETYPE.

Complex types

Elements that have assigned complex types can have both child elements and attributes. The order and structure of the child elements of a complex type are known as its content model. Content models are defined using a combination of model groups, element declarations or references, and wild cards. There are three kinds of model groups: **all**, **choice** and **sequence**.

Complex type elements, including the complex types using compositor elements, are represented in the Type Designer type properties with the value XML_COMPLEXTYPE.

Elements

Element declarations are either local or global. Local elements are represented in the Type Designer type properties with the value: XML_ELEMENT.

When the schema defines several global elements, the generated type tree includes a Global choice group that contains all of the global elements. A Global choice group is represented in the Type Designer type properties with the value XML_BODY.

Attributes

Attributes are another building block of XML. The import process uses attribute declarations to name attributes and associate them to particular simple types. Attribute declarations can be either local or global.

Attributes, including both local and global, are represented in the Type Designer type properties with the value XML_ATTRIBUTE.

Groups and substitution groups

One way an XML Schema allows the creation of reusable content is by using named model groups. These global groups can be referenced throughout a schema.

Substitution groups are a flexible way to designate element declarations as substitutes for other element declarations in a content model. New element declarations can easily be designated as substitutes from other schema documents or namespaces without changing the original content model.

Groups and substitution groups are represented in the Type Designer type properties with the value XML_GROUP.

Identity constraints

The XML parser in IBM Transformation Extender enforces declared identity constraints. The XML Schema Importer sets these properties based on the XML Schema. In general, the identity constraint definition serves in one of the roles listed below:

Identity Constraint	Description
Unique	

- Enforces that a value (or combination of values) is unique within a given scope. For example, all product numbers must be unique within a catalogue.
- Key**
- Enforces uniqueness and requires that all values be present. For example, every product must have a number and it must be unique within the catalogue.
- Key References**
- Enforces that a value (or combination of values) corresponds to a value represented by a key or uniqueness constraint. For example, for every product number that is displayed as an item in a purchase order there must be a corresponding product number in the product description section.
-

Namespaces

The XML Schema Importer supports namespace and prefix properties for all elements and attributes. The importer assigns the values of these properties based on the input grammar. For each namespace in the input grammar, the importer allows you to specify the namespace prefix to be used for output documents.

During the import process, a **Location** value is specified in the type tree without a fully qualified path. In this case, a map execution process would, by default, look for the XML Schema at this location. However, if you need to change this path, you can do so.

For example, if you refer to the proper location of the schema in the XML document, then you can manually remove the path location from the type properties. When the map execution process does not find a **Location** value specified in the type tree, the process then looks to the "[XSDL Hints](#)" specified in the XML document. When XSDL hints are not present, validation fails.

The value specified in the type tree takes precedence over the XML document.

Element type declarations

The XML DTD and XML Schema importers can generate a simple type tree when encountering character content within mixed data. During the import process, when character data (CDATA/PCDATA) is encountered, the importer incorporates it into existing text items within the type tree.

Element and attribute wildcards

DTDs and XML Schemas allow the specification of wildcard elements within a grammar. The IBM Transformation Extender XML Schema and DTD importers recognize element wildcards, build the appropriate types, and set the appropriate properties for those types to represent element wildcards in the type tree.

During the import process, when a wildcard element resolves to an element that is not defined in the imported grammar, it is represented by a text item.

The value of these text items is the text string from the input buffer that runs from the start of the open tag to the end of the close tag for that element.

XML Schemas allow the specification of an attribute wildcard that matches any number of undeclared attributes within the element tag. The XML Schema Importer recognizes the attribute wildcard and creates a sequence of name and value text item pairs within the attribute list of the enclosing element. The XML validation library fills this sequence with the names and values of attributes that do not match any declared attributes in the attribute list.

Item restrictions

Item restrictions are an optional feature you can use to specify valid or invalid data objects for an item type. Item restrictions are grouped into three categories: **Value**, **Character**, and **Range**.

Restrictions of an item type are the valid or invalid values for that item. "Include" restrictions are all of the valid values for an item. "Exclude" restrictions specify all of the invalid values. For example, a unit of measure field in the data must be one of a set of values: CN, BX, PK, BR. These values should be defined as "include" restrictions of the item **UnitOfMeasure**.

Defining restrictions for an item restricts the valid data for that item. Partial lists of the valid values are not possible. For example, it is not possible to define the values for a certain item to be {A, B, C, "some other possible values"}.

While you can define restrictions for any item, most items will not have restrictions. For example, you would not want to restrict the valid values of a name field because you would probably want to accept any name as valid data for that field. However, you could assign restrictions to it if needed.

- [Defining item restrictions](#)
- [Restrictions settings](#)
- [Ignoring restrictions](#)

Related tasks

- [To view a specific component/item restriction](#)
-

Defining item restrictions

About this task

By default, restrictions are case-sensitive. To ignore case when defining the properties of the item, choose **Yes** for the **Ignore Case** property.

In the following instructions, it is assumed that the **Enter** key is the "commit changes" key. (See [Customizing the Type Designer environment](#).)

To define restrictions:

Procedure

1. Double-click the item for which you want to define restrictions.
2. Type each restriction value in the appropriate cell and press **Enter**.
The column headings of the restriction cells will differ based on the item properties set.
3. (Optional) In the corresponding cell, enter a description for the restriction value. A description is especially helpful when the value of the restriction is not obvious.
You can add new rows for restrictions by pressing the **Insert** key or by right-clicking and choosing Insert from the context menu. The new row inserts *after* the selected row.

Related tasks

- [Syntax objects with variable values](#)

Related information

- [Error detection and recovery](#)

Restrictions settings

Using the Restrictions property, you can create a "restriction list" to limit an item to a particular value or set of values. Depending on the Item Subclass, you can set value, character, or range restrictions.

- [Value restrictions](#)
- [Character restrictions](#)
- [Range restrictions](#)
- [Inserting symbols](#)

Value restrictions

In the value restrictions view for an item type, there is either an Include or Exclude column, depending on the Restrictions Rule setting.

Include column

Use the Include column to specify the restrictions to use in determining if input data is valid or invalid for the item. One-to-many values are allowed. The order in which restrictions appear in the list is not relevant.

"Include" values are considered valid. An input item is considered invalid if a character other than those specified in the **Include** column appears in the data.

Exclude column

Use the Exclude column to specify the text character values to be excluded.

"Exclude" values are considered invalid. This option provides a means to specify a default set of restrictions.

To add value restrictions to an item:

1. Go to the Item Subclass Restrictions property.
2. Choose Value from the drop-down list.
3. The restriction list is case-sensitive by default. If you do not want case-sensitivity, set the Ignore case subproperty to Yes to ignore case.
4. Set the Rule subproperty to Include or Exclude the values that you will provide.
5. Double-click on the type to open the value restrictions view.
6. Enter the value restrictions Include or Exclude column.
7. (Optional) Enter corresponding descriptions in the Description column.
8. Analyze type tree.
9. Save changes.

Related tasks

- [Inserting symbols](#)

Character restrictions

In the character restrictions view for an item type, there is either an Include or Exclude column, depending on the Restrictions Rule setting.

Include character restrictions

Include First column

Use the Include First column to define the first character of a character list.

Include After column

Use the Include After column to define the character list of the characters to follow the first character (defined in the Include First column).

For example, the first character could be a letter and all characters that follow could be either letters or digits.

Exclude character restrictions

Exclude column

Use the Exclude column to specify the substrings in the input data that must be excluded and replaced with the associated reference string.

Reference String column

Use the Reference String column to define the characters that are to replace the excluded character substrings.

On output, a character text item is built by using the corresponding reference string when the content contains any of the Exclude character substrings listed. For example, you can define XML item content that excludes markup delimiters by using the corresponding reference strings (because XML character text items cannot contain the markup delimiters <, >, or & in the raw form).

To make it easier to specify a common range of characters, you can use reserved words and symbols. See the [Design Studio Introduction documentation](#) for a list of reserved words and symbols.

- [To add character restrictions to an item](#)
-

To add character restrictions to an item

Procedure

1. Go to the Item Subclass Restrictions property.
2. Choose Character from the drop-down list.
3. The restriction list is case-sensitive by default. If you do not want case-sensitivity, set the Ignore case subproperty to Yes to ignore case.
4. Set the Rule subproperty to Include or Exclude the values that you will provide.
5. Double-click on the type to open the character restrictions view.
6. Enter the character restrictions in the Include First or Exclude column.
7. Enter corresponding values in the Include After or Reference String column.
8. Analyze type tree.
9. Save changes.

Related tasks

- [Inserting symbols](#)
-

Range restrictions

In the range restrictions view for an item type, there is either an Include or Exclude column, depending on the Restrictions Rule setting.

Include range restrictions

Include Minimum column

(Required) Defines the minimum value of the range.

Include Maximum column

(Required) Defines the maximum value of the range.

Description column

(Optional) Enter a brief description of the range restrictions.

Exclude range restrictions

Exclude Minimum

Defines the minimum value of the range to be excluded or considered invalid.

Exclude Maximum

Defines the maximum value of the range to be excluded or considered invalid.

Description column

(Optional) Enter a brief description of the range restrictions.

Note: Unbound restrictions are not supported.

- [Value not in range](#)
 - [To add range restrictions to an item](#)
-

Value not in range

About this task

The minimum value and the maximum values can be specified as not included in the range. For example, when a number in an Include Minimum field displayed the **Value NOT In Range** icon, it indicates that the number is not included as the minimum value. The range will therefore extend from any value that is greater than that number to the maximum value specified. Similarly, if the **Include Maximum** value is designated as "not in range", the valid range would extend up to any value this is less than the maximum value.

To specify a value as "not in range":

Procedure

1. Select the field (cell) that contains the value to be designated as "not in range".

2. Right click on the value and choose the Value NOT In Range menu item.

Related tasks

- [To add range restrictions to an item](#)
-

To add range restrictions to an item

Procedure

1. Go to the Item Subclass > Restrictions property.
2. Choose Range from the drop-down list.
3. The restriction list is case-sensitive by default. If you do not want case-sensitivity, set the Ignore case subproperty to Yes to ignore case.
4. Set the Rule subproperty to Include or Exclude the values that you will provide.
5. Double-click on the type to open the range restrictions view.
6. Enter the range restrictions in the Include Minimum and Include Maximum or Exclude Minimum and Exclude Maximum columns.
7. (Optional) Enter corresponding descriptions in the Description column.
8. Analyze type tree.
9. Save changes.

Related tasks

- [Value not in range](#)
 - [Inserting symbols](#)
-

Inserting symbols

About this task

Symbols are used to indicate nonprintable characters. You can insert a symbol by typing it or by using the Symbols dialog.

For example, to define a carriage return/line feed as an item restriction, from the item restriction view area, right-click and select Insert Symbols from the context menu. From the list of symbols, double-click CR and double-click LF and select OK. The carriage return/line feed symbols are displayed in angle brackets in the item restriction view area: <CR><LF>

To insert a symbol:

Procedure

1. From the context menu, select Insert Symbols.
The Symbols dialog is displayed.
2. Select a symbol and click Insert or just double-click the symbol.
The symbol appears in the Value field.
3. Click OK.

Related reference

- [Value restrictions](#)
-

Ignoring restrictions

There might be instances when you do not require the data for an item to match any of its restrictions. Suppose you defined restrictions for your data, but you want to run a test on some test data, and you do not want to use the restrictions for this time only. You can ignore the restrictions for a given execution of a map.

In another example, you have a list of valid part numbers that can appear in any data circulated within your company. Suppose you receive data from some other company in the same format as your internal data. The other company may be using different part numbers, so you do not want to make their data conform to the restrictions. When you map the other company's data, you could ignore the restrictions.

For instructions on ignoring restrictions during map execution, see Map Designer and Command Server documentation.

Components

A component represents a data object that is part of another data object.

- [Components are required for group types](#)
 - [Group views](#)
-

- [Defining components](#)
 - [Required and optional data](#)
 - [Defining component rules](#)
 - [Component attributes](#)
-

Components are required for group types

Categories and groups can have components. Components of group and category types display in the group and category views. Item types do not have components.

Group types represent actual data objects, so groups must have components. The one exception is a partitioned group which is explained in "[Partitioning](#)".

By contrast, a category is used for organizing types and for type property inheritance reasons. Categories do not define actual data objects in detail. A category does not need components.

Each group must have at least one component, unless it is partitioned.

IBM Transformation Extender supports large metadata. It supports more than 65,536 types and 65,536 components in a single type tree. To reduce the number of types that maps that use large type trees must process, trim these type trees so that they contain only the required types.

- [Components must be in the same type tree](#)
- [Importance of component order](#)
- [Component range](#)

Related information

- [Type inheritance](#)
-

Components must be in the same type tree

A component must be a type in the same type tree as the type that contains the component.

You cannot define the components of a type by opening up a different type tree and dragging components from that tree. You can, however, copy types from one type tree to another.

Related information

- [Managing types](#)
-

Importance of component order

In the group view, components are listed from top to bottom in the order they appear in the data stream. The component in the first cell appears first in the data stream. The component in the second cell appears next, and so forth.

Component range

A component range can be specified for any component. A component range defines the number of consecutive occurrences of that component. The range (s) represents some or any number greater than one.

When a component is selected, the component name is displayed in the rule bar. Click in the rule bar after the component name to type the component range.

The range is displayed in parentheses immediately after the component name. The range is two numbers separated by a colon. The first number indicates the minimum number of consecutive occurrences of that component. The second number indicates the maximum number of consecutive occurrences of that component. The syntax of the component name and range is:

Component (MIN:MAX)

To enter a range after a component name, type in the rule bar or use the **Set Range** command.

The maximum component range is 2147483647.

Whenever a component has a range, each occurrence of that component may be referred to as a "member" of a series. The words "occurrence" and "member" are used interchangeably in the documentation.

- [Indefinite number](#)
- [Single occurrence](#)

Related tasks

- [Specifying a component range](#)

Indefinite number

When there is no maximum number of occurrences of a component (the maximum is indefinite) use the letter **s** to stand for "Some - you do not know how many". Therefore, a file that contains at least one record and has no maximum number of records has this component:

Record(1:s)

If the range minimum is zero, omit the **0** and only enter **s**. If a file has a minimum of zero records and no maximum number of records, it would have this as its component:

Record(s)

Remember that when you see **(s)**, the minimum of zero is implied. Therefore,

Record(0:s) is the same as Record(s)

If you enter any number alone in the parentheses, the range changes to a minimum of 0 and a maximum of that number. For example, if you enter **(5)** for a range and save and close the group view, the next time you open it, you see the range displays:

(0:5)

Single occurrence

If there is a minimum and a maximum of one consecutive occurrence of a component, its range is **(1:1)**. This is the default. If there is no range after a component name, the range **(1:1)** is implied. When you drag a type to make it a component, it is automatically created with the default component range of **(1:1)**.

Group views

When you select a group in the type tree editor, the components are listed in the Composition view.

When you double-click a group in the type tree editor, the group view opens and you can see the components of the group here as well. The contents of a selected cell are displayed in the rule bar of the group view.

- [Viewing nested components](#)

Viewing nested components

You can view nested components from the Composition view or from the group view.

As an example, open `install_dir\examples\general\deliver\deliver.mtt`. To view the components in the Composition view, select the **Collection** group type. **Record(s)** appears in the Composition view. Expand **Record(s)** to view the nested components. When you double-click the **Collection** group type, the group view opens. In the Component column, expand **Record(s)** to view the nested components.

You cannot add component rules to nested components or partitions.

Defining components

About this task

Most groups need at least one component. Categories do not have components, however, you can define components for a category for inheritance purposes.

To determine the components of a group, ask the question:

Group type _____ consists of what?

For example, "The file consists of what?" The answer might be "records". So, **Record(s)** is a component of that group or category type.

Suppose you have a data file containing order records for an office supply store. You need to define the components of the type **File**.

To define components:

Procedure

1. In the type tree, double-click the group or category type whose components you want to define.
The group or category view opens.
2. Drag each type from the type tree into the component column.
The type name is displayed in the Component column.
3. Enter component ranges where necessary.
4. Drag each component from the type tree into the group view.
5. Enter component ranges where necessary. A component range indicates the number of occurrences. The number of records in the file is indefinite, so the range is **(s)**, which represents "some." When a component is highlighted, it appears in the component rule bar.

For example, in the component rule bar, place the cursor after **Record**, type (s), and press **Enter**.
6. Close the view and click Yes if prompted to save changes.

- [Complete type name](#)
 - [Relative type names](#)
 - [Ambiguous type names](#)
 - [Always drag components](#)
 - [Viewing the component number](#)
 - [Specifying minimum and maximum consecutive occurrences in the component list](#)
 - [Fixed and variable ranges](#)
 - [Specifying a component range](#)
 - [Viewing the range column](#)
 - [Types that can be components](#)
 - [Variable component names](#)
 - [Opening a component view](#)
-

Complete type name

The complete name of a group is displayed in the title bar of its window. A type's complete name is similar to a path name, beginning at the type and including all types in the path up to the root. Spaces appear between types in a complete type name.

You can have duplicate type names as long as they are not on the same level. Each type has a unique complete name, so there is no doubt as to which particular type is being referenced.

Relative type names

A component name is similar to a relative path name. Component names exclude types that the defined type and component type have in common in their complete names.

The relative type name excludes the names that appear in the complete names of both the type and the component type. For example, because **ROOT** is included in the complete type name of both **Row ROOT** and **Product Column ROOT**, it is excluded from the relative type name **Product Column**.

If more than one type in a type tree has the same relative type name as another in a group, then the full path of the type is exported instead of the relative type name. No two types that have different full type names but identical relative type names can be added to a component list.

- [Moving types with the same relative type name](#)
-

Moving types with the same relative type name

In a type tree, there could be two types with the same name that exist in different places within the tree. The relative type names could both evaluate to be the same, with respect to the component.

For example, the relative type **Group Category ROOT** might have a type named "Item" in both **ROOT** and **Category ROOT**. Both evaluate to "**Item**" with respect to the **Group** component. In this scenario, you can drag only one component named **Item** into the component list. An attempt to add a second component with the same name will be blocked.

Related concepts

- [Ambiguous type names](#)
-

Ambiguous type names

A component name can refer to more than one type. In this situation, you must change type names so that the component name is no longer ambiguous. The Type Designer does not allow components with the same relative type name to be added to component lists.

If you delete a type that is used as a component of another type, and then perform an "Undo", thus adding the type back, and the component name is ambiguous; the component name remains unresolved in the component list. In this scenario, the "Undo" operation does not result in a complete reversal of the action.

- [Manual entry of types with same relative type names](#)
-

Related concepts

- [Moving types with the same relative type name](#)
-

Manual entry of types with same relative type names

Although the drag-and-drop method is the recommended method of entering component names into the component list, it is possible to type the component list manually into the text entry area of the group view.

When you manually enter a type name in the component list that has the same relative type name as an existing component, a special icon appears. This ambiguous type name icon precedes the component name within the component column.

An *ambiguous type name* indicates that there is another type with the same relative type name in the type tree. There are two ways of eliminating the ambiguity:

- Drag one of the types from the type tree onto the type name in the component list. When you do this, the ambiguous type icon will change to a solid color to indicate that the ambiguous type name is resolved. If you attempt to drag the second type name from the type tree into the component list, you will be blocked.
- Type a path name in the group view. You must type either the full path name of the type, or if you type a partial path name it must contain enough of the path name to make the type unique.

For example, if you enter **Item Category ROOT**, or **Item Category**, the type name **Item** will be unique and the ambiguity will be removed. When you enter this unique path name, the type will be successfully entered into the component list. The ambiguity is thereby resolved.

Always drag components

Because the component rule bar allows you to add a component range, the entire component name is editable. You could enter a component name by typing it. However, this is not advisable. Always drag components into component lists. This is recommended for the following reasons:

- To avoid typographical errors
- To enter type names in the exact case in which they were created. (Type names are case-sensitive.)
- To avoid incorrectly entering a relative type name

When you drag types into a component list, the correct relative type name is automatically entered.

Viewing the component number

About this task

You can view the number of each component in a group view by changing your user preferences.

To view component numbers:

Procedure

1. From the Window menu, select Preferences > Transformation Extender > Type Tree > Group Window.
2. Enable the **Show component number** check box.
3. Click Apply and click OK.

Specifying minimum and maximum consecutive occurrences in the component list

The following table contains examples of how to specify in the component list the minimum and maximum consecutive occurrences for specific data objects:

Data Object	Min	Max	How to Specify
DateField	1	5	DateField (1:5)
DetailRecord	1	100	DetailRecord (1:100)
AddressField	2	3	AddressField (2:3)

Fixed and variable ranges

Sometimes the range of a component is described as fixed or variable. As an example, a fixed range has the same minimum and maximum, (5:5). A variable range has a different minimum and maximum, (1:10) or (s) for example.

Specifying a component range

About this task

You can specify a component range by using the Set Range functionality.

To enter a component range:

Procedure

1. Right-click on a component and select Set Range from the context menu.

2. In the Set Range dialog, enter the minimum and maximum number of occurrences.
3. Click OK.

Results

You can select multiple components and use Set Range to apply the same range to each component.

A range of (1:s) means that there will always be at least 1 occurrence, but could occur an infinite number of times. A range of (0:s) means that an occurrence is optional, but could occur an infinite number of times.

What to do next

Tip: You can view a shortcut menu for the component by clicking the down arrow located on the view toolbar: 

Related concepts

- [Component range](#)
-

Viewing the range column

About this task

To view component ranges from the Group view:

Procedure

1. From the Window menu, select Preferences > Transformation Extender > Type Tree > Group Window.
 2. Enable the **Show range column** check box.
 3. Select Apply; select OK.
-

Types that can be components

There are certain guidelines to follow when defining types. It is not necessary to memorize these guidelines because the Type Designer interface assists you when defining components. The Type Designer does not allow the dragging of invalid components.

- [Guidelines for defining components](#)
-

Guidelines for defining components

The guidelines below are numbered to allow references to each other. The numbering does not suggest a priority or a sequence to be followed in observing the guidelines.

1. Categories cannot have components.
A category is only used for organizing your type tree and for setting common properties.
 2. A partitioned group cannot have components.
A partitioned group always represents a choice among the subtypes of that group. You never map a partitioned group without its subtree, so it does not need components.

Note: A *subtree* is a branch of a type tree that includes a type and all of the subtypes that stem underneath it.
 3. If a group is not partitioned, it must have at least one component.
Nonpartitioned groups are sequences of data objects rather than choices. A sequence must contain at least one component.
 4. A type and one of its subtypes cannot be in the same component list.
 5. If a type has components, a subtype can inherit any of those components or any type in the subtree of one of those components.
 6. If a type has no components, a subtype can inherit any type that could be in that type's component list.
 7. A type that has an initiator and a terminator can have itself or one of its ancestors as a component.
 8. A type cannot have one of its subtypes as a component.
-

Variable component names

A component can refer to more than one type. To refer to all possible types whose names could appear at a certain place in the component name, use the word ANY.

The word ANY is like a wild card. It represents any type whose name could appear in that place.

The use of ANY is restricted to **Categories** and partitioned groups. For more information about using ANY in a partitioned group, see ["Partitioning"](#). For a list of reserved words and symbols, see the Design Studio Introduction documentation.

Opening a component view

About this task

Procedure

Double-click the component.

For example, if the Order window is open and it has the component **Row (s)**, when you double-click the component **Row (s)**, it opens the window of **Row**.

Required and optional data

Sometimes, a certain data object is optional; it does not have to be present in the data. For example, in purchase order data, there might be a billing address and a shipping address. If the company wants the items shipped to the billing address, the shipping address would not appear in the data. The shipping address would be optional; it might not appear in the data.

Another example is a middle name field. Some people do not have a middle name, so the middle name field might be optional.

The Type Designer needs to know what data is optional. This is evident from the component range. The range minimum tells how many occurrences of that object must be present in the data. These are the required occurrences. Optional occurrences are the ones that are not required.

For example, for the following component, the range minimum is zero. No occurrences must be present. It is optional data:

DateField (0:1)

Suppose the component looks like this:

DateField (1:5)

The range is between one and five occurrences. This means that one occurrence of **DateField** is required and the remaining four occurrences are optional.

The following table lists examples of components and explanations of their status.

Component	Status	Reason
LineItem (1:s)	1 occurrence required	Range minimum is 1
Note Field (5:5)	5 occurrences required	Range minimum is 5
RecordID	1 occurrence required	Range minimum is 1
ShipTo (0:1)	0 occurrences required (Optional)	Range minimum is 0
OrderRecord (s)	0 occurrences required (Optional)	Range minimum is 0

- [Significance of required data](#)

Significance of required data

If you define an occurrence of a component as required, you are saying that, for the data containing the component to be *valid*, this component must exist. If it does not exist, the data is *invalid*.

For example, if you define **Record** as having the component **Field (3:3)**, you are saying that there must be three **Fields** in the **Record**. If there are not three **Fields**, then either it is a **Record** in error or it is not a **Record**.

There are other factors, besides the existence of all required components that make data valid. The existence of required components is necessary, but not sufficient, for data to be valid.

Defining component rules

A component rule is an expression about one or more components. It indicates what must be true for that component to be valid. For given data, it evaluates to either "true" or "false". A component rule is similar to a test. If the data does not pass the test, it is invalid.

Component rules are used for validating data. Some important points about component rules are:

- Only components of a group can have component rules. Components of a category cannot have component rules.
- A component rule cannot be longer than 32K.
- If a component is optional and does not appear in the data, the component rule is evaluated after determining that the data is missing. In a component rule, you can specify relationships that depend on existence or nonexistence of data.

Sometimes components have relationships among each other. For example, an address field in purchase order data is preceded by a qualifier field which tells whether the address is a bill-to or a ship-to address. If the value of the qualifier field is **BT**, the address field following it is a bill-to address. If the value is **ST**, the following address is a ship-to address.

The qualifier and address fields are dependent on each other. They only make sense as a pair. If one of these fields is optional and is missing, the other field is not meaningful. If the qualifier is missing, you do not know whether the address is a bill-to or a ship-to. If the address is missing, the qualifier does not qualify anything!

You might want to define this kind of relationship and other relationships among data objects. To do this, use a component rule. For example, use a rule on the address field component to indicate that the qualifier must be present if the address is present.

The following instructions, assume that the `Enter` key has been assigned to commit changes.

To enter a component rule:

1. In the group view, select the rule cell of the component.
2. Enter the component rule in the edit area and press `Enter`.

- [Examples of component rules](#)
- [Component rule syntax](#)
- [Entering object names in component rules](#)
- [Shorthand notation](#)
- [Component rules are context-sensitive](#)
- [Special characters in component rules](#)
- [Inserting functions into component rules](#)
- [Formatting a component rule](#)
- [Comments in component rules](#)
- [Syntax errors](#)
- [Managing components](#)

Related tasks

- [Using search and replace](#)

Examples of component rules

A component rule can limit the acceptable values of a component as shown in the following example:

```
Quantity < 10000
Interest Rate > .13 & Interest Rate < .20
WHEN (PurposeCode != "PF", ShipValue = 200|ShipCode < 0)
```

The following example illustrates how a component rule can make the presence of one component mandatory, if another component is present:

```
WHEN (PRESENT (Address Field), PRESENT (Qualifier Field))
WHEN (PRESENT (PhoneNumber), PRESENT (AreaCode), ABSENT (AreaCode))
```

A component rule can compare a component to the result of an arithmetic operation as shown in the following example:

```
SUM (((QuantityOrdered:Item Record:Detail) = TotalQuantity:Summary Record:Detail
Account Balance = Credits - Debits
Extension = Quantity*Price
#Items Field = COUNT (Item Record IN Invoice))
```

Component rule syntax

A component rule is a complete expression that evaluates to either "true" or "false". It can contain functions (for example, PRESENT, COUNT, and SUM). It can also contain arithmetic operators (such as `- + * /`).

A component rule is a statement, so it does not start with an equal sign.

For more information about the syntax of expressions, see the Functions and Expressions documentation.

Entering object names in component rules

About this task

A component rule can refer to a component within a component. The syntax for a component is the component name, followed by a colon (:), followed by the object of which the component is a part. Whenever you see the colon (:) in an expression, it can be interpreted as "component of" or simply "of".

For example, the following expression means "The item number of the order record of the order":

```
Item#:OrderRecord:Order
```

All of the objects that can be used in component rules are shown in the group view. You can enter an object name into a component rule by pressing `Alt` and dragging the object into the edit area. The complete object name is automatically entered.

A component rule can refer to:

- The component it applies to and any nested components.
- Any component above the given component in the component list, and its nested components.
You can enter an object name in a component rule by typing it; however, the recommended method is to press `Alt` and drag the component into the edit area.

Shorthand notation

In a component rule, the dollar sign (\$) represents the component itself. When you enter an object name in a rule by pressing Alt and dragging the object, the dollar sign is automatically entered in the rule to represent the given component.

Component rules are context-sensitive

Component rules apply to components, not types. It applies to data in a certain context when the data is a component of a given group.

Suppose you have some order data that contains two kinds of records: a regular order record and a bulk order record. In the bulk order record, the quantity ordered must be greater than 1000. In the regular order record, the quantity can be any number. You could put a rule on the quantity ordered component of a bulk order, but not on the quantity ordered of a regular order.

Special characters in component rules

About this task

To enter the actual value of a special character in a component rule, you must enter the Hex value for one of the characters. For example, to enter the text value <WSP>, enter the Hex value for the less than sign <<3C>>, and then the rest:

```
<<3C>>WSP>
```

See the Design Studio Introduction documentation for a list of non-printable Hex and decimal values.

Inserting functions into component rules

About this task

You can use most of the functions provided with the product in component rules.

Component rules shown in the [examples](#) section use the functions COUNT, PRESENT, SUM, and WHEN. The function WHEN is similar to the function IF. However, WHEN evaluates to "true" or "false" and IF evaluates to a data object. WHEN is most often used in a component rule.

To insert a function into a component rule:

Procedure

1. In the type tree editor, double-click the type.
The Component view opens. The Component view consists of an edit area and two columns: Component and Rule.
2. In the Component column, select the component to which you want to add a function.
3. Press Ctrl+Space bar to open the list of functions.
4. From the list of functions, double-click on the function you want to insert.

Formatting a component rule

About this task

Depending on your user preference settings, you can add a new line to a component rule by pressing Ctrl and Enter (when Commit changes with is set to Enter), or by pressing Tab (when Commit changes with is set to Tab).

Comments in component rules

About this task

You can add comments to component rules. Comments do not affect how component rules are evaluated.

A comment begins with the characters /* and ends with the characters */. A comment can appear anywhere in a rule as long as it does not separate object names.

For example, the following component rule has a comment:

```
SIZE (Phone# Field:$) >=7  
/* Phone numbers must include area code */
```

Syntax errors

If you type a component rule that is syntactically incorrect and press **Enter**, the part of the rule in error is highlighted and you cannot "commit" it until the error has been corrected.

Managing components

About this task

You can easily move components around, copy them, or delete them. For information on general drag-and-drop procedures, see "[Managing types](#)".

When you move, copy, or delete a component, if it has a range and/or a component rule, the range and rule are moved, copied, or deleted with the component.

Component attributes

Three attributes can be assigned to a component in a component list:

- Identifier 
- Restart 
- Sized 

When an attribute is applied to a component, a corresponding icon is displayed with a component name in the component list.

To assign an attribute to a component:

1. In the type tree editor, double-click a component. The component view opens.
2. In the component list of the component view, right-click on the component and choose the attribute from the context menu.
To remove the attribute, repeat the procedure.

- [Identifier attribute](#)
- [Restart attribute](#)
- [Sized attribute](#)

Identifier attribute

The identifier attribute can be used on a component of a group. The identifier indicates the components that can be used to identify the type to which a data object belongs. All the components, from the first, up to and including the component with the identifier attribute, are used for type identification.

When this data is validated, it knows that, when it reaches the identifier, it has found a specific group. That group, therefore, is known to exist, even if part of the group following the identifier is missing. The Map Designer documentation discusses how data validation occurs.

The identifier attribute is also useful when identifying an object of a partitioned group. For information about using an identifier with partitioned data, see "[Partitioning](#)".

In a component list, there can be only one identifier attribute.

Restart attribute

About this task

To continue processing your input data when a data object of a component is invalid, assign the restart attribute to that component.

Do not put the restart attribute on a required component. There must be a sufficient number of valid instances to cover all required components. If you have a required component that is not valid, the restart attribute does not validate the data.

For additional information about using the **Restart** attribute, see the Map Designer documentation.

Related concepts

- [Using the Restart attribute](#)

Sized attribute

The sized attribute is used on a component in which the value specifies the size (in bytes) of the component immediately following it. The sized attribute can be used on more than one component of a group.

For example, you might have a variable length component with a number immediately preceding it that indicates the length of the component: **10Washington**. The size of the component would be 10.

Some important points about using the sized attribute are:

- The component with the sized attribute must be defined as an unsigned integer.
- If a binary byte stream item does not have a fixed size, the component preceding it must specify its size and the sized attribute must be used on that component.

The size of a component is the number of bytes from the beginning of that component, up to and including the end of the component. If a component has a series range (such as [1:3]), the size includes all of the members in the series of that component. If a delimiter separates each member of that series, the delimiters must be included in the size. Also, if release characters appear in the component, they must be included in the size.

The size does *not* include delimiters that separate one component type from the next.

- [Include self in size](#)

Include self in size

If the value of the component with the sized attribute includes the size of the component, use the **Include Self in Size** option. For example, suppose the component with the sized attribute has a length of seven bytes. The value of the component with the sized attribute would be seven. So, its data would be one byte long, to hold the character seven. If its value includes the length of itself, its value would be $7 + 1 = 8$.

For example, a sized component is 7 bytes.

+ 1 byte (the length of the character 7)

If **Include Self in Size** is selected: 8 bytes

Partitioning

By partitioning, you can define your data to distinguish the difference between data objects based on values in the data or differences in the syntax.

Partitioning is a method of subdividing objects into mutually exclusive subtypes.

A special icon in the type tree identifies a partitioned type. The partitioned type maintains the same class.

To partition a type:

1. Select the type you want to partition. (You can only partition items and groups.)
 2. From the Type menu, choose Properties.
 3. For the **Partitioned** property, choose **Yes**.
- [Determining when to partition](#)
 - [Partitioning types](#)

Related concepts

- [Distinguishable components of an implicit group](#)

Determining when to partition

There are some cases in which you will *need* to partition your data and others in which you will *want* to partition your data. You are *required* to partition for unordered data when a data object at a certain place in the data stream can be any number of types and each type has different definitions. *Choose* to partition for convenience, either to simplify mapping rules or to put additional logic into your data's definition.

- [Required partitioning](#)
- [Partitioning for convenience](#)
- [Benefits of partitioning](#)

Required partitioning

Partitioning is required when components are randomly or partially ordered. The following example represents unordered data:

BGI - 13100,REM,931104,19970424...

AXR - 10930,INV,003X114,19970422...

PVY - 19496,ORD,PO-104-1499,19970425...

BGI - 13100,ORD,PO-182-2587,19970425...

AXR - 10930,INV,003X-114,19970422...

The file would contain three different types of transactions: **Invoice**, **Order**, and **Remittance**. Each transaction has a different definition based on the type of information it represents.

Partitioning for convenience

You might decide to use partitioning in your type tree to build additional logic into the definition of your data. You may also use partitioning to simplify the rules needed in your map.

The following is an example of partitioning to simplify rules. The example compares the differences between rules needed *with* and *without* partitioning. In the rule without partitioning, you would specify a condition for each state abbreviation in each region. This could make your mapping rules long, difficult to read, and difficult to maintain. The mapping rule with partitioning is more concise, self-documenting, and easier to maintain.

Map rule without partitioning:

```
=IF(ShipToCode Field::Input=="NY" |
    ShipToCode Field::Input=="NJ" |
    ShipToCode Field::Input=="PA",
    F_MapEast (Record:Input), NONE)
```

Map rule with partitioning:

```
=F_MapEast (EXTRACT (Record:Input,
PARTITION (ShipToCode Field::Input, East)))
```

Benefits of partitioning

Using the [previous example](#), explore the benefits of partitioning.

- The rule is shorter than the rule without partitioning. The knowledge of which states belong to each region is maintained in the type tree rather than the map rule.
- It is easy to read this map rule and understand the mapping function being performed. For example, if the state belongs to the list of states in the eastern region, execute the **MapEast** functional map.
- The partitioning method is easier to maintain. If a value for **State** is added or moves from one region to another, it can be easily changed in the type tree and automatically reflected in any mapping rules that reference the partitioned object.
- Partitioning using a restriction list used with the **Ignore Case** setting eliminates the need for PROPER, LOWERCASE, or UPPERCASE functions to compare each state with a literal.

Partitioning types

When data objects of different types appear in the same place in the data, the types must be distinguishable. This means that the data needs to be distinguishable by their definitions in the type tree.

When the data object at any given point in the data may belong to any of a number of different types, there must be some way to tell the difference between them. To do this, you create a type and define a mutually exclusive subtype for each data object that may appear in the same place in the data. Once the subtypes are created, the subtypes also need to be distinguishable. Subtypes are distinguishable based on a value in the data or in the syntax of the different types.

- [Partitioning items](#)
- [Partitioning an item type using initiators](#)
- [Partitioning an item type using restrictions](#)
- [Partitioning an item type by format](#)
- [Partitioning groups](#)

Related concepts

- [Objects of a partitioned type](#)

Partitioning items

Use one of the following three methods to partition items in type trees:

- Initiators
- Restrictions
- Format

Partitioning an item type using initiators

To partition by initiator, each subtype must have an initiator and the value of the initiator must be unique for each subtype.

Partitioning by initiator is the most efficient method of partitioning.

Related concepts

- [Partitioning a group type using initiators](#)
-

Partitioning an item type using restrictions

If an item has restrictions, you can partition that type, create mutually exclusive subtypes, and divide the restrictions between subtypes. A restriction cannot appear in more than one subtype of that item.

- [Example of using restrictions](#)
-

Example of using restrictions

You have new data that needs to be entered into the Type Designer. Each new record contains the name of the employee and his or her department.

The **Employee List** type tree illustrates components of **Record**.

The following is the new data containing the name of the employee and the department.

Steven Barlow,Doc

Heather Proust,Qa

Mary Whiting,Doc

Genie Elks,Sup

Francine Maxwell,Dev

Mark Brown,Sup

Daryl Schwartz,Acc

Harry O'Brian,Sal

Ellen Randolph,Dev

Paula Keller,Qa

Define the values of **Department** as a character text item.

Define the example data as valid restrictions (enter in the **Include** column) of **Department**:

If the departments are located in different offices, you can divide the data into separate files; one file per office. To do this, map the data from the main office to one file, the data from the development office to one file, and the data from the support office to another file. Then create subtypes of **Department: MainOffice**, **DevelopmentOffice**, and **SupportOffice** and partition **Department**. When you partition **Department** and create subtypes, you are saying that a given department data object belongs to only one of the subtypes based on its value.

The subtypes of **Department** inherit the restrictions of **Department**. Now allocate restrictions among subtypes. To do this, delete the restrictions from the subtype that do not apply to that particular office. For example, the **MainOffice** item has only the departments in that office, **DevelopmentOffice** item has only the departments in that office, and the **SupportOffice** item has only the departments in that office.

Partitioning an item type by format

Subtypes that differ by their format are distinguishable from each other.

Partitioning groups

Use one of the following three methods to partition groups in type trees:

- Initiators
 - Identifiers
 - Component rules
- [Partitioning a group type using initiators](#)
 - [Partitioning a group type using identifiers](#)
 - [Partitioning a group type using component rules](#)
-

Partitioning a group type using initiators

To partition by initiator, each subtype must have an initiator and the value of the initiator must be unique for each subtype.

The method of partitioning by initiators for group types is similar to item types.

Related concepts

- [Partitioning an item type using initiators](#)
-

Partitioning a group type using identifiers

About this task

In a component list, only one component can have the identifier attribute.

The identifier attribute distinguishes the components that can be used to identify the type to which a data object belongs. Typically, use this technique to distinguish group partitions when components following the identifier are different for each partition, or, if you have a multilevel partitioned subtree. In the latter case, using an identifier accelerates data validation.

A partition is valid when each component up to and including the identifier is validated. If the set of components is valid, the partition exists.

If the identifier set of components is not valid, the partition is determined not to exist. Either validation occurs for the next partition at the same level (if there is one) or it is determined that the partitioned group does not exist.

If the partition exists, what occurs next depends on the position of the partition type in the subtree.

- If the partition is partitioned (that is, it has subtypes), the rest of the components are skipped and the process begins to validate subtypes until a subtype is valid, exists and is in error, or does not exist.
- If the partition has no subtypes, the remaining components are validated. If all remaining components are valid, the partition not only exists, but also is valid. If one or more components are found to be in error, the partition exists, but its type is in error. If the partition exists, but its type is in error, the error is propagated back up the partitioned subtree until the group being validated is reached. When a partition is found to exist, the system will not continue to search for partitions.

To specify a component as the identifier:

Procedure

1. From the type tree editor, double-click a component.
The Component view opens.
2. In the Component column, right-click on the component and choose Identifier from the context menu.

Results

The identifier symbol appears in the component column next to the component name.

Partitioning a group type using component rules

Component rules are used to partition data when a value or range of values can be used to distinguish one partition from another.

Type inheritance

Properties, components, and restrictions of a type can be inherited by the types created as subtypes under it. Some properties of a type can also be propagated to already existing subtypes.

When you create a type, it becomes a subtype of whatever type is selected at the time. Everything that defines a type gets passed down: properties (with the exception of the Partitioned property), components, and restrictions. After a type is created, you can then modify any aspect of its definition.

When a group is created within a category, the item properties do not apply, so they are not inherited. When an item is created under a category, the group properties are not inherited.

- [Inheritance of item properties and restrictions](#)
- [Inheritance of category properties and components](#)
- [Propagation of type properties](#)

Creation time is not the only time type properties can be passed from a type to its subtypes. You can use propagation to pass along certain type properties.

Related concepts

- [Components are required for group types](#)
-

Inheritance of item properties and restrictions

Properties and restrictions of items are inherited when a new item is created.

For example, an item named **Department** is defined as a character text item that has a content size minimum of 2 and maximum of 3. Department has a list of "include" restrictions that consists of valid departments: ACC (Accounting), SLS (Sales), and MKT (Marketing).

The type **MainOffice** is created as a subtype of **Department**. **MainOffice** inherits the properties and restrictions of **Department**.

You can delete any restrictions that are not applicable to **MainOffice**.

Inheritance of category properties and components

Categories can be used for organizing types and for inheritance reasons. Generally, you would use categories when you want to put items, groups, and possibly other categories under it as subtypes.

- [Organizing types under a category](#)
- [Using categories for inheritance](#)
- [When not to use categories](#)

Organizing types under a category

If you have two tables defined in one type tree, you could divide the types of the two tables into different categories. A benefit of using a category type is that you can have any class of subtype: groups, items, and other categories.

Using categories for inheritance

The properties of a category include group and item properties. Assign properties to a category that you want types beneath the category to inherit.

Any group created under a category inherits the property of the category. Any item created under a category inherits the category's item properties. A category created under a category inherits both the group and item properties. Each type created under a category inherits the other properties of the category, such as initiator and terminator.

As an example, if most groups in your **LabInfo** data are infix delimited with ~ and most items in your **LabInfo** data are unsigned integers, you can define these as the properties of the category **LabInfo**. Any types you create under **LabInfo** inherit its properties.

You can change the properties of the root type, which is a category. Categories can have components, so you can also use them for inheriting components.

When not to use categories

It is best not to use a category type when the subtypes will specifically be item types or specifically be group types.

For example, you have a type named **Field** for which the only subtypes will be items. If you make **Field** a category type, each time you create a subtype, the new type will also be a category type by default. You will have to change the **Class** property from **Category** to **Item** for each subtype that you create. Instead, make **Field** an item type so that the **Class** property for any subtypes you create will automatically be defined as item types.

Propagation of type properties

Creation time is not the only time type properties can be passed from a type to its subtypes. You can use propagation to pass along certain type properties.

Propagation passes the setting of a particular type property from the given type to all of the types in the subtree as applicable and defined by their context. When you select a type property for propagation that does not apply to a given subtype, it is not propagated.

- [Propagation of type properties common to all types](#)
You can propagate some type properties that are common to all types such as Item Subclass, Description and Type Syntax, to each of its subtypes, without the process checking additional criteria to determine if the type property can be propagated.
- [Propagation of type properties for specific types](#)
You can propagate some other type properties such as Separators and Restrictions, to each of its subtypes for specific types, with the process checking additional criteria to determine if the type property can be propagated.
- [Propagating type properties](#)
These are the instructions for propagating type properties in type trees.

Propagation of type properties common to all types

You can propagate some type properties that are common to all types such as Item Subclass, Description and Type Syntax, to each of its subtypes, without the process checking additional criteria to determine if the type property can be propagated.

The type properties that fall into this category are the properties in the root of the tree, or are children properties that apply to all types to which their parent property is set, and are global in that these properties apply to all items.

The propagation process checks that the settings for some type properties such as the Item Subclass property, the Interpret as property, and if applicable, the Presentation property in a type, match the settings for these same properties in its subtypes, and if they match, the selected property is propagated. But, the propagation process does not check the settings of other properties to determine if the selected property exists in the item, because the selected property in this case, applies to all item types.

Examples of propagating a type property that has no children properties

A given category type that is in the root of the tree has the following type property setting: Description._>An example. The Description type property has no children properties. When you select the Description type property setting of An example to be propagated, the process propagates the Description setting of An example to all types in the tree. The Description property is a type property that is common to all types.

A given category type has the following type property settings: Item Subclass._>Text and for its child property, Interpret as._>Character. When you select the child Interpret as type property to be propagated, the process propagates the Interpret as setting of Character to each subtype of that category that also has the same Text type property setting. The Interpret as property is a type property that is common to all types except syntax types.

Examples of propagating a type property that has children properties

A given category type has the following type property settings: Item Subclass._>Number, and for its children properties, Interpret as._>Binary, and Presentation._>Integer. When you select the parent Item Subclass type property to be propagated, the process propagates the Item Subclass setting of Number, as well as the settings for its children properties, Interpret as setting of Binary, and the Presentation setting of Integer, to each subtype of that category. The Item Subclass property is a type property that is common to all types.

A given category type has the following type property settings: Item Subclass._>Text and for its child property, Interpret as._>Character. When you select the parent Item Subclass type property to be propagated, the process propagates the Item Subclass setting of Text, as well as the setting for its child property, Interpret as setting of Character, to each subtype of that category. The Item Subclass property is a type property that is common to all types.

Related tasks

- [Propagating type properties](#)

Propagation of type properties for specific types

You can propagate some other type properties such as Separators and Restrictions, to each of its subtypes for specific types, with the process checking additional criteria to determine if the type property can be propagated.

The type properties that fall into this category are children properties that do not apply to all types to which their parent property is set, and are specific to a type, in that these properties do not apply to all item types.

The propagation process checks that the settings for some type properties such as the Item Subclass property, the Interpret as property, and if applicable, the Presentation property in a type, match the settings for these same properties in its subtypes, and if they match, the selected property is propagated. But, depending on the property being propagated and the context, the process also checks other property settings. If you are propagating a child property, depending on the context, the process also checks the settings of the parent property, because the selected property might only apply to specific item types.

Examples of propagating a type property that applies to a specific type

A given category type has the following type property settings: Item Subclass._>Number, and for its children properties, Interpret as._>Character, and Presentation._>Decimal. When you propagate the Separators child property, the process affects only types in the type tree that also have those same Number, Character, Decimal type property settings. The Separators property is a type property that applies to the specific Presentation types, Decimal and Integer. For those types that have the Presentation type property setting of Zoned, the Separators property does not apply; it is not an available type property.

A given category type has the following type property settings: Restrictions._>Value and for its child property, Ignore case._>No. The Ignore case type property has no children properties. When you select the child Ignore case type property setting of No to be propagated, the process propagates the Ignore case setting of No to all types in the category whose parent Restrictions property setting is Value. The Ignore case property is a type property that applies to a specific Restrictions type, Value. For those types that have the Restrictions type property setting of Range, the Ignore case property does not apply; it is not an available type property.

Related tasks

- [Propagating type properties](#)

Propagating type properties

These are the instructions for propagating type properties in type trees.

About this task

Use Propagate to pass a type property setting from a type to its subtypes.

Procedure

To propagate a type property:

1. Open the Properties view for a type.
2. Right-click the property you want to propagate, such as Item Subclass > Restrictions, and select Propagate from the context menu.
3. When a confirmation message is displayed, select Yes.
Upon completion of the propagation, a message is displayed that confirms the number of types that the propagation process has changed.
4. Click OK to close the message box.

Results

The setting of the specified type property from the selected type is propagated to all of the appropriate subtypes in the type tree.

Related reference

- [Propagation of type properties common to all types](#)
 - [Propagation of type properties for specific types](#)
-

Error detection and recovery

- [Error detection](#)
 - [Error recovery](#)
-

Error detection

During map execution, the input data is compared to the data definition in the type tree. If the data does not match the definition, it is invalid or "in error".

To validate a data object as belonging to a certain type, the data must be matched to its type definition. For data to be valid, the following must be true:

- The data must have the properties that were defined for the type.
- If the type is an item that has restrictions, the data object must match one of the restrictions.
- If the type is a group, the components of the data object must match those defined in the group view, and each component rule must evaluate to "true" at map execution time.

Ultimately, all data objects, no matter how complex, consist of items because items represent the smallest unit of data. When all of the items that collectively comprise a component are found, the component has been "found."

When executing a map, invalid data is recorded in the trace file. You can decide what you want the system to do when errors are encountered: you can map the errors to an output and also have the system ignore invalid data. For methods on ignoring invalid data, see ["Restart attribute"](#).

- [How error detection works](#)
 - [Existence indicators](#)
 - [Existence versus presence of components](#)
-

How error detection works

When data is validated, the system may encounter data that does not match its type definition. A combination of how that type is defined and what data shows up determines what happens next. As data is validated, the system tracks valid data objects, data objects that exist but are in error, and data that cannot be recognized as belonging to any type.

If a data object is valid, the information is recorded, as needed, and validation continues.

If the system is looking for a data object of a particular type, and something in the data does not exactly match the type definition, the system bases the determination on whether to continue validation on whether that data object is known to exist.

If a data object contains enough information to determine its existence, errors associated with that data object are recorded and validation continues.

If a data object exists but contains errors, it is marked as an error. If a restart point is specified for the component of the type in error, these errors are ignored and validation continues.

If a data object does not contain enough information to determine its existence, the following occurs:

- If no validation recovery mechanism is specified, validation is stopped because it will get lost if it continues. Validation recovery mechanisms are discussed in the section ["Error Recovery"](#).
- If recovery mechanisms are specified, the system returns to the nearest restart point and resets what it is looking for. It proceeds by examining the data byte-by-byte until it either recognizes something or reaches the end of the data stream. All rejected bytes (from the first byte not associated with a type to the last unrecognizable byte) are collectively marked as an unidentified foreign object (UFO). A UFO is data in error with no valid data contained within it.

In summary, the data object of an input card could be valid, yet contain errors. If the data object of any input card is invalid, output data is not built. If all input card data objects are valid, the input is mapped to the output based on the map rules. Operators, most functions, and map references operate on valid data. To map invalid data, use the **REJECT** function.

Existence indicators

When the source or destination of a data object exists, the system knows that the entire data object of that source or destination exists. For example, if a source specified to contain a transaction is a file, and that file exists, the transaction exists. On the other hand, if you get a `Source not available` message, the data object of that source does not exist because the source itself does not exist.

When the data object of a source or destination exists, specific information about errors will appear in the data. For example, you might have any of the following:

- The data object is valid because it conforms to its type definition.
- The type of the data object exists, but has no content.
- The data object is in error because it does not conform to its type definition.
- The data object is valid but contains errors. You indicated that you wanted to ignore certain errors.

In addition to these conditions, the system can also tell if there is any unknown data remaining after the card object has been recognized. This can occur if there really is "junk" at the end. It may also be that enough data was identified to determine the status of the entire object and the data at the end could not, for some reason, be determined to have anything to do with the data object.

When a data object exists, the status of that object is determined by validating its type properties. If an entire data object is an item, it is easy to determine what the status is: whether it is valid or whether it does not match its item format, whether its value is not one of the specified restrictions, or whether it is missing an initiator or terminator. If the source or destination exists and the entire data object is an item, there is nothing else it could possibly be.

If the type of the entire data object is a group, the status of each component must be determined. The system uses existence indicators to determine whether a group component exists. If a component exists, the system can also tell you its status. If a component is required in another component that exists, the system will notify you if that required component is missing.

Existence versus presence of components

A component exists if the data object in which it is contained exists and if the component takes up space (if there is at least one byte in the data stream representing that component).

For example:

- If a delimiter appears as a placeholder for that component.
- If the type of that component has an initiator that distinguishes that component from any other component that may appear at that position in a data stream, the component exists if the initiator is there.
- If the type of that component is a group with an identifier and all components up to and including that identifier have been found to be valid.
- If the type of the component is an item with a restriction list and one of its restrictions appear in the data.
- If the type of the component is an item with a **Padded To** length specified and the item contains at least the number of bytes in the **Pad To** length.
- If the type of an optional component is valid, but the component rule evaluates to FALSE, the component only exists if there is a syntactic placeholder.

Data might exist, but might not be present. The presence of data is determined by whether the data has content. The following definitions are used in other parts of the documentation:

EXISTS

Something in the data represents that data object.

CONTENT

The data contains at least one byte other than a syntactical placeholder, such as a pad character or delimiter.

PRESENT

The data object exists and has content.

ABSENT

The data object does not exist or it exists and has no content.

Required components must exist. They do not necessarily have to be present. For example, a required item with no minimum content size can exist, be absent, and still be valid. However, if some minimum content size is specified, a required component is not valid if there is no minimum content.

The existence of optional components depends on the context in which they are used. For example:

- In a delimited group, an optional component must exist if data follows.
- In a fixed group, an optional component must exist if trailing white space is required. If no trailing white space is required, an optional component must exist if data follows.
- In an implicit group, an optional component is not required.

Unlike a required component, when an optional component exists, it might not have content - even when the minimum content specification is greater than zero bytes. The following table explains the conditions that must be met for required and optional components.

	EXISTS	CONTENT	PRESENT	ABSENT
Required Component	Must exist in any context	Must conform to content specification	Must be present if minimum content specification is greater than zero bytes.	Can be absent only if minimum content specification is zero.
Optional Component	Depends on context	Need not have content	Not necessary	Can be absent

Error recovery

It is good practice to map invalid data when it is encountered. For example, you have a file of records and some records are invalid. If your structure is designed to map invalid data when it is encountered, processing can continue without interruption.

To map invalid data of a particular object, you can assign the **Restart** attribute. The **Restart** attribute allows the processing of input data to continue when a data object of a component is invalid. Errors are ignored for invalid data object during validation. Using error-handling functions, you can map the invalid data.

- [Using the **Restart** attribute](#)

- [How the Restart attribute works](#)
- [Mapping invalid data](#)

Using the Restart attribute

The Restart attribute provides instructions for handling errors encountered in a data stream. The Restart attribute gets assigned to a component and takes effect during data validation.

During data validation, the Restart attribute identifies a "start over" point when an unidentified foreign object (UFO) is encountered in the data. All unrecognized data is considered an error of the type with the Restart attribute assigned.

If you are mapping data of a component with a Restart attribute in place, only the valid occurrences of that component are mapped. For example, suppose your input is a file of records, and the **Record** component of **File** has the Restart attribute. Suppose some of the records are invalid. When you map the **Record** objects, only the valid records are mapped. To map the invalid records (for example, to an error file) use an error-handling function, such as REJECT, in the map rule.

For information about error-handling functions, see the Functions and Expressions documentation.

Typically, you would assign the Restart attribute to a component that has a series range. For example, (1:20) or (s), and the objects in the series are independent of one another. If each record in your file is independent from the others, it makes sense to ignore an error if it encounters an invalid one. A bad record does not make the next record meaningless. But suppose the records are related. Maybe they are records related to one purchase order; if any record is bad, you want to stop after validation because it is important to have all records. Losing any records would make the overall data incomplete.

For additional information about using the **Restart** attribute, see the Map Designer documentation.

To assign the Restart attribute to a component

Tip: To capture all errors of the component using Restart, place the Restart attribute at the highest level possible in the type tree.

1. From the type tree editor, double-click the component. The component view opens.
2. Right-click the component and select Restart from the context menu. You can remove the Restart attribute by performing the same action again.

JSON native schema restart

JSON schema fields with `_V` following the name and fields starting with `oneOf`, `allOf` and `anyOf` are the virtual fields.

Virtual JSON fields do not appear in the data and have no syntax. Syntax is controlled by real fields above these virtual fields.

Real fields are fields that appear in the JSON data. It is impossible to restart to a data location that does not appear in the data.

Selecting a virtual field for restart may fail. Use a real field either above or below the virtual fields for restart.

Related tasks

- [Restart attribute](#)

How the Restart attribute works

The Restart attribute can do the following tasks:

- Identify valid data objects that contain objects in error.
- Tell the system where to start over when a unidentified foreign object (UFO) is encountered in the data during the validation process.
All unrecognized data is considered to be an error of the type with Restart assigned to it.
- Identify the UFOs and existing data objects in error that are ignored when mapping input to output.

When an invalid data object is a component that does not have the Restart attribute assigned, that component is marked in error. If components from the beginning of the data are marked in error because no Restart attribute is assigned, the following results occur. Processing stops after validation, and input data is not mapped to output.

Do not put the Restart attribute on a required component. There must be a number of valid instances that is enough to cover all of the required components. If you have a required component that is not valid, the Restart attribute does not validate the data.

When the Restart attribute is added to components that have only one mandatory element, it is ignored. Adding Restart to the parent level works only when there are no children that have one mandatory value. In this scenario, the ISERRORS and REJECT functions would report the first error. But if there are other errors, they are not captured because Restart stopped and restarted at the next parent level.

The Restart attribute is supported on components of Xerces type trees.

For more information about using the **Restart** attribute, see the Map Designer documentation.

Mapping invalid data

You can map the invalid or rejected data which you can use to determine what is wrong with the data. To do this, you use the restart attribute, in conjunction with the **REJECT**, **CONTAINSERRORS**, and **ISERROR** functions in map rules. For information on using these functions, see the Functions and Expressions documentation.

Type Tree Analyzer

Use the type tree analyzer to analyze your type definitions.

The type tree analyzer analyzes type definitions and ensures internal consistency. For example, if you defined a group as fixed, but accidentally defined one of its items with no **Padded To** length, errors occur during analysis.

The analyzer checks your data definitions for logical consistency. It does not compare your definitions to your actual data. The resulting analyzer messages indicate whether your type tree definitions are acceptable; not whether they match your data.

- [Mapping effects](#)
- [Internal consistency](#)
- [When to Analyze Structure or Logic](#)
- [To analyze a type tree](#)
- [Error and warning messages](#)

Related information

- [Distinguishable objects](#)

Mapping effects

The analyzer helps you define your data by locating objects in your input data and creating the objects in your output data. The analysis indicates whether there is something in your definition that may prevent a correct mapping of your data.

If you do not analyze a tree before you map the data or you analyze a tree and do not resolve the errors, you will be warned that you may receive unpredictable results when you map.

Internal consistency

The analyzer checks the logic of your data definitions. For example, suppose you defined a group **PO** as fixed and consists of **Line(s)**. For the **PO** to be fixed, it cannot have an indefinite number of **Lines**. An analysis error would occur, indicating that you defined **PO** as fixed but it has a variable number of components.

When to Analyze Structure or Logic

You can choose to analyze the structure or the logic of your type definitions, or both.

- [Logical analysis](#)
Logical analysis addresses the integrity of the relationships that you define.
- [Structural analysis](#)
Structural analysis addresses the integrity of the underlying database.

Related tasks

- [To analyze a type tree](#)

Logical analysis

Logical analysis addresses the integrity of the relationships that you define.

Logical analysis detects, for example, undefined components, components that are not distinguishable from one another, item restrictions that do not match the properties of that item, and circular type definitions. The analyzer also checks delimiter relationships to each other and to components, undefined inherited relationships, and logic errors contained in component rules.

Structural analysis

Structural analysis addresses the integrity of the underlying database.

Generally, you should not encounter structural analysis errors. Structural analysis might be able to detect and possibly correct defects caused by system environment failures.

To analyze a type tree

About this task

Follow this procedure to analyze a type tree.

Procedure

To analyze a type tree:

1. Select a type and open the properties.
2. In the type tree editor, right-click the root type.
3. From the context menu, choose Analyze...
 - Logic and Structure,
 - Logic Only, or
 - Structure Only.

The Analysis Results view opens.

The Analysis Results view displays the name of the type tree, the type of analysis, and the status of the process. When the analysis process completes, the number of errors and warnings are displayed.

Select the option under Transformation Extender preferences to change the default behavior to automatically save the type tree after analysis.

Error scenario

After analysis, the Analysis Results dialog displays error L201.

The error concerns the type **File**. **File**'s first component, **Record(s)** is a series. The way the data is defined does not allow one **Record** to be distinguishable from the next **Record** in the series.

There are a number of reasons why the records might not be distinguishable. To see how you can define **Record** to solve the problem, you can look at the list of things that make objects of a component distinguishable in ["Distinguishable Objects"](#).

A good way to figure out how to resolve an error is to look at the data. When you look at the data, it is clear where a record ends. Ask the question: *How can I tell the difference between one record and the next?* Perhaps each record ends with a carriage return/linefeed.

In this case, perhaps you forgot to define CR/LF as the terminator of **Record**.

To resolve the problem, open the Properties for the **Record** type and define the terminator. After you define a terminator and analyze the tree again, no errors should occur.

Related reference

- [Customizing the Transformation Extender Development perspective](#)

Related information

- [When to Analyze Structure or Logic](#)

Error and warning messages

Analysis results might contain error and warning messages.

Warnings are relatively insignificant. They indicate an inconsistency that occurred when you changed something in the tree that was resolved. For example, if you change a group to an item, the analyzer removes the components of that type because items do not have components. To remind you of this change, the analyzer issues a warning.

Errors are important. An error is a problem in your type definitions that you should correct. An error may result in unpredictable results in your mapping.

Occasionally there are errors that prevent the analysis of the rest of your definitions (for example, when a component type cannot be found in the tree). **Analysis halted before completion** is displayed. This error might be due to one of the following:

- Undefined COMPONENTs found: ending analysis.
- Circular reference found in COMPONENT list: ending analysis.

When this occurs, click Results, correct the errors, and analyze the tree again.

Related information

- [Return codes and error messages](#)

Distinguishable objects

Differences can be identified for objects in a data stream. This information is helpful when a type tree analysis produced an error message concerning objects that are not distinguishable.

- [Objects in a data stream](#)
- [Type tree analyzer and distinguishable objects](#)

- [Bound types](#)
- [Bound components](#)
- [Group starting set](#)
- [Group unbound set](#)
- [Initiator-distinguishable types](#)
- [Distinguishable objects of the same component](#)
- [Content-distinguishable components](#)
- [Content-distinguishable types](#)
- [Ending-distinguishable types](#)
- [Distinguishable data objects of an implicit group](#)
- [Distinguishable data objects of an explicit group](#)
- [Distinguishable syntax objects](#)

Related concepts

- [Distinguishable components of an implicit group](#)

Related information

- [Type Tree Analyzer](#)

Objects in a data stream

A data stream is a byte-by-byte flow of data. Objects in a data stream include both data objects and syntax objects. Syntax objects indicate where a data object begins or ends. They include separators, initiators, terminators, delimiters, release characters, and pad characters. Sometimes, data values are used to identify where another data object begins or ends.

It might be necessary to distinguish between data objects in a component series, data objects of different types, or even syntax objects.

Type tree analyzer and distinguishable objects

The type tree analyzer indicates whether your data definitions are sufficient to distinguish the objects in your data stream. The following discussion explains how you can define your data so that the objects that need to be distinguishable are distinguishable. Chances are, if you analyzed your tree and received a message that involves distinguishable objects, you need to define the types differently or more specifically.

Bound types

A type is bound if its definition makes it clear where an instance of that type ends. If a type is bound, different objects of that type can be distinguished in a data stream. A bound type is easier to distinguish between an object of that type and an object of another type. The following tables describe how types may be bound.

An object of this type:

Is bound if any of the following is true:

Item

It is padded to a fixed size or its minimum and maximum content size are equal.
It has a terminator.

It has an include restriction list.

Partitioned item

Each non-partitioned item in the subtree is bound.

Sequence Group

It has an explicit fixed format.
It has a terminator.

Its last component is bound.

It is postfix delimited and its last component has a fixed range. For example, **Comment Field (3:3)** has a fixed range of **(3:3)**.

Partitioned Group

Each non-partitioned group in its subtree is bound.

Choice Group

It has a terminator.
The type of each selection component is bound.

Unordered Group

It has a terminator.

Bound components

A component is bound if it is possible to tell if a data object belongs to that component without comparing it to a component that follows it.

- [Component of a fixed group](#)
 - [Component of an explicit delimited group](#)
 - [Component of an implicit group](#)
 - [Component of a choice group](#)
 - [Component of an unordered group](#)
-

Component of a fixed group

Condition

Example

A range maximum is specified (it is not "s"), and its type is either a fixed group or an item whose length is fixed.

The component **InventorySection (0:3)** is bound because it is assumed there will be spaces in the data stream for 3 **InventorySections**.

Component of an explicit delimited group

Condition

Example

A range maximum is specified (it is not "s") if a component of the same group follows.

In the component list, **Security Sequence (0:10) Trailer** is bound in an explicit delimited group. It is assumed there will be a delimiter for all 10 **Security Sequences** in the data stream because **Trailer** is required.

The component is the last one and the explicit group has a terminator.

In the component list, **Security Sequence (s)** is bound in an explicit delimited group with a terminator. The system assumes the terminator will appear to indicate that there are no more **Security Sequences**.

Component of an implicit group

Condition

Example

Its range is fixed and its type is bound.

The component **Inventory Record (3:3)** is bound, if the type **Inventory Record** is bound according to any of the conditions listed under "[Bound Types](#)".

It has a component rule that binds it.

The component **PO Record (s)** is bound, if it has the following component rule which binds it:

PO# Field:PO Record = PO# Field:PO Record[LAST]

The type tree analysis checks only that the component has a rule that refers to the component itself. The analysis does not check that the rule binds the component. You must ensure that the rule is one that binds the component.

It is sized (using the Sized attribute) by the component that precedes it.

The component **Name Field (0:2)** is bound, if the previous component in the group has the Sized attribute.

Component of a choice group

Condition

Example

A component is bound if the type of a component is bound.

The component **Customer Record** is bound if the type **Customer Record** is bound.

Component of an unordered group

Condition

Example

A component is bound if its range is fixed and its type is bound.

The component **Address Field** is bound if its range is fixed, for example **(3:3)**, and the type **Address Field** is bound.

Group starting set

Data objects in a data stream need to be distinguishable when they could belong to two different groups. Specifically, the difference between the data that can come first in one group and the data that can come first in the other group. All of the possible types of data that may appear first in a group are referred to as the group's starting set.

The following describes the starting set of a group based on its group format:

Group Format	Starting Component Set
--------------	------------------------

Group Format	Starting Component Set
Explicit <ul style="list-style-type: none"> • Delimited • Fixed 	Includes the type of its first component.
Implicit <ul style="list-style-type: none"> • Sequence • Unordered 	<ul style="list-style-type: none"> • Includes the type of all components up to and including the first component that has a minimum range of at least one. • Includes the type of each component.
Choice	Includes the type of each component.

Group unbound set

Based on the nature of a bound group, the end of a bound group is determined without analyzing the data that follows it. However, if a group is not bound, the data that belongs to that group must be distinguishable from data that belongs to another type that might follow it in the data stream.

The unbound set of a choice or unordered group consists of the type of each component. The unbound set of a partitioned group consists of the unbound set of each unbound partition.

- [Unbound set of a sequence group](#)

Unbound set of a sequence group

The unbound set does not include a type that could come last, if that type is a required occurrence. For example, if the type **Comment Record** could appear last and the component is specified as **Comment Record (2:2)**, it is not in the unbound set, because the two occurrences of **Comment Record** are required. However, if **Comment Record** could appear last and the component is specified as **Comment Record (1:2)**, then **Comment Record** is in the unbound set, since its second occurrence is optional.

To determine the unbound set of a group, start with the last component of the group and proceed backward up the component list. If a component is unbound or has a minimum range of zero, continue up the list. Stop at the first component that is bound or that is unbound but has a minimum range greater than zero. Essentially, a group's unbound set is everything that is not clearly defined at the end of the group.

Initiator-distinguishable types

Initiator-distinguishable types are used during validation to determine existence of a type object based on the existence of its initiator.

- [Determining if a component is initiator-distinguishable from its following set](#)
- [Determining if a partition is initiator-distinguishable from its following set](#)
- [Determining if two types are initiator-distinguishable](#)

Determining if a component is initiator-distinguishable from its following set

A component is initiator-distinguishable from its following set if:

- The component is not a member of the identifier set and
- The type of the component has an initiator and
- The following set is empty or
- The type of the component is initiator-distinguishable from each type in its following set.

Type trees are analyzed to determine if components are initiator-distinguishable. Each implicit sequence group, choice group, and unordered group is analyzed to determine if their components are initiator-distinguishable. A component is marked as initiator-distinguishable when that component is initiator-distinguishable from its following set. The basis for this determination is found in ["Determining If Two Types are Initiator-Distinguishable"](#).

Determining if a partition is initiator-distinguishable from its following set

In a partitioned type, a partition is initiator-distinguishable from its following set if:

- The type of a partition has an initiator and
- The following set is empty or
- The type of the partition is initiator-distinguishable from each partition in its following set and the following set of a partition is the type of each partition that may follow.

Type trees are analyzed to determine if partitions are initiator-distinguishable. Each partitioned type is analyzed to determine if its partitions are initiator-distinguishable.

Determining if two types are initiator-distinguishable

The following table lists ways two types may be initiator-distinguishable. This is helpful if data validation errors indicate a type does not exist.

Type1	Type2	How to define them as initiator-distinguishable
item	item	If Type1 and Type2 have an initiator, and the initiators are different.
item	sequence group	<p>Either:</p> <ul style="list-style-type: none"> Type1 and Type2 both have an initiator and the initiators are different. <p>or</p> <ul style="list-style-type: none"> Type1 has an initiator, Type2 does not, Type2 has no delimiter, and Type1 is initiator distinguishable from type of each component in the starting component set of Type2.
item	choice group or unordered group	<p>Either:</p> <ul style="list-style-type: none"> The Type1 and Type2 both have an initiator and the initiators are different. <p>or</p> <ul style="list-style-type: none"> Type1 has an initiator, Type2 does not, Type2 has no delimiter, and Type1 is initiator distinguishable from type of each component of Type2.
item	partitioned item or partitioned group	Type1 has an initiator and is initiator distinguishable from each partition of Type2.
partitioned item	item or sequence group	Type1 has an initiator and each partition is initiator distinguishable from Type2.
partitioned item	partitioned item or partitioned group	Type1 has an initiator and each partition is initiator distinguishable from each partition of Type2.
sequence group	item	Type1 has no identifier, Type1 and Type2 both have initiators, and the initiators are different.
sequence group	sequence group	<p>Type1 has no identifier and</p> <ul style="list-style-type: none"> Type1 and Type2 both have initiators and the initiators are different <p>or</p> <ul style="list-style-type: none"> Type1 has an initiator, Type2 does not have an initiator, Type2 has no delimiter, and Type1 is initiator distinguishable from the starting component set of Type2.
sequence group	choice group or unordered group	<p>Type1 has no identifier and</p> <ul style="list-style-type: none"> Type1 and Type2 both have initiators and the initiators are different. <p>or</p> <ul style="list-style-type: none"> Type1 has an initiator, Type2 does not have an initiator, Type2 has no delimiter, and Type1 is initiator distinguishable from the type of each component of the Type2.
sequence group	partitioned item	Type1 has no identifier and Type1 must be initiator distinguishable from each partition of Type2.
sequence group	partitioned group	Type1 has no identifier and the Type1 must be initiator distinguishable from each partition of Type2.
partitioned group	partitioned item	Type1 has no identifier and each partition Type1 must be initiator distinguishable from Type2.
partitioned group	sequence group	Type1 has no identifier and each partition of the Type1 must be initiator distinguishable from Type2.
partitioned group	partitioned item	Type1 has no identifier and each partition of Type1 must be initiator distinguishable from each partition of Type2.
partitioned group	partitioned group	Type1 has no identifier and each partition of Type1 must be initiator distinguishable from each partition of Type2.
choice group or unordered group	item	Type1 and Type2 both have an initiator and the initiators are different.
choice group or unordered group	partitioned item	Type1 is initiator-distinguishable from each partition of Type2.
choice group or unordered group	choice group or unordered group	<p>Either:</p> <ul style="list-style-type: none"> Type1 and Type2 both have an initiator and the initiators are different. <p>or</p> <ul style="list-style-type: none"> Type1 has an initiator and Type2 does not, Type2 has no delimiter, and Type1 is initiator-distinguishable from the type of each component of Type2.
choice group or unordered group	sequence group	<p>Either:</p> <ul style="list-style-type: none"> Type1 and Type2 both have an initiator and they are different. <p>or</p> <ul style="list-style-type: none"> Type1 has an initiator and Type2 has no initiator and no delimiter, and Type1 is initiator-distinguishable from the type of each component in the starting component set of Type2.
choice group or unordered group	partitioned group	Type1 is initiator distinguishable from each partition of the Type2.

Distinguishable objects of the same component

When a component has a series range, for example, (1:10) or (s), one occurrence of that component must be distinguishable from the next occurrence of that same component.

Different data objects of a component are distinguishable if any of the following is true:

- The component type is bound.
- The component type is a non-partitioned group with an unbound set that is content-distinguishable from the component type as a whole.
- The component type is a partitioned group and the unbound set of each non-partitioned group in its subtree is content-distinguishable from the component type as a whole.

Content-distinguishable components

A component is distinguishable from its following set when the component is:

- Initiator-distinguishable from its following set.

or

- Content-distinguishable from its following set.

A component is content-distinguishable from its following set when:

- The following set is empty.

or

- The type of the component is content-distinguishable from each component in its following set.

Content-distinguishable types

The following table lists the ways in which two types may be content-distinguishable. The table is helpful particularly if you analyzed your tree and received an error indicating that two types are not distinguishable. Look up the combination of types in the first two columns and read the list of ways to define them that would make them distinguishable.

For example, if you get an error indicating that **X** is not distinguishable from **Y**, where **X** is an item, and **Y** is a partitioned group, you would find the row in the table where **Type1** is an item and **Type2** is a partitioned group.

Remember that the order in which you compare two types matters. When you ask the question "Is type A content-distinguishable from type B?", this is not the same question as "Is type B content-distinguishable from type A?"

If you are trying to determine whether two types are distinguishable and you follow the guidelines in the following tables, you may encounter a situation in which you are comparing a type to itself. A type is never distinguishable from itself.

Another thing to keep in mind is that the context in which a type is used matters. When you ask the question, "Is type A, as a component of type C, content-distinguishable from type B?" this is not the same question as, "Is type A, as a component of type D, content-distinguishable from type B?"

The following table shows how to define types as content-distinguishable.

Type 1	Type 2	How to define them as content-distinguishable
item	item	<p>The first component or partition is either:</p> <ul style="list-style-type: none">• An item and marked as initiator distinguishable and <p>Both items are different partitions of the same partitioned subtree, or The second type has an initiator and those initiators are mutually exclusive, or Both types have the same initiator and the value of the first item is distinguishable from the value of the second item, or The second type has no initiator and the initiator of the first type is distinguishable from the value of the second type.</p> <ul style="list-style-type: none">• An item and not marked as initiator distinguishable and <p>Both items are different partitions of the same partitioned subtree, or The first item has an initiator, second has an initiator and initiator value of first is distinguishable from initiator value of the second. Both types have initiators and they are the same, or both types have no initiator and the value of the first item is distinguishable from the value of the second item. The first item has an initiator, second does not, and value of initiator distinguishable from the value of second item. The first item has no initiator, second does, and the first item's value is distinguishable from the value of the second item's initiator.</p>
item	sequence group, choice group, or unordered group	<p>The first component or partition is either:</p> <ul style="list-style-type: none">• An item marked as initiator distinguishable and <p>The second type has an initiator and those initiators are mutually exclusive, or The second type has no initiator and the item is initiator-distinguishable from each type in the starting component set of the group.</p> <ul style="list-style-type: none">• An item not marked as initiator distinguishable and <p>The item and group both have an initiator and the initiator of the item is distinguishable from the initiator of the group, or The item has no initiator, the group has an initiator, and the item's value is distinguishable from the value of group's initiator, or Both types have initiators and they are the same or both types have no initiator, and the item's value is distinguishable from the type of each component in the starting component set of the group, or The item has an initiator, the group has no initiator, and the item is content-distinguishable from the type of each component in the starting component set of the group, or The item has no initiator, the group has an initiator, and the item's value is distinguishable from each type in the starting component set of the group.</p>

Type 1	Type 2	How to define them as content-distinguishable
item	partitioned item or partitioned group	The first component or partition is content-distinguishable from each partition of the second type.
partitioned item	item	Each partition of the first type is content-distinguishable from the second type.
partitioned item	sequence group	Each partition of the first type is content-distinguishable from the second type.
partitioned item	choice group	Each partition of the first type is content-distinguishable from second type.
partitioned item	unordered group	Each partition of the first type is content-distinguishable from second type.
partitioned item	item	Each partition of the first type is content-distinguishable from second type.
partitioned item	partitioned group	Each partition of the first type is content-distinguishable from second type.
group	item	Each partition of the partitioned group is content-distinguishable from the item.
partitioned group	partitioned item	Each partition of the partitioned group is content-distinguishable from each partition of the item.
partitioned group	sequence group	Each partition of the partitioned group is content-distinguishable from the second group.
partitioned group	partitioned group	Each partition of the first partitioned group is content-distinguishable from each partition of the second partitioned group.
partitioned group	choice group	Each partition of the partitioned group is content-distinguishable from the choice group.
partitioned group	unordered group	Each partition of the partitioned group is content-distinguishable from the unordered group.
sequence group	item	<p>The first component is either:</p> <ul style="list-style-type: none"> • A group marked as initiator distinguishable and <p>The item has an initiator and the initiator value of the group is distinguishable from the initiator value of the item, or The second type has no initiator and the initiator value of the first type is distinguishable from the value of the second type.</p> <ul style="list-style-type: none"> • A group not marked as initiator distinguishable and <p>The group has an initiator, the item has an initiator and the initiator value of the group is distinguishable from initiator value of the item, or The group has an initiator, the item does not, and value of the initiator is distinguishable from value of the item, or Both types have initiators and they are the same, or both types have no initiator and each type in the starting component set of the group is content-distinguishable from the value of the second item.</p>
sequence group	sequence group	<p>The first component is either:</p> <ul style="list-style-type: none"> • A group marked as initiator distinguishable, and <p>Both groups are different partitions of the same partitioned subtree, or The second type has an initiator and those initiators are mutually exclusive, or The second type has no initiator and the first group is initiator-distinguishable from each type in the starting component set of the second group.</p> <ul style="list-style-type: none"> • A group not marked as initiator distinguishable, and <p>Both groups are different partitions of the same partitioned subtree, or Both groups have an initiator and the initiator value of the first group is distinguishable from the initiator value of the second group, or The first group has no initiator, the second group has an initiator and the type of each component in the starting component set of the first group is content-distinguishable from the second group, or The first group has an initiator, the second group has no initiator and the first group is content-distinguishable from the type of each component in the starting component set of the group. Both types have initiators and they are the same or both types have no initiator and The first group is content-distinguishable from the starting component set of the second group, or The second group is content-distinguishable from the starting component set of the first group, or The starting component set of the first group is content-distinguishable from the starting component set of the second group.</p>
sequence group	choice group or group	<p>The first component is either:</p> <ul style="list-style-type: none"> • A group marked as initiator distinguishable and <p>Both groups are different partitions of the same partitioned subtree, or The second type has an initiator and those initiators are mutually exclusive, or The second group has no initiator and the first group is initiator-distinguishable from the type of each component of the second group.</p> <ul style="list-style-type: none"> • A group not marked as initiator distinguishable and <p>Both groups are different partitions of the same partitioned subtree, or Both groups have an initiator and the initiator of the first group is distinguishable from the initiator of the second group, or The first group has an initiator, the second group has no initiator and the first group is content-distinguishable from the type of each component of the second group, or The first group has no initiator the second group has an initiator, and each component in the starting component set of the first group is distinguishable from the second group, or Both types have initiators, either the initiators are the same or both types have no initiator, and each component in the starting component set of the first group is content-distinguishable from the type of each component of the second group.</p>
sequence group	partitioned item or partitioned group	<p>The first component is either:</p> <ul style="list-style-type: none"> • A group marked as initiator-distinguishable and the group is initiator-distinguishable from each partition of the second type, or • A group not marked as initiator-distinguishable and the group is content-distinguishable from each partition of the second type.

Type 1	Type 2	How to define them as content-distinguishable
choice group or unordered group	item	<p>The first component is either:</p> <ul style="list-style-type: none"> A group marked as initiator distinguishable and <ul style="list-style-type: none"> The item has an initiator and the initiator value of the group is distinguishable from the initiator value of the item, or The second type has no initiator and the initiator value of the first type is distinguishable from the value of the second type. A group not marked as initiator distinguishable and <ul style="list-style-type: none"> Both types have an initiator and the initiator value of the group is distinguishable from initiator value of the item, or The group has an initiator, the item does not, and the value of the initiator is distinguishable from value of the item, or The group has no initiator, the item has an initiator and the starting component set of the group is content-distinguishable from the item, or Both types have initiators and they are the same, both types have no initiator and the type of each component of the group is distinguishable from the value of the second item.
choice group or group	sequence group	<p>The first component is either:</p> <ul style="list-style-type: none"> A group marked as initiator distinguishable and <ul style="list-style-type: none"> Both groups are different partitions of the same partitioned subtree, or The second type has an initiator and those initiators are mutually exclusive, or The second type has no initiator and the first group is initiator-distinguishable from each type in the starting set of the second group. A group not marked as initiator distinguishable, and <ul style="list-style-type: none"> Both groups are different partitions of the same partitioned subtree, or Both groups have an initiator and the initiator value of the first group is distinguishable from the initiator value of the second group, or The first group has an initiator, the second group has no initiator and the first group is content-distinguishable from the type of each component in the starting component set of the second group. The first group has no initiator, the second group has an initiator and the type of each component of the first group is content-distinguishable from the second group, or. Both types have initiators and they are the same or both types have no initiator, and the type of each component of the first group is content-distinguishable from the type of each component in the starting component set of the second group.
choice group or group	choice group or group	<p>The first component is either:</p> <ul style="list-style-type: none"> A group marked as initiator distinguishable and <ul style="list-style-type: none"> Both groups are different partitions of the same partitioned subtree, or The second type has an initiator and those initiators are mutually exclusive, or The second group has no initiator and the first group is initiator-distinguishable from the type of each component of the second group. A group not marked as initiator distinguishable and <ul style="list-style-type: none"> Both groups are different partitions of the same partitioned subtree, or Both groups have an initiator and the initiator of the first group is distinguishable from the initiator of the second group, or The first group has an initiator, the second group has no initiator and the first group is content-distinguishable from the type of each component of the second group, or The first group has no initiator the second group has an initiator, and each component of the first group is distinguishable from the second group, or Both types have initiators and the initiators are the same or both types have no initiator, and the type of each component of the first group is content-distinguishable from the type of each component of the second group.
choice group or group	partitioned item or partitioned group	<p>The first component is either:</p> <ul style="list-style-type: none"> A group marked as initiator-distinguishable and the group is initiator-distinguishable from each partition of the second type, or <ul style="list-style-type: none"> A group not marked as initiator-distinguishable and the group is content-distinguishable from each partition of the second type.

Related concepts

- [Guidelines for defining an implicit delimited sequence](#)

Ending-distinguishable types

Ending-distinguishability is used to determine if the end of a data object is distinguishable from the start of any other data object that could be next in the data.

The following table describes how two types may be ending-distinguishable. This is helpful if you are validating data and you receive a message that says a type exists, but it belongs to the wrong component.

Type1	Type2	How to define them as ending-distinguishable
item	item or sequence group or choice group or unordered group	Type1 is bound or the value of Type1 is content-distinguishable from Type2.
item	partitioned item or partitioned group	Type1 is bound or the value of Type1 is content-distinguishable from each partition of Type2.

Type1	Type2	How to define them as ending-distinguishable
sequence group	item or sequence group or choice group or unordered group	<p>Either:</p> <ul style="list-style-type: none"> • Type1 is bound. <p>or</p> <p>For each component in the unbound set of Type1</p> <ul style="list-style-type: none"> • If the component has a fixed range it must be ending-distinguishable from Type2. • If the component range is variable, it must be content-distinguishable from Type2 and ending-distinguishable from Type2.
sequence group	partitioned item or partitioned group	<p>Either:</p> <ul style="list-style-type: none"> • Type1 is bound. <p>or</p> <p>For each component in the unbound set of Type1</p> <ul style="list-style-type: none"> • If the component has a fixed range it must be ending-distinguishable from each partition of Type2. • If the component range is variable, it must be content-distinguishable from each partition of Type2 and ending-distinguishable from each partition of Type2.
choice group or unordered group	item or sequence group or choice group or unordered group	<p>Either:</p> <ul style="list-style-type: none"> • Type1 is bound. <p>or</p> <ul style="list-style-type: none"> • For each component of Type1, <p>The type of that component is bound, or The type of that component is unbound, and that type must be ending-distinguishable from Type2.</p>
choice group or unordered group	partitioned item or partitioned group	<p>Either:</p> <ul style="list-style-type: none"> • Type1 is bound. <p>or</p> <ul style="list-style-type: none"> • For each component of Type1, <p>The type of that component is bound, or The type of that component is unbound, and that type must be ending-distinguishable from each partition of Type2.</p>
partitioned item or partitioned group	item, sequence group, choice group, or unordered group	<p>For each partition of Type1, either:</p> <ul style="list-style-type: none"> • The partition is a partitioned type and ending-distinguishable from the second type, or • The partition of the first type is bound, or • The partition of the first type is unbound and ending-distinguishable from the second type.
partitioned item or partitioned group	partitioned item or partitioned group	<p>For each partition of the first type, either:</p> <ul style="list-style-type: none"> • The partition is a partitioned type and ending-distinguishable from the second type, or • The partition of the first type is bound, or • The partition of the first type is unbound and ending-distinguishable from the second type.

Distinguishable data objects of an implicit group

When a data object belongs to an implicit sequence, determining the component a data object belongs to depends on the format of the sequence, the type definition of a component, and the component properties.

- [Guidelines for defining an implicit delimited sequence](#)
- [Guidelines for defining an implicit sequence that has no delimiter](#)
- [Guidelines for defining an implicit unordered group that is delimited](#)
- [Guidelines for defining an implicit unordered group that has no delimiter](#)

Guidelines for defining an implicit delimited sequence

- If the type of a component has a terminator, that terminator must be different from the delimiter of the explicit group. If the delimiter and terminator are both defined as literal values, a type analysis confirms both have different values.
- The type of a component cannot be a binary item whose length is not fixed or sized.
- If the type of a component or a contained component is not bound, the type of the component cannot have a delimiter that is the same as the delimiter of the implicit sequence.
- If the range of a component is not bound, the type of that component must be content-distinguishable from the type of each component in the following set of that component.

- If the type of a component has both an initiator and terminator, the nested delimiters do not have restrictions. If this is not the case, all contained delimiters must be different.

Related concepts

- [Content-distinguishable types](#)

Guidelines for defining an implicit sequence that has no delimiter

- The type of a component cannot be a binary item whose length is not fixed or sized.
- If the range of a component is bound, but the type of the component is unbound, the type must be ending-distinguishable from each type in the component's following set.
- If the range of a component is not bound, all the following rules apply:
 - The type of the component must be content-distinguishable from each type in the component's following set.
 - If the maximum range for that component is greater than one, and the type is unbound, the type must be ending-distinguishable from itself.
 - If the type of the component is unbound, the type must be end-distinguishable from each type in the component's following set.

Guidelines for defining an implicit unordered group that is delimited

- The type of a component cannot be a binary item whose length is not fixed.
- The type of a component must be content-distinguishable from the type of each other component.
- If the type of a component has both an initiator and terminator, the nested delimiters do not have restrictions. If this is not the case, all contained delimiters must be different.

Guidelines for defining an implicit unordered group that has no delimiter

- The type of a component cannot be a binary item whose length is not fixed.
- The type of a component must be content-distinguishable from the type of each other component.
- If the type of a component is unbound, the type must be ending-distinguishable from the type of each other component.
- If the maximum range is greater than one and the type is unbound, the type must be ending-distinguishable from itself.

Distinguishable data objects of an explicit group

When a data object belongs to an unordered group, distinguishing one data object from another depends on the format of the unordered group and the type definition of each component.

- [Guidelines for defining an explicit fixed group](#)
- [Guidelines for defining an explicit delimited group](#)
- [Objects of a choice group](#)
- [Objects of a partitioned type](#)

Guidelines for defining an explicit fixed group

- In a fixed group, each component must be either an item of fixed size, an item padded to a fixed length, or an explicit fixed group.
- The range maximum for each component must have a specific value; it cannot be **s**. A fixed amount of space is assumed in the data stream for each series member of a component in a fixed group.

Guidelines for defining an explicit delimited group

A component of an explicit delimited group can be a group or item, with the following restrictions:

- If the type of a component has a terminator, that terminator must be different from the delimiter of the explicit group. If the delimiter and terminator are both defined as a literal, analysis confirms if both have different values.
- If a component of an explicit delimited group is a binary item whose length is not fixed, that component must be sized by the component that precedes it.
- If the type of a component, or a contained component, does not have a terminator and the type of the component has a delimiter that is the same as the delimiter of the explicit group, the type of the component *must* be an explicit sequence. In this case, the delimiter appears as a placeholder for every data object if data of the same delimiter follows it.
- If an explicit group has a delimiter, the range maximum of any component other than the last one must have a specific value; it cannot be **s**. A delimiter is assumed in the data stream for each series member of a component of an explicit group with a delimiter.

Objects of a choice group

When a data object belongs to a choice group, which component the data object belongs to is based on the order the components appear in the type definition.

In a choice group, if a component is unbound it must be content-distinguishable from the type of each component that follows it in the component list.

Objects of a partitioned type

When data objects of different types appear in the same place in the data, the types must be distinguishable. The data must be distinguishable in the type definition. When a partition is part of a partitioned object, the process of cycling through the partitioned subtree begins to make the determination of which partition the data object belongs to.

In a partitioned type, each partition must be content-distinguishable from the type of each partition that follows it.

Related concepts

- [Partitioning types](#)

Distinguishable syntax objects

The system must be able to distinguish between different syntax objects contained within a group. To ensure that the syntax objects are distinguishable, use the following guidelines:

- If a component is unbound, make sure its delimiter and any delimiter contained in it is distinguishable from the delimiter of the group. If you get an analysis error, look for missing placeholders or components with a range maximum of **s**.
In a component or a contained component with the same delimiter as the type delimiter, the system assumes that the delimiter appears as a placeholder for every data object if data with the same delimiter follows it.
- If the delimited group has a terminator, make sure that the delimiter and any contained delimiter is distinguishable from the terminator.
The type tree analysis verifies if the above requirement is met in a non-partitioned group.

Utilities for XML

There are XML utilities provided with the Type Designer.

- Use the XML Type Tree/Schema Synchronization Utility to synchronize a type tree with an XML schema or DTD.
- Use the XML Type Tree Compatibility Utility to add non-XML types that were added to the type tree after it was originally created, to the target type tree. The XML Type Tree Compatibility Utility is also available from the command line (dsxmlconv).
- Use Any-2-XML to create a map that can transform any input data into XML output. Any-2-XML is also available from the command line (dtxany2xml)

There is also a Map Migration Utility (dsmapconv) that updates map source files and the associated XML type trees to the current XML format that is available from the command line.

See the Utility Commands documentation for information about all command line utilities.

- [XML Type Tree/Schema Synchronization utility](#)
- [XML type tree compatibility utility](#)
- [Modifying the target type tree](#)
- [Any-2-XML](#)

The Any-2-XML utility is an extension of the type tree Export as Schema functionality that you can use to create a map that can transform any input data into XML output, based on a type tree structure that you select.

XML Type Tree/Schema Synchronization utility

About this task

Use this utility to synchronize a type tree with an XML schema or DTD.

You can use this utility only to update type trees that are generated from a DTD or schema when you use the XML DTD or XML Schema importers.

To synchronize a type tree with an XML DTD or schema:

Procedure

1. Right-click on the root type and select Synchronize from the menu.
The XML Type Tree/Schema Synchronization Utility opens.
2. From the drop-down list, select the importer that was originally used to create the type tree: XML, which is the default setting, or DTD.

3. Click the browse button and choose a type tree.
 4. In the next field, click the browse button and select the modified DTD or schema file to which you want to synchronize the type tree.
The Create backup file option is enabled by default and the location for the backup file is automatically displayed (based on the type tree you selected). You can change the location and file name extension if wanted.
 5. If applicable, select the national language of the original type tree.
 6. Click Next to begin synchronization.
- The following list includes the possible results:
- If no modifications were made to the original type tree created from the DTD or schema, the synchronization process completes successfully.
 - If modifications were made to the original type tree created from the DTD or schema, they are displayed in the left pane of the utility in red text. You can incorporate any differences into the new (target) type tree.
- For example, you can take an attribute that was added to the original type tree and add it to the target type tree (see "Modifying the Target Type Tree").
7. Change as needed and click Next.
- When you see that the type tree was synchronized successfully, click Finish to close the utility.

XML type tree compatibility utility

About this task

If you have type trees that are generated with the XML Schema or DTD importers of IBM Transformation Extender 6.7 or 7.5 or earlier, and you are now using this version, you must update your type trees to use the current XML validation. To update your type trees, you can use the XML Type Tree Compatibility Utility. This utility is also available from the command line (`dsxmlconv`).

When you use the XML Type Tree Compatibility Utility, there are two important requirements. One is that you must specify the correct version of the IBM Transformation Extender XML Schema or DTD importer that was used to create the original type tree. The other is that you must have the original DTD or Schema.

To make a type tree compatible with this version of IBM Transformation Extender XML validation:

Note: The original DTD/schema file is required when you use the XML Type Tree Compatibility Utility.

Procedure

1. Right-click the root type and select Convert from the menu.
The XML Type Tree Compatibility Utility opens.
 2. From the drop-down list, select the importer that was originally used to create the type tree: XML Schema, which is the default setting, or DTD.
 3. Click the browse button and choose a type tree.
 4. From the drop-down list, choose the version of the IBM Transformation Extender importer that is used to create the type tree.
Note: You must specify the correct version of the IBM Transformation Extender importer that was used to create the original type tree. If you use an incorrect version number, the conversion process can complete successfully, however, the type tree would be invalid.
 5. In the Select the XML grammar location field, accept the default value for the original XML DTD or Schema used to create the type tree if it is accurate. Otherwise, enter the correct location of the XML grammar.
 6. Enable or disable the Create backup file option. By default, the option is enabled and the location for the backup file is automatically displayed in the subsequent field, which is based on the type tree that you selected. You can change the location and file name extension if necessary.
 7. If applicable, select the national language of the original type tree.
 8. Click Next to begin the update process.
- The following list includes the possible results:
- If no modifications were made to the original type tree created from the DTD or schema, the process completes successfully.
 - If modifications were made to the original type tree, they are displayed in the right pane of the utility in red text. You can incorporate any differences into the new (target) type tree (see "Modifying the Target Type Tree").
- For example, the non-XML types that were added to the original type tree can be added to the new (target) type tree.
9. When you see that the type tree was synchronized successfully, click Finish to close the utility.

Modifying the target type tree

About this task

When the XML utility encounters any non-XML types that have been added to the type tree after it was first created from a schema or DTD, the unrecognized types are displayed in red text.

The instructions below describe how to add the types back into your target tree, however, you have the ability from within this utility to make any necessary modifications to the target type tree in the same manner as you would from the type tree editor (using add, delete, copy, and paste operations).

The following procedures can be used for both XML utilities: XML Type Tree Compatibility Utility and XML Type Tree/Schema Synchronization Utility.

To add a type to the target tree

Procedure

1. In the left pane of the XML utility, right-click on the type and select `Edit > Copy`.
2. In the right pane (which displays the target type tree), right-click on the corresponding (parent) type and select `Edit > Paste`.
The type appears in the target tree.
3. After making all changes, click Next to continue.

To add a component to the target tree

Procedure

1. In the right pane (which displays the target type tree), select the type to add the component to.
 2. At the bottom of the pane, select the Components tab.
 3. Press **Alt** and then select-and-drag the type to the **Component** column of the Components tab.
 4. After adding all components, click Next to continue.
-

Any-2-XML

The Any-2-XML utility is an extension of the type tree Export as Schema functionality that you can use to create a map that can transform any input data into XML output, based on a type tree structure that you select.

How it works

Choose a type within a type tree that models the XML output you want to produce. The utility converts the type into an XML Schema from which it creates a new XML type tree. A new map source file is created containing the new map. The input card of the new map references the original type tree and input and the output card references the new XML type tree and the new output file name. The result of running the new map is XML output.

- [Using Any-2-XML](#)
-

Using Any-2-XML

Procedure

1. Open a type tree that models the XML output structure you want to produce.
 2. In the type tree editor, select a type that represents the portion of the input file that you want to export.
This must be a group or item.
 3. Right-click the type and click Export as Schema.
The Export as Schema dialog opens.
 4. Optional: You can specify the schema name and path if you do not want to use the default values. Click the button at the end of the Schema field, and in the Select window, specify the schema name and path.
 5. Select the Create a map that transforms the input data described by the type tree into XML output check box.
 6. Click the button at the end of the Input file name field, and in the Select window, select the input file that corresponds to the type tree.
 7. In the Output file name field, enter a filename and path for the XML output file that is generated as a result of running the new map. You can also click the button at the end of the field, and make your selections in the Select window.
 8. In the Map file name field, enter a name for the new map source file.
 9. Click OK to start the utility.
The following files are created:
 - XML Schema (*orig_type_tree_name_plus_exported_type_name.xsd*, which is the default, or *name_you Optionally_specified.xsd*)
 - Type tree (*orig_type_tree_name_plus_exported_type_name.mtt*)
 - Map source file (.mms)A log file (*map_name.log*) is created only when an error occurs.
 10. Open the new map.
 11. Build and run the map.
An XML output file is created.
-

Return codes and error messages

- [Type Tree Analyzer errors and warnings](#)
 - [Type tree analysis logic error messages](#)
 - [Logic error and warning messages](#)
 - [Type tree analysis structure error messages](#)
 - [Type tree analysis structure warning messages](#)
-

Type Tree Analyzer errors and warnings

The Type Tree Analyzer checks the logic and internal consistency of your data definitions. Type tree analyzer error and warnings messages are issued on the type of analysis that is performed: logic and structural.

- Logical analysis addresses the integrity of the relationships that you define in the type tree.
- Structural analysis addresses the integrity of the underlying database.

Warnings indicate a successful analysis and are relatively insignificant. Warning messages provide information about inconsistencies that occurred, when the type tree was changed, and were automatically resolved.

Errors are important. Error messages provide information about errors in your type definitions that you should correct. An error may result in unpredictable results in your mapping.

Words in italics represent information that varies, indicating information specific to the type for which the message is generated.

Type tree analysis logic error messages

The following table lists the logic error messages that result from a logical analysis of a type tree:

Return Code	Message
L100	COMPONENT neither inherited nor local: ' <i>type name</i> ' of TYPE: ' <i>type name</i> ' Hint: Look at the super-type's component list. The component is a valid type, but the supertype has a component list that restricts you from using this type as a component. You may have added subtype components before adding supertype components. Either remove all supertype components or add the components in error to the component list of the supertype.
L101	This GROUP must have at least one component - TYPE: ' <i>type name</i> ' Hint: If you want to map this group, add components. If you do not want to map it, make it a category.
L102	Circular reference found in COMPONENT # (' <i>type name</i> ') - TYPE: ' <i>type name</i> ' Hint: Look at the type of the component in error. It is probably missing an initiator or terminator.
L103	Circular reference found in Floating Component type - TYPE: ' <i>type name</i> ' Hint: Look at the floating component type in error. It is probably missing an initiator or terminator.
L104	DELIMITER for TYPE - ' <i>type name</i> ' must have a value Hint: All delimited groups need a delimiter. Edit delimited group properties to insert the missing delimiter.
L105	DELIMITER type neither inherited nor local - TYPE: ' <i>type name</i> ' Hint: The delimiter name has been entered incorrectly. It should be the name of a local type, or the name of an inherited delimiter, or the name of a type in the sub-tree of the inherited delimiter.
L106	Default DELIMITER not specified - TYPE: ' <i>type name</i> ' Hint: This Type was specified with a FIND option for its delimiter. Please add a default value to define what to use for building outputs.
L107	Default DELIMITER not in restriction list - TYPE: ' <i>type name</i> ' Hint: This delimiter was specified as a syntax item. Add the default value to the restriction list for that syntax item.
L108	DELIMITER type is not a SYNTAX ITEM - TYPE: ' <i>type name</i> ' Hint: Delimiters specified as an item must be specified to be interpreted as SYNTAX to set the value of the delimiter if it appears as a component in a data stream.
L109	DELIMITER type has no restriction list - TYPE: ' <i>type name</i> ' Hint: All syntax items need a restriction list.
L110	RELEASE CHARACTER neither inherited nor local - TYPE: ' <i>type name</i> ' Hint: The release character name has been entered incorrectly. It should be either the name of a local type, the name of an inherited release character, or the name of a type in the sub-tree of the inherited release character.
L111	Default RELEASE CHARACTER not specified - TYPE: ' <i>type name</i> ' Hint: This Type was specified with a syntax item for its release character. Please add a default value to define a value for the release character that has not been encountered in the data.
L112	Default RELEASE CHARACTER not in restriction list - TYPE: ' <i>type name</i> ' Hint: This Type was specified with a syntax item for its release character. Please add the default value to the restriction list of that syntax item.
L113	RELEASE CHARACTER type is not a SYNTAX ITEM - TYPE: ' <i>type name</i> ' Hint: Release characters specified as an item must be specified to be interpreted as SYNTAX to set the value of the release character if it appears as a component in a data stream.
L114	RELEASE CHARACTER type has no restriction list - TYPE: ' <i>type name</i> ' Hint: All syntax items need a restriction list.
L115	Floating Component TYPE neither inherited nor local - TYPE: ' <i>type name</i> ' Hint: The floating component name has been entered incorrectly. It should be either the name of a local type, the name of an inherited floating component, or the name of a type in the sub-tree of the inherited floating component.

- L116 INITIATOR type neither inherited nor local - TYPE: '*type name*'
 Hint: The initiator name has been entered incorrectly. It should be either the name of a local type, the name of an inherited initiator, or the name of a type in the sub-tree of the inherited initiator.
- L117 Default INITIATOR not specified - TYPE: '*type name*'
 Hint: This Type was specified with a syntax item for its initiator. Add a default value to define a value for that initiator has not been encountered in the data.
- L118 Default INITIATOR not in restriction list - TYPE: '*type name*'
 Hint: This Type was specified with a syntax item for its initiator. Add the default value to the restriction list of that syntax item.
- L119 INITIATOR type is not a SYNTAX ITEM - TYPE: '*type name*'
 Hint: Initiators specified as an item must be specified to be interpreted as SYNTAX to set the value of the initiator if it appears as a component in a data stream.
- L120 INITIATOR type has no restriction list - TYPE: '*type name*'
 Hint: All syntax items need a restriction list.
- L121 TERMINATOR type neither inherited nor local - TYPE: '*type name*'
 Hint: The terminator name has been entered incorrectly. It should be either the name of a local type, the name of an inherited terminator, or the name of a type in the sub-tree of the inherited terminator.
- L122 Default TERMINATOR not specified - TYPE: '*type name*'
 Hint: This Type was specified with a syntax item for its terminator. Add a default value to define a value for that terminator has not been encountered in the data.
- L123 Default TERMINATOR not in restriction list - TYPE: '*type name*'
 Hint: This Type was specified with a syntax item for its terminator. Please add the default value to the restriction list of that syntax item.
- L124 TERMINATOR type is not a SYNTAX ITEM - TYPE: '*type name*'
 Hint: Terminators specified as an item must be specified to be interpreted as SYNTAX to set the value of the terminator if it appears as a component in a data stream.
- L125 TERMINATOR type has no restriction list - TYPE: '*type name*'
 Hint: All syntax items need a restriction list.
- L126 COMPONENT range minimum (#) greater than range maximum (#) - COMPONENT '*type name*' - TYPE: '*type name*'
 Hint: The minimum range must be less than or equal to the maximum range.
- L127 COMPONENT range minimum (#) less than inherited range minimum(#)- COMPONENT '*type name*' - TYPE: '*type name*'
 Hint: The component in error has been inherited. Look at the range of the component with the same name in the super-type's component list.
- L128 COMPONENT range maximum (#) greater than inherited range maximum(#)- COMPONENT '*type name*' - TYPE: '*type name*'
 Hint: The component in error has been inherited. Look at the range of the component with the same name in the super-type's component list.
- L129 COMPONENT RULE references a COMPONENT later in the component list - '*type name*' - TYPE: '*type name*'
 Hint: Move the component rule to the component later in the list.
- L130 COMPONENT RULE references undefined type - COMPONENT # of TYPE: '*type name*'
 Hint: Verify the spelling of the data object name. The rule should reference a data object name of the component or a data object name of a component earlier in the component list.
- L131 COMPONENT RULE references components of a partitioned group - COMPONENT # of TYPE *type name*
- L132 Invalid partitioning: TYPE has no SUBTYPES - TYPE: '*type name*'
 Hint: Remove the partitioned option from the class window or add sub-types to the Type in error.
- L133 Type of COMPONENT exists, but its relative name is not valid: '*type name*' in TYPE: '*type name*'
 Hint: To get the correct relative name, drag the type you want to use as a component and drop it in the component list of the Type. (Remember to delete the invalid component!)
- L134 Reference to 'ANY' not allowed: COMPONENT number # of TYPE: '*type name*'
 Hint: In this case, the Type in error is a group and it is not the root of a partitioned tree. ANY cannot be used if that component needs to be validated. So, if that group is partitioned, you cannot use ANY for a component up to and including the identifier (if there is one). If that group is not partitioned, you cannot use ANY at all.
- L135 COMPONENT number # cannot reference a CATEGORY in TYPE: '*type name*' (because group is not partitioned)

Hint: In this case, the Type in error is a group and it's not the root of a partitioned tree. A category cannot be used if the component must be validated. So, if that group is partitioned, you cannot use a category for a component up to and including the identifier (if there is one). If that group is not partitioned, you cannot use a category as a component at all.

L136

COMPONENT '*type name*' occurs more than once in list - TYPE: '*type name*'

Hint: Each component in the same component list must have a unique type name. Try to make sub-types of the type name in error and replace each non-unique component with one of the new sub-types.

L137

COMPONENT '*type name*' and its super-type cannot be in same COMPONENT LIST (in TYPE: '*type name*')

Hint: Try making another sub-type of the super-type and replace the super-type reference with the new sub-type.

L138

COMPONENT '*type name*' is same type as delimiter - TYPE: '*type name*'

Hint: A component and a delimiter cannot have the same name. You may need to add sub-types to the type name used in error to resolve this one.

L139

COMPONENT '*type name*' is sub-type of delimiter - TYPE: '*type name*'

Hint: This occurs when a syntax item is used to specify a delimiter. You can add another sub-type to the syntax item and replace the delimiter name with the new sub-type name.

L140

COMPONENT '*type name*' is super-type of delimiter - TYPE: '*type name*'

Hint: This occurs when a syntax item is used to specify a delimiter. You can add another sub-type to the syntax item and replace the component name with the new sub-type name.

L141

COMPONENT '*type name*' is same type as initiator - TYPE: '*type name*'

Hint: A component and an initiator cannot have the same name. You can add sub-types to the type name used in error and replace both the component name and the initiator name.

L142

COMPONENT '*type name*' is sub-type of initiator - TYPE: '*type name*'

Hint: This occurs when a syntax item is used to specify an initiator. You can add another sub-type to the syntax item and replace the initiator name with the new sub-type name.

L143

COMPONENT '*type name*' is super-type of initiator - TYPE: '*type name*'

Hint: This occurs when a syntax item is used to specify an initiator. You can add another sub-type to the syntax item and replace the component name with the new sub-type name.

L144

COMPONENT '*type name*' is same type as terminator - TYPE: '*type name*'

Hint: A component and a terminator cannot have the same name. Try adding sub-types to the type name used in error and replace both the component name and the terminator name with one of the new sub-types.

L145

COMPONENT '*type name*' is sub-type of terminator - TYPE: '*type name*'

Hint: This occurs when a syntax item is used to specify a terminator. You can add another sub-type to the syntax item and replace the terminator name with the new sub-type name.

L146

COMPONENT '*type name*' is super-type of terminator - TYPE: '*type name*'

Hint: This occurs when a syntax item is used to specify a terminator. You can add another sub-type to the syntax item and replace the component name with the new sub-type name.

L147

COMPONENT '*type name*' is same type as Floating Component - TYPE: '*type name*'

Hint: Make both the floating component name and the component name sub-types of the floating component.

L148

COMPONENT '*type name*' is sub-type of Floating Component - TYPE: '*type name*'

Hint: Make both the floating component name and the component name sub-types of the floating component.

L149

COMPONENT '*type name*' is super-type of Floating Component - TYPE: '*type name*'

Hint: Make both the floating component name and the component name sub-types of the floating component.

L150

COMPONENT '*type name*' is same type as release character - TYPE: '*type name*'

Hint: This occurs when a syntax item is used to specify a release character. You can add sub-types to the syntax item and replace both the component name and the release character name with the new sub-type names.

L151

COMPONENT '*type name*' is sub-type of release character - TYPE: '*type name*'

Hint: This occurs when a syntax item is used to specify a release character. You can add another sub-type to the syntax item and replace the release character name with the new sub-type name.

L152

COMPONENT '*type name*' is super-type of release character - TYPE: '*type name*'

Hint: This occurs when a syntax item is used to specify a release character. You can add sub-types to the syntax item and replace the component name with the new sub-type name.

- L153
DELIMITER '*type name*' is same type as initiator - TYPE: '*type name*'
 Hint: A delimiter and an initiator cannot have the same name. You may need to add sub-types to the type name used in error and replace both the delimiter and initiator names to refer to the new sub-types.
- L154
DELIMITER '*type name*' is sub-type of initiator - TYPE: '*type name*'
 Hint: This occurs when a syntax item is used to specify both an initiator and a delimiter. You can add another sub-type to the syntax item and replace the initiator name with the new sub-type name.
- L155
DELIMITER '*type name*' is super-type of initiator - TYPE: '*type name*'
 Hint: This occurs when a syntax item is used to specify both an initiator and a delimiter. You can add another sub-type to the syntax item and replace the delimiter name with the new sub-type name.
- L156
DELIMITER '*type name*' is same type as terminator - TYPE: '*type name*'
 Hint: A delimiter and a terminator cannot have the same name. You may need to add sub-types to the type name used in error and replace both the delimiter and terminator names to refer to the new sub-types.
- L157
DELIMITER '*type name*' is sub-type of terminator - TYPE: '*type name*'
 Hint: This occurs when a syntax item is used to specify both a delimiter and a terminator. You can add another sub-type to the syntax item and replace the terminator name with the new sub-type name.
- L158
DELIMITER '*type name*' is super-type of terminator - TYPE: '*type name*'
 Hint: This occurs when a syntax item is used to specify both a delimiter and a terminator. You can add another sub-type to the syntax item and replace the delimiter name with the new sub-type name.
- L159
DELIMITER '*type name*' is same type as release character - TYPE: '*type name*'
 Hint: A delimiter and a release character cannot have the same name. You may need to add sub-types to the type name used in error and replace both the delimiter and release character names to refer to the new sub-types.
- L160
DELIMITER '*type name*' is sub-type of release character - TYPE: '*type name*'
 Hint: This occurs when a syntax item is used to specify both a delimiter and a release character. You can add another sub-type to the syntax item and replace the release character name with the new sub-type name.
- L161
DELIMITER '*type name*' is super-type of release character - TYPE: '*type name*'
 Hint: This occurs when a syntax item is used to specify both a delimiter and a release character. You can add another sub-type to the syntax item and replace the delimiter name with the new sub-type name.
- L162
DELIMITER '*type name*' is same type as Floating Component - TYPE: '*type name*'
 Hint: A delimiter and a floating component cannot have the same name. Try adding sub-types to the type name used in error and replace both the delimiter and floating component names to refer to the new sub-types.
- L163
DELIMITER '*type name*' is sub-type of Floating Component - TYPE: '*type name*'
 Hint: This occurs when a syntax item is used to specify both a delimiter and a floating component. You can add another sub-type to the syntax item and replace the floating component name with the new sub-type name.
- L164
DELIMITER '*type name*' is super-type of Floating Component - TYPE: '*type name*'
 Hint: This occurs when a syntax item is used to specify both a delimiter and a floating component. You can add another sub-type to the syntax item and replace the delimiter name with the new sub-type name.
- L165
INITIATOR '*type name*' is same type as terminator - TYPE: '*type name*'
 Hint: An initiator and a terminator cannot have the same name. You may need to add sub-types to the type name used in error and replace both the initiator and terminator names to refer to the new sub-types.
- L166
INITIATOR '*type name*' is sub-type of terminator - TYPE: '*type name*'
 Hint: This occurs when a syntax item is used to specify both an initiator and a terminator. You can add another sub-type to the syntax item and replace the terminator name with the new sub-type name.
- L167
INITIATOR '*type name*' is super-type of terminator - TYPE: '*type name*'
 Hint: This occurs when a syntax item is used to specify both an initiator and a terminator. You can add another sub-type to the syntax item and replace the initiator name with the new sub-type name.
- L168
INITIATOR '*type name*' is same type as release character - TYPE: '*type name*'
 Hint: This occurs when a syntax item is used to specify both an initiator and a release character. You can add sub-types to the syntax item and replace both the initiator name and the release character name with a new sub-type name.
- L169
INITIATOR '*type name*' is sub-type of release character - TYPE: '*type name*'
 Hint: This occurs when a syntax item is used to specify both an initiator and a release character. You can add another sub-type to the syntax item and replace the release character name with the new sub-type name.

- L170 INITIATOR '*type name*' is super-type of release character - TYPE: '*type name*'
 Hint: This occurs when a syntax item is used to specify both an initiator and a release character. You can add another sub-type to the syntax item and replace the initiator name with the new sub-type name.
- L171 INITIATOR '*type name*' is same type as Floating Component - TYPE: '*type name*'
 Hint: An initiator and a floating component cannot have the same name. Try adding sub-types to the type name used in error and replace both the initiator and floating component names with the new sub-types.
- L172 INITIATOR '*type name*' is sub-type of Floating Component - TYPE: '*type name*'
 Hint: This occurs when a syntax item is used to specify both an initiator and a floating component. You can add another sub-type to the syntax item and replace the floating component name with the new sub-type name.
- L173 INITIATOR '*type name*' is super-type of Floating Component - TYPE: '*type name*'
 Hint: This occurs when a syntax item is used to specify both an initiator and a floating component. You can add another sub-type to the syntax item and replace the initiator name with the new sub-type name.
- L174 TERMINATOR '*type name*' is same type as release character - TYPE: '*type name*'
 Hint: A terminator and a release character cannot have the same name. You may need to add sub-types to the type name used in error and replace both the terminator and release character names with the new sub-types.
- L175 TERMINATOR '*type name*' is sub-type of release character - TYPE: '*type name*'
 Hint: This occurs when a syntax item is used to specify both a terminator and a release character. You can add another sub-type to the syntax item and replace the release character name with the new sub-type name.
- L176 TERMINATOR '*type name*' is super-type of release character - TYPE: '*type name*'
 Hint: This occurs when a syntax item is used to specify both a terminator and a release character. You can add another sub-type to the syntax item and replace the terminator name with the new sub-type name.
- L177 TERMINATOR '*type name*' is same type as Floating Component - TYPE: '*type name*'
 Hint: A terminator and a floating component cannot have the same name. You may need to add sub-types to the type name used in error and replace both the terminator and floating component names with the new sub-types.
- L178 TERMINATOR '*type name*' is sub-type of Floating Component - TYPE: '*type name*'
 Hint: This occurs when a syntax item is used to specify both a terminator and a floating component. You can add another sub-type to the syntax item and replace the floating component name with the new sub-type name.
- L179 TERMINATOR '*type name*' is super-type of Floating Component - TYPE: '*type name*'
 Hint: This occurs when a syntax item is used to specify both a terminator and a floating component. You can add another sub-type to the syntax item and replace the terminator name with the new sub-type name.
- L180 RELEASE CHARACTER '*type name*' is same type as Floating Component - TYPE: '*type name*'
 Hint: A release character and a floating component cannot have the same name. You may need to add sub-types to the type name used in error and replace both the release character and floating component names to refer to the new sub-types.
- L181 RELEASE CHARACTER '*type name*' is sub-type of Floating Component - TYPE: '*type name*'
 Hint: This occurs when a syntax item is used to specify both a release character and a floating component. You can add another sub-type to the syntax item and replace the floating component name with the new sub-type name.
- L182 RELEASE CHARACTER '*type name*' is super-type of Floating Component - TYPE: '*type name*'
 Hint: This occurs when a syntax item is used to specify both a release character and a floating component. You can add another sub-type to the syntax item and replace the release character name with the new sub-type name.
- L183 COMPONENT NAME ambiguous: '*type name*' in TYPE: '*type name*'
 Hint: This type has a component whose relative name can be associated with more than one type in the type tree. Rename the conflicting types.
- L184 RESTRICTION longer than max TYPE size - RESTRICTION # of TYPE: '*type name*'
 Hint: The Type in error is an item. Either change the maximum size of the item or remove the restriction.
- L185 RESTRICTION used in an earlier partition - RESTRICTION # of TYPE: '*type name*'
 Hint: Item Partitions must have mutually exclusive restrictions. Remove the restriction from one of the partition restriction lists.
- L186 Type of COMPONENT does not exist - '*type name*' in TYPE: '*type name*'
 Hint: You probably entered an incorrect type name. Try the drag and drop approach to get the correct one.
- L187 TYPE must be partitioned (since in a partitioned tree and has sub-types) - TYPE: '*type name*'
 Hint: All types in a partitioned sub-type must have mutually exclusive data objects. Set the partitioned property for the type in error.

- L188
TYPE is FIXED, COMPONENT # must have a maximum range value - TYPE *type name*
- L189
TYPE is FIXED, but COMPONENT # is not fixed - TYPE: `*type name*'
Hint: If the component is not intended to be fixed in size, change the group format for the Type to implicit. If the group format is intended to be fixed, check the component: if that component is an item, make sure it has a Padded To length; if that component is a group, change its type to be of fixed syntax.
- L190
BINARY text ITEM used as COMPONENT neither FIXED nor SIZED - COMPONENT # of TYPE: `*type name*'
Hint: The size of a binary text item must either have a Padded To length or it must be sized by the previous component.
- L191
COMPONENT with SIZED attribute is not an UNSIGNED INTEGER ITEM TYPE - COMPONENT # of TYPE: `*type name*'
Hint: A component used to size the component that follows it must be defined as an unsigned integer item type.
- L192
The last COMPONENT in the COMPONENT LIST may not have a SIZED attribute: TYPE: `*type name*'
Hint: Specify a component to follow the one with the sized attribute.
- L193
Range of COMPONENT # must have a maximum value to indicate how many placeholders are needed for its series in TYPE: `*type name*'.
Hint: Change the range maximum to a specific value (not "s") if you may re-define the data this way.
- L194
Cannot distinguish delimiter from terminator in TYPE: `*type name*'.
Hint: Make the range of the last component in the type fixed or make the delimiter of the type different from its terminator.
- L195
Cannot distinguish delimiter contained in COMPONENT # from terminator of TYPE: `*type name*'.
Hint: Make that component bound or make that contained delimiter different from the type terminator.
- L196
Cannot distinguish delimiter of COMPONENT # from delimiter of TYPE: *type name* because COMPONENT # has no placeholder.
Hint: Make that component bound or make that component's delimiter different from the type delimiter.
- L197
Cannot distinguish delimiter of COMPONENT # from delimiter of TYPE: *type name* because COMPONENT # has no range maximum.
Hint: Make that component bound, or make that component's delimiter different from the type delimiter, or specify a range maximum that has a specific value (not "s") for the last component of COMPONENT #.
- L198
Cannot distinguish delimiter contained in COMPONENT # from delimiter of TYPE: `*type name*'.
Hint: Make that contained component bound or make that contained component's delimiter different from the type delimiter.
- L200
Cannot distinguish delimiter contained in COMPONENT # from delimiter of TYPE: `*type name*'.
Hint: Either make that contained component bound, make that contained component's delimiter different from the type delimiter, or specify a range maximum that has a specific value (not "s") for the last component of the contained component.

Logic error and warning messages

The tables in this section list the logic warning messages that result from a logic analysis of a type tree.

The following table lists the warnings than can result when a map is compiled.

Warnings should be resolved because they may produce unpredictable results at mapping time.

Return Code

Message

- L199
COMPONENT # is not distinguishable from COMPONENT # that may follow in TYPE: `*type name*'.
Hint: Make the first COMPONENT bound, or look at the tables in "[Distinguishable objects](#)" to see how you can define the two component types as distinguishable.
- L201
Different data objects of COMPONENT # are not distinguishable in TYPE: `*type name*'.
Hint: See "[Distinguishable objects](#)" for more information about distinguishable objects.
- L202
RESTRICTION list deleted: TYPE is not an ITEM - TYPE: `*type name*'
Hint: Type class was changed from an item to a group or category, so the restriction list was deleted. If this was not your intent, change it back to the way it was.
- L203
COMPONENT list deleted: TYPE is an ITEM - TYPE: `*type name*'
Hint: Type class was changed from a group to a item or category, so the program deleted its component list. If this was not your intent, change it back to the way it was.
- L204
DELIMITER deleted: TYPE is not a DELIMITED GROUP - TYPE: `*type name*'
Hint: Group format was changed from delimited to something else, so the program deleted its delimiter. If this was not your intent, change it back to the way it was.
- L205

COMPONENT RULE deleted: TYPE is a CATEGORY - TYPE: 'type name' (warning)
Hint: Type class was changed from a group to a category, so its component rule was deleted. If this was not your intent, change it back to the way it was.

L206

DELIMITER cannot be found (because first component is not required) - TYPE: 'type name' (warning)
Hint: If the delimiter is missing, a previously set initiator value or the default value is used.

L251

COMPONENT NAME could apply to more than one type:'type name' in TYPE: 'type name' (warning).

Type tree analysis structure error messages

The following table lists the structure error messages that result from a structural analysis of a type tree:

Return Code	Message
S100	Invalid TYPE Name: SubTYPE # of TYPE: 'type name'
S101	Invalid TYPE chain: SubTYPE # of TYPE: 'type name'
S118	Invalid TYPE NAME WhereUsed chain - TYPE NAME: 'type name' (error).
S133	Referenced COMPONENT not 'InUse' - COMPONENT # of TYPE: 'type name' (error).
S134	COMPONENT previously referenced - COMPONENT # (COMP #) of TYPE: 'type name' (error).
S149	Bad Parent COMPONENT Index - COMPONENT 'type name' - TYPE: 'type name' (error)

Type tree analysis structure warning messages

The following table lists the structure warning messages that result from a structural analysis of a type tree:

Return Code	Message
S102	Unused DELIMITER deleted: 'type name' (at index #)
S103	Invalid DELIMITER pointer deleted - TYPE: 'type name'
S104	Invalid default DELIMITER pointer deleted - TYPE: 'type name'
S105	Invalid RELEASE Char pointer deleted - TYPE: 'type name'
S106	Invalid default RELEASE Char pointer deleted - TYPE: 'type name'
S107	Invalid INITIATOR pointer deleted - TYPE: 'type name'
S108	Invalid default INITIATOR pointer deleted - TYPE: 'type name'
S109	Invalid TERMINATOR pointer deleted - TYPE: 'type name'
S110	Invalid default TERMINATOR pointer deleted - TYPE: 'type name'
S111	Resetting DELIMITER Use Count (was # now #) - DELIMITER: 'type name'
S112	Unused DESCRIPTION deleted: 'type name' (at index #)
S113	Invalid DESCRIPTION pointer deleted - TYPE: 'type name'
S114	Resetting DESCRIPTION Use Count (was # now #) - DESCRIPTION: 'type name'
S115	Invalid Floating Component TYPE pointer deleted - TYPE: 'type name'
S116	Invalid TYPE UsedInComp chain repaired - TYPE: 'type name'
S117	Unused TYPE NAME deleted - TYPE NAME: 'type name' (at index #)
S119	Resetting TYPE NAME use count (was # now #) - TYPE NAME: 'type name'
S120	Repaired empty TYPE NAME WhereUsed chain - TYPE NAME: 'type name'
S121	Unused RESTRICTION NAME deleted: 'type name' (at index #)
S122	Invalid RESTRICTION NAME deleted no DESCRIPTION was available - TYPE: 'type name' .

```

S123    Invalid RESTRICTION NAME deleted DESCRIPTION was `type name' - TYPE: `type name'
S124    Resetting RESTRICTION NAME Use Count (was # now #) - RESTRICTIONS: `type name'
S125    Unused RESTRICTION DESCRIPTION deleted: `type name' (at index #)
S126    Invalid RESTRICTION DESCRIPTION deleted - TYPE: `type name'
S127    Resetting RESTRICTION DESCRIPTION Use Count (was # now #) - RESTRICTIONS: `type name'
S128    Unused RULE deleted: `type name' (at index #)
S129    Invalid RULE pointer deleted - COMPONENT # of TYPE: `type name'
S130    Resetting RULE Use Count (was # now #) - RULE: `type name'
S131    Invalid COMPONENT TYPE Description pointer - COMPONENT #
S132    COMPONENT marked `InUse' found in Free Chain- COMPONENT #
S135    COMPONENT in Free Chain referenced by a TYPE - COMPONENT #
S136    COMPONENT recovered and added to Free Chain - COMPONENT #
S137    TYPE in Free Chain referenced by another TYPE - TYPE #
S138    TYPE recovered and added to Free Chain - TYPE X'%04X'
S139    TYPE marked `InUse' but not referenced - TYPE #
S140    Referenced TYPE not marked `InUse' - TYPE #
S141    TYPE Free Chain not in order: sorting
S142    COMPONENT Free Chain not in order: sorting
S143    Overlap found in LIST Free Chain
S144    Free Chain extends into unallocated region
S145    Overlap found in COMPONENT LIST SPACE: list cleared COMPONENTS will be deleted
S146    Invalid COMPONENT LIST pointer: all COMPONENTS DELETED - TYPE: `type name'
S147    Resetting COUNT in COMPONENT LIST: some COMPONENTS may be lost
S148    RULE truncated (due to internal error): `type name' (at index #)
S150    CATEGORY `type name' was missing GROUP and/or ITEM attributes
S151    GROUP `type name' was missing GROUP attributes
S152    ITEM `type name' was missing ITEM attributes

```

Type Tree Importer

Introduction to the Importer Wizard

Use the Importer Wizard to convert metadata into type tree script (.mts) files. The Importer Wizard can import various data formats to create type trees (.mtt).

The Importer Wizard uses a series of maps to convert metadata into a type tree script file (.mts). The Type Tree Maker then processes this type tree script and generates a type tree that contains all the supported types that are defined in the imported metadata.

Many of the type trees that are generated by the Importer Wizard can be immediately used for map development. However, depending on the contents of the interface-specific metadata file, it might be necessary for the generated type tree to be modified using the Type Designer.

You cannot simultaneously have a type tree open in the Type Designer while importing a type tree with the same name.

- [List of importers](#)
- [Application-specific importers](#)
- [Configuring JVM options from the dtx.ini file](#)
- [Accessing the Importer Wizard](#)
- [CORBA IDL Importer](#)
- [Java Class Importer](#)
- [Java Message Service \(JMS\) Importer](#)

- [Text File Importer](#)
 - [Type Library Importer](#)
 - [Type Tree Maker Importer](#)
-

List of importers

The following is a list of importers available in the Importer Wizard:

- COBOL Copybook (deprecated)
- COBOL Copybook
- CORBA IDL
- Java Class
- Java Message Service (JMS)
- JavaScript Object Notation (JSON)
- PL/I
- Text File
- Type Library
- Type Tree Maker
- WSDL
- XML DTD
- XML Schema

Note: Some importers, such as application-specific importers, are add-on components.

Application-specific importers

Additional application-specific importers are available.

For information about using an application-specific importer, see the documentation installed with the particular application pack for IBM Sterling Transformation Extender.

Configuring JVM options from the dtx.ini file

About this task

Importers convert metadata from external sources into the type trees at design time. Most importers have an associated Wizard in the Type Designer.

Importers with the Java native interfaces use the JVM options specified in the **dtx.ini** file to create Java virtual machine (JVM). You can use the JVM options to tune the importer's behavior, such as setting up big initial or maximum heap size.

The following have Java native interfaces:

- CORBA IDL Importer
- Java Class Importer
- JMS Importer
- PL/I Importer
- WSDL Importer
- XML DTD Importer
- XML Schema Importer

The following JVM options can be modified in the **dtx.ini** file to fit your specifications:

- JVM Options
 - JNI Layer Trace
 - [JVM options](#)
 - [JNI layer trace](#)
 - [Configuring JVM options example](#)
-

JVM options

Lists additional options passed to the JVM at initialization time. The syntax of the individual entry is:

Option

Description

optionN

=<single JVM option with its arguments>

Where **N** is an integer number between 1 and 128. If the same entry appears more than once the first occurrence is taken.

JNI layer trace

The Java Native Interface (JNI) layer trace provides trace and debugging information about how IBM Sterling Transformation Extender interacts with the Java Virtual Machine (JVM). Use the JNI layer trace to collect details about a Java adapter problem that are not recorded in the Java-based adapter trace file. For example, if an exception is not recorded by a IBM Sterling Transformation Extender Java-based adapter or command (such as JMS, Java Class, SOAP, JExit), the JNI trace file records the error. If the Java adapter does not load at all, the JNI layer trace file records details about the failure.

When IBM Sterling Transformation Extender creates the JVM with the -verbose option (specified in the **[JVM Options]** section of the dtx.ini file), the JNI layer trace file includes the details about the JVM garbage collection, class loading, and JNI invocation calls.

The JNI layer trace can produce a large amount of information and impact performance. Avoid using the JNI layer trace in a production system, unless you enable it for a limited time and then disable it.

You can define the following options:

Option	Description														
Switch	Master switch. Valid values are on and off. If off, the rest of the options are ignored.														
File	Specifies the full path of the file that receives the tracing information. IBM Sterling Transformation Extender automatically appends the process ID (<i>PID</i>) of the Launcher system or IBM Sterling Transformation ExtenderIBM Integration Bus execution group to the end of the file name. This behavior is the default. As an example, you want IBM Sterling Transformation Extender to write the trace information to the jni.log file in the /apps/wtx84/logs path location. You specify the File option as file=/apps/wtx84/logs/jni.log . IBM Sterling Transformation Extender automatically appends the <i>PID</i> to the end of the file name and writes the trace information to /apps/wtx84/logs/jni.log_pid .														
pid	Turns off the functionality that automatically appends the process ID (<i>PID</i>) to the end of the file name. Add the following entry: pid=off . See the description for the File option.														
Level	Controls the amount of information recorded by the JNI trace. A higher trace level includes the trace information of all the numerically lower levels. For example, trace level 100 includes the information of all trace levels lower than 100. <table border="1"><thead><tr><th>Level</th><th>Description</th></tr></thead><tbody><tr><td>10</td><td>Critical information that tracks execution details of the IBM Sterling Transformation Extender JNI layer</td></tr><tr><td>20</td><td>Errors that surface from the JNI layer as a result of IBM Sterling Transformation Extender JNI calls</td></tr><tr><td>30</td><td>Warning information about the timing and sequence of IBM Sterling Transformation Extender JNI calls. This trace level is the default.</td></tr><tr><td>40</td><td>General information about the execution of a IBM Sterling Transformation Extender Java adapter</td></tr><tr><td>50</td><td>Verbose details regarding the execution of a IBM Sterling Transformation Extender Java adapter</td></tr><tr><td>100</td><td>"Startup" details before attaching to a JVM thread</td></tr></tbody></table>	Level	Description	10	Critical information that tracks execution details of the IBM Sterling Transformation Extender JNI layer	20	Errors that surface from the JNI layer as a result of IBM Sterling Transformation Extender JNI calls	30	Warning information about the timing and sequence of IBM Sterling Transformation Extender JNI calls. This trace level is the default.	40	General information about the execution of a IBM Sterling Transformation Extender Java adapter	50	Verbose details regarding the execution of a IBM Sterling Transformation Extender Java adapter	100	"Startup" details before attaching to a JVM thread
Level	Description														
10	Critical information that tracks execution details of the IBM Sterling Transformation Extender JNI layer														
20	Errors that surface from the JNI layer as a result of IBM Sterling Transformation Extender JNI calls														
30	Warning information about the timing and sequence of IBM Sterling Transformation Extender JNI calls. This trace level is the default.														
40	General information about the execution of a IBM Sterling Transformation Extender Java adapter														
50	Verbose details regarding the execution of a IBM Sterling Transformation Extender Java adapter														
100	"Startup" details before attaching to a JVM thread														

Configuring JVM options example

The following is an example from the **dtx.ini** file:

```
...
[External Jar File]
jar1=c:\MyApp\libs\myapp.jar
jar2=c:\MyApp\classes
[JVM Options]
option1=-Xms200m
option2=-Xmx300m
[JNI Layer Trace]
switch=on
file=c:\dtxjpi.log
...
```

Accessing the Importer Wizard

About this task

The following procedure describes how to access the Importer Wizard.

To access the Importer Wizard:

Procedure

1. Open the Design Studio application.
2. Navigate to the Importer Wizard through either one of the following ways:
Click **File > Import**.
In the Extender Navigator view or Navigator view, right-click the location in a project in which you want to create the new type tree, and select **Import > Import**. The Import window opens.

3. Expand the Transformation Extender folder, select one of the available importers from the import source list, and click Next. Follow the Importer Wizard instructions.

Related tasks

- [Running the COBOL Copybook \(Deprecated\) Importer](#)
 - [Running the COBOL Copybook Importer](#)
 - [Running the JSON Importer](#)
 - [Running the PL/I Importer](#)
 - [Running the XML DTD Importer](#)
 - [Running the XML Schema Importer](#)
 - [Running the WSDL Importer](#)
-

CORBA IDL Importer

The CORBA Interface Definition Language (IDL) Importer is used to generate a type tree representation of an IDL file.

See the *CORBA Adapter* documentation for information about the CORBA IDL Importer.

Java Class Importer

The Java Class Importer allows the user to create a type tree that can be used to manipulate Java objects from within a map using the Java Class Adapter.

See the *Java Class Adapter* documentation for information about the Java Class Importer.

Java Message Service (JMS) Importer

The JMS Importer creates type trees that represent the public fields and constructors of Java classes that are sent as objects using JMS messages.

See the *Java Message Service Adapter* documentation for information about the JMS Importer.

Text File Importer

The Text File Importer automatically generates type trees that describe the format of the data in text files.

See the *Resource Adapters Introduction* for information about the Text File Importer.

Type Library Importer

The Type Library Importer reads the format of COM Components and Methods to automatically generate type trees for use with the COM Automation Adapter.

See the *COM Automation Adapter* documentation for information about the Type Library Importer.

Type Tree Maker Importer

The Type Tree Maker Importer is a facility for automatically generating type trees from a script containing Type Tree Maker commands (.mts file extension).

See the *Type Tree Maker* documentation for detailed information about the Type Tree Maker.

COBOL Copybook (Deprecated) Importer

The COBOL Copybook Importer imports the definition of one or more COBOL copybooks in a file and generates a type tree that contains the corresponding type definitions. The copybook file might contain more than one COBOL copybook. If your file contains multiple COBOL Copybook definitions, all definitions are contained in a single type tree.

The importer uses a series of maps to convert the copybook format into a type tree script file (.mts). The Type Tree Maker then does the following actions: processes this type tree script and generates a type tree that contains all of the supported types that are defined in the copybook.

- [Running the COBOL Copybook \(Deprecated\) Importer](#)
- [Wrapped and unwrapped copybook format](#)
- [Copybook specifications](#)
- [Troubleshooting](#)

- [Modifying a copybook file](#)
 - [Modifying a generated type tree](#)
 - [REDEFINES clause](#)
 - [COBOL Occurs Depending On](#)
 - [COBOL features not supported](#)
 - [Bytes assigned for COMP data](#)
 - [Bytes assigned for COMP-3 data](#)
-

Running the COBOL Copybook (Deprecated) Importer

About this task

Use the COBOL Copybook (Deprecated) Importer to automatically generate COBOL Copybook type trees.

The metadata for the type trees is in the form of COBOL copybooks.

If you are running the COBOL Copybook (Deprecated) Importer to generate a type tree that is used with the CICS Adapter, you must select the [Customer Information Control System \(CICS\) option](#). The option is in the window where you select byte and character sets.

The following procedure describes how to use the COBOL Copybook (Deprecated) Importer Wizard to generate a type tree from COBOL copybooks.

There is an example of a COBOL Copybook file in the *install_dir\examples\general\copybook* folder.

To run the COBOL Copybook (Deprecated) Importer:

Procedure

1. Access the Importer Wizard in the Design Studio application.
2. In the Import window, expand the Transformation Extender folder, select COBOL Copybook (Deprecated) from the import source list, and click Next. The COBOL Copybook Importer window opens.
3. Enter or select the name of the metadata source file that you want to import, following the Importer Wizard instructions, and then click Next. The metadata source files have a default file extension of .cpy.
4. If you are importing metadata to generate a COBOL Copybook type tree, select a character set from the Character set for execution-time data list, and a byte order that is set from the Byte order for execution-time data list. The Generate tree for Customer Information Control System check box is disabled.
5. To generate a type tree that is used with the CICS Adapter, select the Generate tree for Customer Information Control System check box. The Character set for execution-time data list and the Byte order for execution-time data list are disabled.

The importer operation generates the type tree with the following parameters:

- Character data set is **EBCDIC**
 - Byte order is **Big Endian**
 - Standard header information is added to the type tree
6. Click Next.
 7. Enter or select the parent folder of the generated type tree.
 8. For File name, do one of the following choices:
 - To create a type tree file, enter a new name.
 - To overwrite an existing type tree file, enter its name if it does not already display in the field.The default name for the type tree is the COBOL Copybook file name with the .mtt extension.

9. To generate the type tree, click Next.

If you selected a file name for an existing type tree file, you are prompted to continue or stop the import operation in the Overwrite Warning window.

After the import operation runs, it displays the run results in the COBOL Copybook Importer window. If errors or warnings, or both, occur, they display in that same window. The errors and warnings are also saved to error and log files in the product installation directory. The file names are Cpybkimp.err and cobol.log. You can open these files and analyze them in a text editor after the import operation completes. The import operation deletes and regenerates these log files every time a new COBOL Copybook is imported. If errors occur, the import operation cannot generate the type tree until you resolve the problem. If only warnings occur, the import operation can still generate the type tree. However, to ensure that the reported warnings do not interfere with the requirements necessary for a valid instance document, inspect the generated type tree.

10. After the type tree generation process is complete, click Finish.

If the type tree was generated, you are given the option to open the type tree in the Type Tree editor. The Properties view for the generated type tree contains the imported data format for the generated type tree. You can also click the type tree in the Extender Navigator view or the Navigator view to view the date and time that the import operation generated the type tree, under the Info property in the Properties view.

Wrapped and unwrapped copybook format

The COBOL Copybook Importer automatically processes copybook files in wrapped and unwrapped formats.

There is an example of a COBOL Copybook file in the *install_dir\examples\general\copybook* folder.

- [Wrapped format](#)
 - [Unwrapped format](#)
-

Wrapped format

A wrapped copybook file has COBOL source records that are terminated by either a carriage return/line feed (Hex 0D 0A) or line feed (Hex 0A). Each source record can be fixed or variable length. The important distinguishing characteristic of a wrapped copybook file format is the terminator (newline or carriage return/linefeed) at the end of each source record.

Ensure that the last statement ends with a carriage return/line feed or line feed as appropriate.

Unwrapped format

An unwrapped copybook file has fixed 80-byte source records with no line feed or carriage return/line feed at the end of each source record.

Copybook specifications

The COBOL Copybook Importer uses as input a copybook file (.cpy) containing data description entries, as defined by the ANSI X3.23 specification with the following exception and additions:

- Each entry must have either a data-name or the reserved FILLER clause. (This is an exception to the ANSI X3.23 specification.)
- The COBOL Copybook Importer also supports these USAGE clauses:
 - COMPUTATIONAL-3
 - COMPUTATIONAL-4
 - COMPUTATIONAL-5
 - COMPUTATIONAL-X
 - COMP-3
 - COMP-4
 - COMP-5
 - COMP-X

Trees that are generated by the COBOL Copybook Importer can be used immediately for map development. However, depending on the contents of the copybook file, it might be necessary for the generated type tree to be modified using the Type Designer.

Troubleshooting

This section discusses modifications that you can make before processing a copybook file to accomplish successful type tree generation by the Importer Wizard. It also provides information about the restrictions and limitations under which the Importer Wizard for COBOL Copybook metadata files generates type trees so that, after processing, you can accurately modify the generated type tree.

Modifying a copybook file

If the copybook file does not conform to what the COBOL Copybook Importer Wizard expects, a type tree is not successfully generated. Before importing the copybook file, you might want to:

- Check that the level number statements (for example, 01, 05, 10) in your copybook do not start before Column 8.
- Check for statements not supported by the COBOL Copybook Importer. The importer does not expand any embedded COPY clauses. Add the data included in the embedded COPY clauses before you run the COBOL Copybook Importer.

Note: If you do not modify the copybook file before running the COBOL Copybook Importer wizard, use the Type Designer to manually modify the generated type tree. See the section "[COBOL Features Not Supported](#)" for information.

Modifying a generated type tree

The COBOL Copybook Importer automatically generates type trees from COBOL copybooks that describe the format of data. However, a copybook might contain certain statements or have insufficient information that result in type trees being generated that require some manual modification. Use the Type Designer to modify any type tree that is generated by the COBOL Copybook Importer so that the types contained in the tree accurately represent your data. The situations that might require type tree modifications are:

- The copybook contains REDEFINES clause(s).
- The copybook contains VALUES THROUGH or VALUES THRU clause(s).
- The copybook contains statements that are not supported by the wizard.
- The Byte-Storage Mode is used.

REDEFINES clause

If a copybook contains a REDEFINES clause, the COBOL Copybook Importer adds a group subtype for the REDEFINES object. However, a copybook does not contain sufficient information for the COBOL Copybook Importer to determine how the redefined type is used.

For example, REDEFINES might be used to overlay different **Record** subtypes. In this case, you might want to:

- Copy the **Record** subtype as another subtype.
- Modify the new subtype to use the appropriate **REDEFINES** objects.
- Partition **Record** and specify the component rule that distinguished **Record** subtypes.
- Add a **File** type and specify the pattern of **Record** subtypes as components of **File**.

COBOL Occurs Depending On

COBOL provides support for a variable-length table data structure whose length is determined by the Occurs Depending On (ODO) clause and the data element that it identifies. A variable-length table provides support for a table structure that contains a number of rows that are not defined at design time; for example, an account table, which contains a varying number of accounts.

- [COBOL Occurs Depending On component rules](#)

COBOL Occurs Depending On component rules

When you use the COBOL Copybook Importer to import a COBOL copybook that includes an ODO clause, the ODO restriction is not automatically imported into the type tree. If you need to impose the rules necessary to enforce the ODO in the type tree, you need to make changes to the type tree after it has been imported.

The ODO restrictions are most commonly implemented in a IBM Sterling Transformation Extender type tree as component rules. To implement an ODO restriction as a component rule, put the component rule on the component that is identified with the ODO clause, used to determine the length of the variable-length table. The component rule references the previously defined component, identified by the ODO clause, which limits the number of occurrences of the current component.

- [When component is numeric](#)
- [When component is not numeric](#)

When component is numeric

If the component that is limiting the number of occurrences is numeric, create a component rule on the component that is identified with the ODO clause as described in this topic.

The following example represents two lines from a COBOL copybook:

```
05 ORDREV_OUT_NUM_ACCTS PIC 9(04).
05 ORDREV_OUT_ACCOUNT_INFO OCCURS 0 TO 1000 TIMES
  DEPENDING ON ORDREV_OUT_NUM_ACCTS.
```

Add the following type tree component rule to the construct representing `ORDREV_OUT_ACCOUNT_INFO`:

```
ORDREV_OUT_ACCOUNT_INFO
RULE: COUNT($) <= ORDREV_OUT_NUM_ACCTS Field
```

When IBM Sterling Transformation Extender parses the data at run time, the `ORDREV_OUT_NUM_ACCTS` data field contains a numeric value. For this example, assume that at run time, the value of `ORDREV_OUT_NUM_ACCTS` is 50. That value will be used in the component rule: `COUNT($) <= ORDREV_OUT_NUM_ACCTS Field`, so `COUNT($) <= 50`.

When this rule is run, the first 50 occurrences of `ORDREV_OUT_ACCOUNT_INFO` are processed. IBM Sterling Transformation Extender determines any data that follows the data consumed as the 50 occurrences, to be the successive construct in the copybook.

When component is not numeric

If the component that is limiting the number of occurrences, such as `ORDREV_OUT_NUM_ACCTS` in the previous example, is not numeric, then you might need to specify some additional component rule functions.

The following example specifies `ORDREV_OUT_NUM_ACCTS` as text, not numeric.

```
05 ORDREV_OUT_NUM_ACCTS PIC X(04).
05 ORDREV_OUT_ACCOUNT_INFO OCCURS 0 TO 1000 TIMES
  DEPENDING ON ORDREV_OUT_NUM_ACCTS.
```

You need to specify a rule that converts the text item to a number as in the following example:

```
ORDREV_OUT_ACCOUNT_INFO
RULE: COUNT($) <=
  TEXTT tonumber(ORDREV_OUT_NUM_ACCTS Field)
```

COBOL features not supported

The COBOL Copybook Importer does not handle certain COBOL features that might be in the definition file. If an unsupported statement is encountered, a warning message appears. Types for the following COBOL statements must be added manually to the type tree that is generated by the Importer.

- The COBOL Copybook Importer does not support the THROUGH option in the VALUES clause. The range of values must be added manually to the restriction list for the level 88 item or items that use the THROUGH clause. Alternatively, a component rule could be placed on the item where it is used. For example, the following item defined in the copybook could have a component rule specified as: \$ >= "A" & \$ <= "F":

```
15 CODE-ID PIC X(1).  
     88 OTM-TRD-GSP VALUE `A' THROUGH `F'.  

```

If you are not concerned with validating codes or values, or both, in your input data that conforms to the copybook definition for a particular item, delete the entire restriction list generated by the COBOL Copybook Importer.

- The COBOL Copybook Importer does not expand any embedded COPY clauses. The data included in the embedded COPY clauses must be manually added to the type tree or added to the copybook before you run the wizard.
- The COBOL Copybook Importer does not support the SYNCHRONIZED clause.

Bytes assigned for COMP data

The COBOL Copybook Importer calculates COMP's item space using the Word-Storage Mode. Therefore, if you use the Byte-Storage Mode, you will have to manually change the type tree definitions in the type tree that is generated.

For example, if your data is **PIC 9(5)**, the item in the generated type tree is assigned four bytes.

The following table shows how many bytes are required for each mode:

Digits (9s) Signed	Digits (9s) Unsigned	Bytes Assigned by Importer (Required for Word-Storage Mode)	Bytes Required for Byte-Storage Mode
1-2	1-2	2	1
3-4	3-4	2	2
5-6	5-7	4	3
7-9	8-9	4	4
10-11	10-12	8	5
12-14	13-14	8	6
15-16	15-16	8	7
17-18	17-18	8	8

Bytes assigned for COMP-3 data

If your COBOL data is a PIC clause that is COMP-3, the number of bytes the COBOL Copybook Importer assigns to the item created in the type tree is shown in the following table:

Number of Digits in PIC (With or Without a Sign)	Number of Bytes Assigned to Item
1	1
2-3	2
4-5	3
6-7	4
8-9	5
10-11	6
12-13	7
14-15	8
16-17	9
18	10

For example, if your COBOL data is PIC S999 COMP-3, the Importer Wizard creates an item that is a 2-byte byte stream. If it is PIC S99V9, the item created in the type tree is also a 2-byte Byte Stream. If it is S99V999, the item created in the tree is a 3-byte Byte Stream.

COBOL Copybook Importer

The COBOL Copybook Importer imports the definition of one or more COBOL copybooks in a file and generates a type tree that contains the corresponding type definitions. The copybook file might contain more than one COBOL copybook definition. If your file contains multiple COBOL Copybook definitions, you can import the definitions into a single type tree in the following ways:

- Import a selected subset of the definitions.
- Import all of the definitions.
- Import definitions from different COBOL Copybook files.

- [Running the COBOL Copybook Importer](#)
Use the COBOL Copybook Importer to automatically generate COBOL Copybook type trees.
 - [Selecting communication data structures](#)
 - [COBOL Copybook Importer compiler preferences](#)
 - [Troubleshooting](#)
 - [COBOL and type tree mapping](#)
-

Running the COBOL Copybook Importer

Use the COBOL Copybook Importer to automatically generate COBOL Copybook type trees.

About this task

The metadata for the type trees is in the form of COBOL copybooks.

If you are running the COBOL Copybook Importer to generate a type tree that is used with the CICS Adapter, you must select the Customer Information Control System (CICS) option. The option is in the Output File Location window.

The following procedure describes how to use the COBOL Copybook Importer Wizard to generate a type tree from COBOL copybooks. You specify the data in the COBOL Copybook Sources page, and the options and data structures in the Importer Options and Structure Selection page.

There is an example of a COBOL Copybook file in the *install_dir\examples\general\copybook* folder.

To run the COBOL Copybook Importer:

Procedure

1. Access the Importer Wizard in the Design Studio application.
2. In the Import window, expand the Transformation Extender folder, select COBOL Copybook from the import source list, and click Next. The Type Tree Importer window opens.
3. If you are importing a single COBOL structure to generate a type tree, do the following steps:
 - a. In the COBOL Copybook Sources page, make sure that the Choose mapping field is set to Single COBOL Copybook to type tree. This selection is the default setting. If it is not selected, you can select Single COBOL Copybook to type tree from the Choose mapping list.
 - b. Enter or select the name of the COBOL program or COBOL Copybook metadata source file that you want to import, in the COBOL file field.
The COBOL Copybook Importer supports COBOL files with the following extensions: .cbl, .ccp, .cob, and .cpy. You can configure the type of support for each COBOL file extension, under Window > Preferences > Importer > COBOL > More COBOL options > File Extension Support. You can select to import the COBOL Copybook from a full program or from a data structure only. The metadata source files have a default file extension of .cpy.
 - c. Click Next. The Importer Options and Structure Selection page opens.
 - d. Follow the Importer Wizard instructions to select a communication data structure.
4. If you are importing multiple COBOL structures to generate a single type tree, do the following steps:
 - a. In the COBOL Copybook Sources page, select Multiple COBOL Copybooks to type tree from the Choose mapping list.
 - b. Click Add, and then enter or select the name of the COBOL Copybook metadata source file that you want to import, in the COBOL file field.
 - c. Click Next. The Importer Options and Structure Selection page opens.
 - d. Follow the Importer Wizard instructions to select a communication data structure and then click Finish. The selected data structure displays in the Multiple possible output list.
 - e. Repeat these steps as required. All of the selected data structures display in the Multiple possible output list.
5. Click Next. The Output File Location page opens.
6. Enter or select the Project, Folder, and File name.
7. To generate a type tree that is used with the CICS Adapter, select the Generate type tree for Customer Information Control System check box.
8. Click Finish.

If the type tree was generated, you are given the option to open the type tree in the Type Tree editor. The Properties view for the generated type tree contains the imported data format for the generated type tree. You can also click the type tree in the Extender Navigator view or the Navigator view to view the date and time that the import operation generated the type tree, under the Info property in the Properties view.

Related tasks

- [Accessing the Importer Wizard](#)
-

Selecting communication data structures

Communication data structures are used by the COBOL Copybook Importer to import metadata to generate the type tree structure.

After the importer wizard starts, you can make selections for the following items:

- Runtime options (Character set, Byte order)
- Compiler options (Quote, Trunc, Nsymbol)
- Source structures

The runtime options, Character set and Byte order, describe the runtime data. The character set code represents a standard collection of letters, numbers, and symbols. The byte order set code represents a convention a machine processor uses to position its lowest byte within a word. The processor positions the lowest byte beginning either from the leftmost or the rightmost position. The default setting for these properties is Native, which you can override. To override the default setting for character set, click Select and in the Select Value window, select a valid character set code from the Character set list. To override the default setting for byte order, select a valid byte order set code from the Byte order list.

Under Compiler options, Quote, Trunc, and Nsymbol describe the COBOL compiler options that affect how your COBOL program is processed to create data structures for import. You can define compiler options by configuring the options under the COBOL Preferences page before you run the COBOL Copybook Importer. After the importer wizard starts, you can override some of these compiler options under Compiler options. The default values for the Quote, Trunc, and Nsymbol compiler options are set in the following ways. Quote is set to **DOUBLE**. Trunc is set to **STD**. Nsymbol is set to **DBCS**. To override the default settings, select the values from the Quote, Trunc, and Nsymbol lists.

Under Source structures, are the structures that are found in the selected COBOL Copybook file. Select the top-level source data structures that you want to import into the type tree. If you chose to map a single COBOL copybook to a type tree, select one data structure. If you chose to map multiple COBOL copybooks to a type tree, you can select more than one data structure. After you made your selections, click **Apply**.

If you selected a file name for an existing type tree file, you are prompted to continue or stop the import operation in the Overwrite Warning window. If errors occur, the import operation cannot generate the type tree until you resolve the problem.

If the type tree was generated, you have the option to open the type tree in the Type Tree editor. The Properties view for the generated type tree contains the imported data format for the generated type tree. You can also click the type tree in the Extender Navigator view or the Navigator view to view the date and time that the import operation generated the type tree, under the Info property in the Properties view.

COBOL Copybook Importer compiler preferences

You can set the COBOL compiler preferences for an individual file import through the importer wizard. You can also change the default compiler options in the COBOL Preferences page.

To set the default COBOL compiler preferences, go to **Window > Preferences**. Expand the Importer option and select COBOL. Set the COBOL compiler preferences under the various tabs.

If you are importing COBOL copybooks that contain data types G for graphic, or N for national, be sure to set the locale correctly.

When you are importing data from these different file types, you can also select the correct file extension support option for each of the file types. These settings control how the importer operation processes the copybook and generates the type tree. Therefore, the settings must correctly reflect the contents of the copybook that is being imported.

Troubleshooting

When the COBOL Copybook Importer is unsuccessful in importing a COBOL copybook file, the wizard displays an error message with the details about the import operation.

Before you run the COBOL Copybook Importer, make sure that the COBOL copybook is syntactically correct. The COBOL Copybook Importer does a preprocessing compiling step before it begins the type tree generation step. This preprocessing step checks that the copybook is set up using correct COBOL syntax. If the COBOL copybook is not syntactically correct, the import operation stops running and displays an error message.

Make sure that the file extension support option is correctly selected in the COBOL Preferences page for the file type. The option must correctly reflect the contents of the copybook file. As an example, the default file extension support option in the COBOL Preferences page for the .cob file extension is Full program. A COBOL copybook file with the .cob extension must contain a complete COBOL program for the importer operation to successfully process it and generate the type tree.

If the data type is G for graphic, or N for national, be sure to set the locale correctly in the COBOL Preferences page. Otherwise, the import operation cannot generate the type tree and returns an error message.

Correct the problems in the COBOL copybook file that the importer detects, and reimport the file.

- [COBOL Copybook Importer error message types](#)
- [COBOL Copybook Importer log and trace files](#)

COBOL Copybook Importer error message types

The COBOL Copybook Importer produces various types of error messages when it is unsuccessful in importing a COBOL Copybook file into your project.

If the COBOL Copybook files are stored locally in part of the Eclipse workspace, the syntax error messages are listed in the Problems view. To view the source of the error in your code, double-click the error message line that is in the Problems view. To see the error markers, open the COBOL Copybook file that is in the workspace, with the workbench Text Editor. Errors are logged in a generic log file, which is located under the metadata directory in the workspace that you selected when you opened the Design Studio: [workspace]\.metadata\.

If the COBOL Copybook files are not stored locally in part of the workspace, the syntax error messages are logged in the log file.

COBOL Copybook Importer log and trace files

When the COBOL Copybook Importer runs, it writes information to log and trace files.

The log and trace files are created under the metadata directory in the workspace that you selected when you opened the Design Studio: [workspace]\.metadata\log. Use the logging and tracing information for debugging and troubleshooting purposes.

COBOL and type tree mapping

The COBOL Copybook Importer operation maps data names to type tree names.

The COBOL Copybook data names are converted to type tree names according to the following rules:

- Characters that are not valid in type tree element names are replaced with the underscore (_) character. For example, the invalid \$ character in monthly\$total is replaced with the underscore (_) character. Therefore, monthly\$total becomes monthly_total.
- Duplicate names are made unique by appending one or more numeric digits to the names. For example, there are two instances of year. You can make one of the instances year1 and leave the other instance as year.

The COBOL Copybook Importer maps COBOL Copybook data types to type tree elements according to the following table. The importer operation maps FILLER items by appending a number incrementally to the end of the name. Examples are FILLER1, FILLER2, FILLER3, and so on. COBOL Copybook types that are not shown in the table are not supported by the importer.

The following table lists COBOL Copybook data types and the corresponding properties in the resulting type tree.

COBOL Copybook data type	Type tree properties
String, length n Examples of PICTURE clauses for alphanumeric data: PIC X(n) PIC A(n)	<ul style="list-style-type: none"> Item Subclass \geq Text Interpret as \geq Character Size (content) \geq Max \geq Bytes = n Pad \geq Yes Padded to \geq Fixed Size Length = n Sized As \geq Bytes
External decimal, length n , virtual places m Example of a PICTURE clause for external decimal data: PIC 9(11)V99	<ul style="list-style-type: none"> Item Subclass \geq Number Interpret as \geq Character Presentation \geq Zoned Size (digits) \geq Max \geq Bytes = $n + m$ Sign \geq Yes or No Pad \geq Yes Padded to \geq Fixed Size Length = $n + m$ Places \geq Implied = $n + m$
Packed decimal, length n , virtual places m Examples of the USAGE clause for packed-decimal data: <ul style="list-style-type: none"> USAGE COMP-3 USAGE PACKED DECIMAL 	<ul style="list-style-type: none"> Item Subclass \geq Number Interpret as \geq Binary Presentation \geq Packed Length = n Implied = m Sign \geq Yes or No
Integer, length n For the actual sizes of binary and floating point decimals, see the <i>COBOL Language</i> reference on the IBM website, http://www.ibm.com . Examples of the USAGE clause for binary integer data: <ul style="list-style-type: none"> USAGE COMP USAGE COMP-4 USAGE COMP-5 	<ul style="list-style-type: none"> Item Subclass \geq Number Interpret as \geq Binary Presentation \geq Packed Length(bytes) = n Implied = m Sign \geq Yes or No
Float and double Examples of the USAGE clause for internal floating-point data: <ul style="list-style-type: none"> USAGE COMP-1 - float, or short, occupying 4 bytes of storage USAGE COMP-2 - double, or long, occupying 8 bytes of storage 	<ul style="list-style-type: none"> Item Subclass \geq Number Interpret as \geq Binary Presentation \geq Integer Length(bytes) = n Sign \geq Yes or No
External floating point, length n Example of a PICTURE clause for external floating-point data: PIC +9(2).9(2)E+99 DISPLAY	<ul style="list-style-type: none"> Item Subclass \geq Number Interpret as \geq Binary Presentation \geq Float Length(bytes) = 4 or 8
Numeric-edited display, length n Example of a PICTURE clause for numeric-edited data: PIC \$zz,zz9.99 USAGE DISPLAY	<ul style="list-style-type: none"> Item Subclass \geq Text Interpret as \geq Character Size (content) \geq Max \geq Bytes = n Pad \geq Yes Padded to \geq Fixed Size Sized As \geq Bytes
National-edited, length n Example of a PICTURE clause for national-edited data in a date format: PIC NN/NN/NNNN USAGE NATIONAL	<ul style="list-style-type: none"> Item Subclass \geq Text Interpret as \geq Character Size (content) \geq Max \geq Bytes = n Pad \geq Yes Padded to \geq Fixed Size Length = n Sized As \geq Bytes

COBOL Copybook data type	Type tree properties
Graphic, length n Example of a PICTURE clause for graphic (G) data: PIC G (n)	<ul style="list-style-type: none"> Item Subclass \geq Text Interpret as \geq Character Size (content) \geq Max \geq Bytes = n Pad \geq Yes Padded to \geq Fixed Size Length = $2n$ Sized As \geq Bytes National Language \geq Data language \geq UTF-16
FUNCTION-POINTER, length n Example of a FUNCTION-POINTER USAGE clause: FP USAGE FUNCTION-POINTER	<ul style="list-style-type: none"> Item Subclass \geq Text Interpret as \geq Character Size (content) \geq Max \geq Bytes = n Pad \geq Yes Padded to \geq Fixed Size Length = n

Note: The COBOL Copybook Importer is not supported on Linux®.

OpenAPI importer

The OpenAPI importer creates an IBM Sterling Transformation Extender type tree from an Open API JavaScript Object Notation (JSON) document. The importer also can generate a sample map or sample Launcher system that hosts the Open API endpoints.

OpenAPI is a standard, language-agnostic interface to REST APIs, which allows you to import an OpenAPI JSON document into a type tree. OpenAPI also allows you to host REST API as endpoints in an IBM Sterling Transformation Extender map for both input (GET) and output (PUT). OpenAPI supports the following actions:

- GET (retrieve one or more tree definitions)
- POST (create a new tree or edit tree fields)
- DELETE (delete the results of a tree or the tree definition)
- PUT (replace tree fields)

There are two ways to use the OpenAPI importer:

- Use an IBM Sterling Transformation Extender map to the REST API (for more information see [Mapping to a REST API with the OpenAPI importer](#))
- Use IBM Sterling Transformation Extender Launcher to host the REST API (for more information see [Hosting a REST API with the OpenAPI importer via the Launcher](#))
- **Mapping to a REST API with the OpenAPI importer**
The OpenAPI importer creates a type tree from an OpenAPI JSON document. Optionally, you can generate a sample map or sample Launcher system that hosts the API.
- **Hosting a REST API with the OpenAPI importer via the Launcher**
The OpenAPI importer creates a type tree from an OpenAPI JSON document. You can use the IBM Transformation Extender Launcher to host the API.

Mapping to a REST API with the OpenAPI importer

The OpenAPI importer creates a type tree from an OpenAPI JSON document. Optionally, you can generate a sample map or sample Launcher system that hosts the API.

Procedure

1. [Access](#) the Importer Wizard in Design Studio
2. In the Import window, expand the Transformation Extender folder, select OpenAPI from the import source list, and click Next.
3. Enter or select the name of the JSON file that you want to import, and click Next.
4. Choose a path from the list to display the operations that are included in the path.
5. Select the operation that you want to import to generate a type tree and adapter command line properties, and click Next.
6. Select Use a Transformation Extender map to call the REST API, and click Next.
7. Configure the API connection information and click Next.

Scheme

Select the security transfer protocol to use for the API connection

Host

Specify the hostname and port of the HTTP listener in the format *hostname:port*.

Base path

Type the base path on which the API is served, which is relative to the host. If it is not included, the API is served directly under the host. The base path value must start with a leading forward slash (/).

Produces

Select the MIME types that the API can produce. This selection is global to all API calls but can be overridden on specific API calls.

8. Specify the artifact that is generated.

Folder name

Specify the folder in which the artifact will be saved, or create a new folder.

Type tree name

Type a brief descriptive name for the tree. The format is *filename-operationId.mtt*.

Note: If you do not specify an *operationId*, the HTTP operation is used.

HTTP adapter command
Type the name of the file that contains the HTTP adapter command to run. The format is *filename-operationIdHttpCommand.txt*.
Note: If you do not specify an *operationId*, the HTTP operation is used.

Generate sample map source
Click to enable the generation of a sample source map. The default is clear, which indicates that a sample map is not generated.

Map source file
If you enabled Generate sample map source, type the name of the sample map file that is generated. The format is *filename-operationIdSampleMap.mms*.
Note: If you do not specify an *operationId*, the HTTP operation is used.
Tip: The sample map allows you view everything that was generated. You can edit this map so it suits your purposes. The sample map contains three cards:

- httpRequest**
The first card includes that data you want to send to the REST API.
- execute**
The second card invokes the HTTP adapter by using the HTTP command generated by the importer, and using the information on the httpRequest card as the payload.
- httpResponse**
The third card parses the HTTP response to produce output that you can validate against the type tree that you created..

Hosting a REST API with the OpenAPI importer via the Launcher

The OpenAPI importer creates a type tree from an OpenAPI JSON document. You can use the IBM Transformation Extender Launcher to host the API.

About this task

The type tree structure that is generated is very similar to the type tree that is created when you use a map to call the REST API. The difference is that for this type tree structure, there is no definition category to describe the JSON data. Instead, the JSON is expressed as a blob by using the *httpPayload* item.

Procedure

1. [Access](#) the Importer Wizard in Design Studio.
 2. In the Import window, expand the Transformation Extender folder, select OpenAPI from the import source list, and click Next.
 3. Enter or select the name of the JSON file that you want to import, and click Next.
 4. Choose a path from the list to display the operations that are included in the path.
 5. Select the operation that you want to import to generate a type tree and adapter command line properties, and click Next.
 6. Select Use Transformation Extender Launcher to host the REST API, and click Next.
 7. Configure the API connection information and click Next.
- Scheme**
Select the security transfer protocol to use for the API connection.
- Host**
Specify the hostname and port of the HTTP listener in the format *hostname:port*.
- Base path**
Type the base path on which the API is served, which is relative to the host. If it is not included, the API is served directly under the host. The base path value must start with a leading forward slash (/).
- Produces**
Select the MIME types that the API can produce. This selection is global to all API calls but can be overridden on specific API calls.
8. Specify the artifact that is generated.
- Folder name**
Specify the folder in which the artifact will be saved, or create a new folder.
- HTTP message type tree name**
Type a brief descriptive name for the message type tree. The format is *filename-operationId.mtt*.
Note: If you do not specify an *operationId*, the HTTP operation is used.
- Data definitions type tree name**
Type a brief descriptive name for the data definitions type tree. The format is *filename-operationId.mtt*.
Note: If you do not specify an *operationId*, the HTTP operation is used.
- HTTP adapter command**
Type the name of the file that contains the HTTP adapter command to run. The format is *filename-operationIdHttpCommand.txt*.
Note: If you do not specify an *operationId*, the HTTP operation is used.
- Generate sample map source**
Click to enable the generation of a sample source map. The default is clear, which indicates that a sample map is not generated.
- Map source file**
If you enabled Generate sample map source, type the name of the sample map file that is generated. The format is *filename-operationIdSampleMap.mms*.
Note: If you do not specify an *operationId*, the HTTP operation is used. Tip: The sample map allows you view everything that was generated. You can edit this map so it suits your purposes. The sample map contains three cards:
- httpRequest**
The first card includes that data you want to send to the REST API.
 - execute**
The second card invokes the HTTP adapter by using the HTTP command generated by the importer, and using the information on the httpRequest card as the payload.
 - httpResponse**
The third card parses the HTTP response to produce output that you can validate against the type tree that you created..
- Generate sample integration flow designer library (XML format)**

Click to enable the generation of a sample integration flow designer library in XML format. The default is not selected, which indicates that a sample integration flow designer library is not generated.

XML library file

If you enabled Generate sample integration flow designer library (XML format), type the name of the sample designer flow library that is generated. The format is *filename-operationIdSampleSystem.xml*.

Note: If you do not specify an *operationId*, the HTTP operation is used.

Tip: The XML library file is a metadata object that allows for more fine-tuned XML model definitions.

9. Click Finish to generate the type tree and, if you chose to generate a sample map, generate the map.

JavaScript Object Notation (JSON) parser

You can use JavaScript Object Notation (JSON) parsing with IBM Sterling Transformation Extender Design Studio to accept a JSON document as a type tree.

Design Studio accepts a JSON document as type tree input, and creates an MTX file from the JSON document that shows all the fields in the structure that is contained in the JSON document. Depending on the data in the JSON document, two field types are supported (text and numeric). Boolean field types are displayed as text field types with `true` or `false` as the data. Any field that contains a null value is considered a text field.

Additionally, JSON documents include two JSON group types, JSON arrays and JSON objects. The default attribute for both group types is unordered (which is the default), but you can also select a sequenced attribute. Choosing the sequenced attribute causes IBM Sterling Transformation Extender to process the document in sequence order, which ameliorates any confusion that can occur with unordered processing. To change an attribute from unordered to sequenced, select the Set Default Group Class as sequenced option on the Native JSON Customization page and click Finish.

Important: Native JSON converts invalid type tree characters such as `:` and `.` to `~`. Additionally, reserved words such as `any`, `component`, `false`, `in`, `last`, `none`, `true`, and `where` are prefixed by a number sign (`#`). If a field cannot be found in the type tree at the level of the data during parsing, and no any type field is found, the parse fails with the message `Could not find type to assign JSON data`. All escaped characters except `\u` four-hex-digits are converted into actual values on input and then converted to escape characters on output. However, hex digits are not converted because they cannot be converted back on output. The validation that is performed by the JSON parser is minimal because only numeric fields need to be validated. All numeric fields have an optional leading sign and an optional decimal point. Exponential numeric fields are not supported in the core engine and fail validation. Trace and audit functions are almost identical to the IBM Sterling Transformation Extender native core engine parse, except when the JSON Parser is parsing and outputting JSON data. When JSON data fails parsing or validation, you receive a detailed message of what error occurred and where the erroneous data is in the structure. Entries in the trace file appear for all data points and groups. You can also add an AnyType group field to any level of the JSON document. The anytype group field contains two fields, a name fields and a data text field, which allows the JSON tree to handle data that was not in the original document. When you use the AnyType field, any type of data is mapped to the field. To add AnyType group fields to the JSON document, select the Add AnyType fields to every level option on the Native JSON Customization page and click Finish.

Important: Your sample JSON document must include all the fields that might occur.

Guidelines for the JSON prototype object schema file

You must follow these guidelines when you create the JSON prototype object schema file:

- The JSON file that you use for IBM Sterling Transformation Extender native JSON processing must be a prototype object that best represents the data you plan to use.
- Ensure that all fields are defined in the JSON file.
- When you use arrays, ensure that all fields that are to be used in that array are defined in the first iteration of that array.
- When you use native JSON to define the main input or output card, always select the main Type as the first JSON type.
Note: The main input or output card is the card that has the file or other source that defines how to parse or create the document.
- The encoding field on input always returns the value `NONE`. On output, this field can be set to any valid JSON encoding and the JSON document is produced that uses that encoding. If you do not configure the encoding, the default encoding used is `UTF-8`.

JSON importer

The JSON Importer is a utility for creating type trees from JavaScript Object Notation (JSON) data definitions.

The JSON Importer creates type trees that parse JSON data. You can use the type trees in your maps to validate and transform complex data.

- [Running the JSON Importer](#)

Running the JSON Importer

About this task

Use the JSON Importer to automatically generate type trees.

The metadata for the type tree is in the form of JSON data definitions.

The following procedure describes how to use the JSON Importer Wizard to generate a type tree from JSON data definitions. You specify the data import configuration properties in the Data Import page, and the communication data structure in the Importer page.

Procedure

To run the JSON Importer:

1. [Access](#) the Importer Wizard in the Design Studio application.

2. In the Import window, expand the Transformation Extender folder, select JSON from the import source list, and click Next.
3. Enter or select the name of the JSON file that you want to import.
4. Click Next.
5. Enter or select the parent folder for the type tree file that the import process creates.
6. Enter the name of the type tree file that the import process creates.
7. Click Next. The import process runs.
If the type tree was generated successfully, the Type Tree Created Successfully message displays.
8. Click Finish.
If the type tree was generated, you are given the option to open the type tree in the Type Tree editor. The Properties view for the generated type tree contains the imported data format for the generated type tree. You can also click the type tree in the Extender Navigator view or the Navigator view to view the date and time that the import operation generated the type tree, under the Info property in the Properties view.

Related tasks

- [Accessing the Importer Wizard](#)
-

PL/I Importer

The PL/I Importer is a utility for creating type trees from PL/I data definitions.

The importer works with the PL/I compiler in a Windows environment. First it creates data structures from a PL/I source file or an include file. The importer decides the file type by file extension. If the extension is "pli", the importer regards the file as "Full program". If the extension is "inc" or "mac", the importer regards the file as "Data Structure Only". The importer does not support another file extension. The importer then presents the PL/I data structures for the user to generate the type tree from the selected one. The importer will report any syntax errors in PL/I files.

- [Running the PL/I Importer](#)

Use the PL/I Importer to automatically generate PL/I type trees.

- [PL/I Importer preferences](#)

- [PL/I Language Support](#)

- [Troubleshooting](#)

- [PL/I and type tree mapping](#)

Running the PL/I Importer

Use the PL/I Importer to automatically generate PL/I type trees.

About this task

The metadata for the type tree is in the form of PL/I data structures.

If you are running the PL/I Importer to generate a type tree that is used with the CICS Adapter, you must select the CICS option.

The following procedure describes how to use the PL/I Importer Wizard to generate a type tree from PL/I files.

Procedure

To run the PL/I Importer:

1. [Access](#) the Importer Wizard in the Design Studio application.
2. In the Import window, expand the Transformation Extender folder, select PL/I Include from the import source list, and click Next.
The Type Tree Importer window opens, and you can specify the data import configuration properties.
3. If you are importing multiple PL/I structures to generate a type tree, do the following steps:
 - a. Select Multiple PL/I structures to type tree (default) from the mapping list.
 - b. Follow the Importer Wizard instructions to select the names of the metadata source files that you want to import, and to select a communication data structure.
The files display in the Multiple possible output list.
4. If you are importing a single PL/I structure to generate a type tree, do the following steps:
 - a. Select PL/I to type tree from the mapping list.
 - b. Enter or select the name of the metadata source file that you want to import, following the Importer Wizard instructions, and select a communication data structure.
The file displays in the PLI file field.
5. Click Next.
The metadata source files have a default file extension of .pli. The metadata source files have a default file extension of .pli. If a file has the .pli extension, it is assumed to be a complete PL/I program. A file is assumed to be an include file if it has the .inc or .mac extension. If the file is an include member, then it must consist of only one or more data structures. For a PL/I file to be successfully imported into your project, it must be a complete PL/I program or include file. The importer operation can process only those files with the correct file extensions; files with incorrect extensions cause an importer error.
6. Enter or select the parent folder of the generated type tree.
7. For File name, do one of the following choices:
 - To create a type tree file, enter a new name.
 - To overwrite an existing type tree file, enter its name if it does not already display in the field.
- The default name for the type tree is the PL/I file name with the .mtt extension.
8. To generate the type tree, click Next.
If you selected a file name for an existing type tree file, you are prompted to continue or stop the import operation in the Overwrite Warning window.

After the import operation runs, it displays the run results in the PL/I Importer window. If errors or warnings, or both, occur, they display in that same window. The errors and warnings are also saved to error and log files in the product installation directory. The file names are PLIimp.err and PLI.log. You can open these files and analyze them in a text editor after the import operation completes. The import operation deletes and regenerates these log files every time a new PL/I file is imported. If errors occur, the import operation cannot generate the type tree until you resolve the problem. If only warnings occur, the import operation can still generate the type tree. However, to ensure that the reported warnings do not interfere with the requirements necessary for a valid instance document, inspect the generated type tree.

9. After the type tree generation process is complete, click Finish.

If the type tree was generated, you are given the option to open the type tree in the Type Tree editor. The Properties view for the generated type tree contains the imported data format for the generated type tree. You can also click the type tree in the Extender Navigator view or the Navigator view to view the date and time that the import operation generated the type tree, under the Info property in the Properties view.

Related tasks

- [Accessing the Importer Wizard](#)

PL/I Importer preferences

You can set the PL/I compiler preferences for an individual file import through the importer wizard. You can also change the default compiler options in the PL/I Preference page.

To set the default PL/I compiler preferences, go to Window>Preferences . Expand the Importer option and select PL/I.

Under the General tab, you can modify the following parameters:

Parameter	Options	Description
Platform	<ul style="list-style-type: none"> Win32 z/OS Not Specified 	The platform on which the PL/I program runs. Selecting a platform sets the other platform Information attributes to default values that are appropriate for that platform.
Code page		The code page of the runtime data on the target platform.
Enable IMS Support	<ul style="list-style-type: none"> Select Enable IMS Support check box. Clear Enable IMS Support check box. 	The length field is recognized by the following rule: <ul style="list-style-type: none"> The name of the field is LL. The attribute of the field is a 4-byte integer, declared as FIXED BIN(31). The first field of the IMS message structure.
CICS option	<ul style="list-style-type: none"> None CICS Transaction Server for z/OS v3.1 CICS Transaction Server for z/OS v3.2 	The version of the CICS Transaction Server for z/OS if the importer is generating CICS type trees. This specification is for the runtime data on the target platform.
Floating point format	<ul style="list-style-type: none"> IEEE Binary IBM 390 Hexadecimal IEE Decimal 	The floating point format of the runtime data on the target platform.
Endian	<ul style="list-style-type: none"> Little endian Big endian 	The selection for the data character set of runtime data, which defines the way that bytes in the data are ordered.
DBCS option	<ul style="list-style-type: none"> NODBCS DBCS 	<ul style="list-style-type: none"> The NODBCS option causes the listing to show all DBCS shift-code as ":" The DBCS option ensures that the listing is sensitive to possible presence of DBCS.

Parameter	Options	Description
GRAPHIC option	<ul style="list-style-type: none"> • NOGRAPHIC • HIC • GRAPHIC 	<ul style="list-style-type: none"> The NOGRAPHIC option specifies that the source program does not contain double-byte characters. The GRAPHIC option specifies that the source program can contain double-byte characters. The hexadecimal code 0E is treated as the shift-out control code. The hexadecimal code 0F is treated as the shift-in control code. These hexadecimal codes are treated in these ways wherever they are in the source program, including occurrences in comments and string constants.
File extension support		Using this table, you can change the default extension behavior. As an example, you can assign an extension to contain a full program or data structures only.

Under the More PL/I options tab, you can modify the following parameters to change the advanced PL/I compiler preferences:

Parameter	Options	Description
Error message language	Default: US English	The language of error messages of PL/I compiler.
LIMITS	<ul style="list-style-type: none"> • NAME • EXTNAM E • FIXEDBI N • FIXEDDE C 	<ul style="list-style-type: none"> The NAME option specifies the maximum length of variable names in your program. The EXTNAM E option specifies the maximum length for EXTERNAL name. The FIXEDBIN option specifies the maximum precision for SIGNED FIXED BINARY to be either 31 or 63. The FIXEDDEC option specifies the maximum precision for FIXED DECIMAL.
MARGINS	<ul style="list-style-type: none"> • Left • Right 	<ul style="list-style-type: none"> The Left option is the column number of the leftmost character that is processed by the compiler. The Right option is the column number of the rightmost character that is processed by the compiler.
Macro preprocessor	<ul style="list-style-type: none"> • NOMACRO • MACRO • SYSPARM 	<ul style="list-style-type: none"> The NOMACRO option does not start the macro preprocessor. The MACRO option starts the macro preprocessor. The SYSPARM option is used to specify the value of the string that is returned by the macro facility built-in function SYSPARM.
Character Conversion	<ul style="list-style-type: none"> • BLANK • CURRENCY • NOT • OR 	<ul style="list-style-type: none"> The BLANK option specifies up to 10 alternative symbols for the blank character. The CURRENCY option is used to specify an alternative character to be used in picture strings instead of the dollar sign. The NOT option specifies up to seven alternative symbols that can be used as the logical NOT operator. The OR option specifies up to 7 alternative symbols as the logical OR operator. These symbols are also used as the concatenation operator, which is defined as two consecutive logical OR symbols.
NAMES		The NAMES option specifies the extralingual characters that are allowed in identifiers. Extralingual characters are those characters other than the 26 alphabetic characters, 10-digit characters, and special characters that are defined in PL/I Language Reference.

- [Enable IMS Support](#)

Enable IMS Support

If the PL/I source file that you use in your application is for an IMS application, you need to select Enable IMS Support in the PL/I importer preferences.

Once you select Enable IMS Support, the PL/I importer converts 4 bytes LL field to 2 bytes. The conversion is necessary for the following reason: An IMS PL/I application using the PLITDLI interface defines the length field as 4 bytes. IMS requires any input message coming from the client to have a field length of 2 bytes. Therefore, when a message is passed to IMS Transaction Manager to invoke the IMS PL/I application, IMS Transaction Manager edits the message and changes the 2 bytes length field to 4 bytes before passing it to the PL/I application. Therefore, when the application generates a type tree from the imported PL/I source, the original 4 bytes LL field must be converted to 2 bytes to pass the message to IMS. The PL/I importer recognizes the length field for conversion if the name of the field is LL, the field is declared as FIXED BIN(31), and it is the first field of the IMS message structure.

PL/I Language Support

Some limitations exist for the PL/I language support. Only simple REFERs are supported by the PL/I importer. The REFER usage is simple if there is only one structure element using REFER, and that element has no later siblings and no parents with siblings, and that element is:

- Either a scalar string or AREA
- A one-dimensional array of char with constant lbound
- Or an array of elements of constant size and with only the upper bound in the first dimension being non-constant

The following are examples of simple PL/I structures:

```
dcl
 1 nc1      based,
 2 ne       bin fixed(15),
 2 nx1      char( nc refer(ne) );
dcl
 1 nc2      based,
 2 ne       bin fixed(15),
 2 nf       bin fixed(15),
 2 nx2( nc refer(ne) ) char( nd refer(nf) );
dcl
 1 nc3      based,
 2 ne       bin fixed(15),
 2 nx3( nc refer (ne) , 2 ),
```

```
3 nb1      char(4),
3 nb2      char(6);
```

If the source file contains a COMPLEX REFER (which is not a SIMPLE REFER), the data structure that has the complex refer will not appear. You can see the following error message in the Problems View: Message ID: IBM2627I W Message Text: 2627 : No metadata will be generated for the structure [structure name] since its use of REFER is too complex. Note: Translation: The message is not translated, so if you are using the product in a language other than English, the message will appear as follows: Message ID: IBM2627I W.

Troubleshooting

When your application is unsuccessful in importing a PL/I file into your project, your application will display an error message. You might then correct the problem and re-import the file.

- [Error message types](#)

Error message types

The types of PL/I error messages that you might encounter are:

- PL/I files that are part of your workspace (local) that you selected when you opened the Design Studio: When your application is unsuccessful in importing a local PL/I file, which is part of the workspace, into your project, syntax error messages are displayed in the Problems view. To view the source of the error in your code, click the error message line in the Problems view. Errors are also logged in a generic log file, which is located under the metadata directory in the workspace that you selected when you opened the Design Studio: [workspace]\.metadata\CommonBaseEvents.log
- PL/I files that are not part of your workspace: If the file is not part of the workspace the syntax error messages, if any, are logged in the log file.
- Additional Error Feedback:
 - PL/I Exception.getMessageNote()**: returns the following message if the PL/I source has syntax errors: For the PL/I source which is expected to be a full program because of its file extension Possible reasons for failure: Syntax errors were found in the source. Check the problems task list for the list of errors.
 - PL/I Exception.getErrorMessageVector()**: returns a vector that consists of an ErrorMessageInfo object or objects. The purpose of ErrorMessageInfo is to give the information about a syntax error in the PL/I source. Syntax errors include: Error Message, Severity Code, Line Number, and File Location information. You get an ErrorMessageInfo object for each syntax error.
 - PL/I Exception.getMessage()**: The import of the PL/I source has failed: Check the problems task list for the list of errors.
- Data name mangling: Duplicate names are made unique by the addition of two numeric digits. For example, two instances of year become year01 and year02.

PL/I and type tree mapping

PL/I names are converted to type tree names according to the following rules:

- Characters that are not valid in type tree element names are replaced with 'x'. For example, monthly\$total becomes monthlyxtotal.
- Duplicate names are made unique by the addition of one or more numeric digits. For example, two instances of year become year and year1.

PL/I importer maps PL/I data types to type tree elements according to the following table. PL/I types that are not shown in the table are not supported by the importer. The following restrictions also apply:

- Data items with the COMPLEX, FLOAT, VARYING, and VARYINGZ attributes are not supported.
- Data items with the FLOAT attribute are supported. PL/I FLOAT IEEE is not supported.
- VARYING and VARYINGZ pure DBCS strings are supported.
- Data items that are specified as DECIMAL(p,q) are supported only when p > q.
- Data items that are specified as BINARY(p,q) are supported only when q = 0.
- If the PRECISION attribute is specified for a data item, it is ignored.
- PICTURE strings are not supported.
- ORDINAL data items are treated as FIXED BINARY(7) data types.

The following table lists PL/I data types and the corresponding properties and property values in the resulting type tree.

PL/I	Type Tree properties and property values
FIXED BINARY (n) where n = 7	<ul style="list-style-type: none"> Class._>.Item Item Subclass._>.Number Sign._>.Yes Presentation._>.Integer Interpret as._>.Binary Length(bytes)._>.1
FIXED BINARY (n) where 8 = n = 15	<ul style="list-style-type: none"> Class._>.Item Item Subclass._>.Number Sign._>.Yes Presentation._>.Integer Interpret as._>.Binary Length(bytes)._>.2

PL/I	Type Tree properties and property values
FIXED BINARY (n) where 16 = n = 31	<ul style="list-style-type: none"> • Class.>.Item • Item Subclass.>.Number • Sign.>.Yes • Presentation.>.Integer • Interpret as.>.Binary • Length(bytes).>4
FIXED BINARY (n) where 32 = n = 63	<ul style="list-style-type: none"> • Class.>.Item • Item Subclass.>.Number • Sign.>.Yes • Presentation.>.Integer • Interpret as.>.Binary • Length(bytes).>8
UNSIGNED FIXED BINARY(n) where n = 8	<ul style="list-style-type: none"> • Class.>.Item • Item Subclass.>.Number • Sign.>.No • Presentation.>.Integer • Interpret as.>.Binary • Length(bytes).>1
UNSIGNED FIXED BINARY(n) where 9 = n = 16	<ul style="list-style-type: none"> • Class.>.Item • Item Subclass.>.Number • Sign.>.No • Presentation.>.Integer • Interpret as.>.Binary • Length(bytes).>2
UNSIGNED FIXED BINARY(n) where 17 = n = 32	<ul style="list-style-type: none"> • Class.>.Item • Item Subclass.>.Number • Sign.>.No • Presentation.>.Integer • Interpret as.>.Binary • Length(bytes).>4
UNSIGNED FIXED BINARY(n) where 33 = n = 64	<ul style="list-style-type: none"> • Class.>.Item • Item Subclass.>.Number • Sign.>.No • Presentation.>.Integer • Interpret as.>.Binary • Length(bytes).>8
FIXED DECIMAL(n,m)	<ul style="list-style-type: none"> • Class.>.Item • Item Subclass.>.Number • Sign.>.No • Presentation.>.Decimal • Interpret as.>.Character • Separators.>.Yes • Places.> m, n (max)
BIT(n) where n is a multiple of 8. Other values are not supported.	<ul style="list-style-type: none"> • Class.>.Item • Item Subclass.>.Number • Sign.>.No • Interpret as.>.Binary • Length(bytes).> m (m=n/8)
CHARACTER(n)	<ul style="list-style-type: none"> • Class.>.Item • Item Subclass.>.Text • Interpret as.>.Character • Size (content).> Max > n
GRAPHIC(n)	<ul style="list-style-type: none"> • Class.>.Item • Item Subclass.>.Text • Interpret as.>.Binary • Size (content).> Max > n*2
WIDECHAR(n)	<ul style="list-style-type: none"> • Class.>.Item • Item Subclass.>.Text • Interpret as.>.Binary • Size (content).> Max > n*2
ORDINAL	<ul style="list-style-type: none"> • Class.>.Item • Item Subclass.>.Number • Sign.>.Yes • Presentation.>.Integer • Interpret as.>.Binary • Length(bytes).>1

PL/I	Type Tree properties and property values
BINARY FLOAT(n) where n <= 21	<ul style="list-style-type: none"> • Class.>.Item • Item Subclass.>.Number • Presentation.>.Float • Interpret as.>.Binary • Length(bytes).>8
BINARY FLOAT(n) where 21 < n <= 53 Values greater than 53 are not supported.	<ul style="list-style-type: none"> • Class.>.Item • Item Subclass.>.Number • Presentation.>.Float • Interpret as.>.Binary • Length(bytes).>10
DECIMAL FLOAT(n)where n <= 6	<ul style="list-style-type: none"> • Class.>.Item • Item Subclass.>.Number • Presentation.>.Packed • Interpret as.>.Binary • Length(bytes).>8
DECIMAL FLOAT(n)where 6 < n <= 16 Values greater than 16 are not supported.	<ul style="list-style-type: none"> • Class.>.Item • Item Subclass.>.Number • Presentation.>.Packed • Interpret as.>.Binary • Length(bytes).>10

Note: The PL/I importer is not supported on Linux®.

XML Schema and DTD Importers

You can create type trees by importing XML Schemas and DTDs.

Tuning Importer Behavior

The XML Schema and DTD Importers have a Java native interface and use the JVM options specified in the dtx.ini file to create a Java virtual machine (JVM). For information about configuring JVM options to tune the importer's behavior, see [Configuring JVM Options from the dtx.ini File](#).

- [XML Schema Importer](#)
- [XML DTD Importer](#)

XML Schema Importer

Use the XML Schema Importer wizard to generate a type tree from an XML Schema.

Supported Schemas

The XML Schema Importer only supports schemas that conform to the 2001 XML Schema specification defined by the Worldwide Web Consortium (W3C). Other schema options, such as BizTalk or XDR, are not supported. However, some providers such as BizTalk provide tools that can convert a proprietary schema to the latest W3C XML Schema specification.

Limitations

There is a limitation regarding the "union" XML Schema datatype. During the import process, the union datatype is converted to a text item in the type tree. As a result, you cannot create rules for mapping the member subtypes.

- [Running the XML Schema Importer](#)
Use the XML Schema Importer to automatically generate XML Schema type trees.
- [XML type tree description](#)
- [XML schema elements and attributes](#)
- [When XSDL hints are required in output](#)
- [Example using XSDL hints](#)

Related concepts

- [Attr category](#)

Running the XML Schema Importer

Use the XML Schema Importer to automatically generate XML Schema type trees.

About this task

The following procedure describes how to use the XML Schema Importer Wizard to generate a type tree from an XML Schema.

To run the XML Schema Importer:

Procedure

1. [Access](#) the Importer Wizard in the Design Studio application.
2. In the Import window, expand the Transformation Extender folder, select XML Schema from the import source list, and click Next.
The XML Schema Importer window opens.
3. Enter or select the name of the XML Schema file that you want to import, following the Importer Wizard instructions, and then click Next.
4. If applicable, select the national language that describes execution-time data. The default value is Western.
5. Under Validation, select one of the following validation types:
 - Xerces to create a version 8.0 type tree that is validated by using the Xerces parser.
By default, the importer generates all of the global elements. You can also select a subset of the global elements to generate in the type tree. For more information about this option to select a subset of global elements, see the related topic.

By default, the importer does not enable the support xsi:type attributes for derived types. You can select the option that enables the support of xsi:type attributes. For details about xsi:type attributes and derived types, see the related topics.
 - Classic to create a version 7.5.1 type tree that is validated by using classic validation.
6. When XSDL Hints are required in the resulting type tree, to specify the location for the placeholders for them, select the location in the XSDL Hints Location list. The XSDL Hints Location options are None (default), All Elements, Global Elements, and Root Elements.
7. Click Next
8. Enter or select the parent folder of the generated type tree.
9. For File name, do one of the following choices:
 - To create a type tree file, enter a new name.
 - To overwrite an existing type tree file, enter its name if it does not already display in the field.
The default name for the type tree is the XML Schema file name with the .mtt extension.
10. To generate the type tree, click Next.
If you selected a file name for an existing type tree file, you are prompted to continue or stop the import operation in the Overwrite Warning window.

After the import operation runs, it displays the run results in the XML Schema Importer window. If errors or warnings, or both, occur, they display in that same window. The errors and warnings are also saved to a log file in the product installation directory. The file name is either xmlsimp.log or ttmaker.log, depending on where in the process the errors and warnings occur. You can open this file and analyze it in a text editor after the import operation completes. The import operation deletes and regenerates these log files every time a new XML Schema is imported. If errors occur, the import operation cannot generate the type tree until you resolve the problem. If only warnings occur, the import operation can still generate the type tree. However, to ensure that the reported warnings do not interfere with the requirements necessary for a valid instance document, inspect the generated type tree.
11. After the type tree generation process is complete, click Finish.
If the type tree was generated, you are given the option to open the type tree in the Type Tree editor. The Properties view for the generated type tree contains the imported data format for the generated type tree. You can also click the type tree in the Extender Navigator view or the Navigator view to view the date and time that the import operation generated the type tree, under the Info property in the Properties view.

- [About classic validation](#)

Related concepts

- [Global elements](#)
- [xsi:type attribute](#)
- [Use of derived types on base types by using xsi:type attributes](#)

Related tasks

- [Accessing the Importer Wizard](#)

Related reference

- [xsi:type support for abstract and concrete data types](#)
- [When XSDL hints are required in output](#)

About classic validation

For information about "Classic" validation, see the documentation provided with IBM Sterling Transformation Extender version 7.5.1.

The following changes to the type tree structure are not included in the version 7.5.1 documentation:

- If a group representing a compositor (xs:sequence, xs:choice, xs:all) has a single component with a range (1:1), the group is removed.
- If a type tree contains a structure like the following structure:

```
- choice (1:1)
- Seq1
- element (0:s)
- Seq2
- element (0:s)
```

It is transformed into the following structure:

```
- choice (0:1)
- Seq1
- element (1:s)
```

```
- Seq2
  - element (1:s)
```

XML type tree description

Using Xerces validation, the root of an imported type tree is named XSD.

- [Misc category](#)
 - [Prolog category](#)
 - [Attr category](#)
 - [Type category](#)
 - [Element category](#)
 - [Doc group](#)
 - [Global](#)
-

Misc category

The **Misc** category represents generic types that can be contained in XML instance documents.

The **Comment** text item represents an XML comment. The **PI** text item represents an XML processing instruction. **Name** and **value** represent global attributes that will be added in the placeholder group created for wildcard schema elements anyAttribute or anyType (see "Element and Attribute Wildcards" in the Type Designer documentation for information about wildcards). **PCDATA** represents a PCDATA section in the instance document.

Prolog category

The **Prolog** category contains the type tree constructs used to describe the XML prolog in the instance document.

Version, **encoding**, and **standalone** items represent attributes in the XML prolog.

The **Decl** unordered group holds the **version**, **encoding** and **standalone** items. **Version** is mandatory, while **encoding** and **standalone** are optional within the **Decl** group.

Attr category

The **Attr** category contains type tree constructs generated from a Schema document that represents attributes from the XML Schema Instance namespace, such as **nil**, **noNamespaceSchemaLocation**, **schemaLocation**, and **type**.

Related concepts

- [XML Schema Importer](#)
-

Type category

The **Type** category contains constructs that represent the global types in the schema.

The complex type **Address** contains a sequence of elements. The components of the sequence are organized in the **Comp** category. The sequence in the **Address** type has three elements that are represented by these item types in the type tree: **city**, **name**, **street**. The **TypeDef** group from the **Address** category contains the text items that have been defined in its **Comp** category.

The complex type **USAddress** is an extension of the **Address** type that adds two more elements: **state** and **zip**.

- The item **zip** is a numeric item. Its type in the schema is "positiveInteger".
- The item **state** has the **USState** type in the schema. **USState** is defined as a text item, but it has a set of allowed values. To see these values, double-click on the **USState** item.

The complex type **UKAddress** is also an extension of the **Address** type in the schema, but it adds the element **UKPostcode** and the attribute **exportCode**. It is represented in type tree by the **UKAddress** category, that besides the **Comp** category, contains the **Attr** category and the **AttrList** group. The **AttrList** group contains the items that have been defined in the **Attr** category-in this case the **exportCode** numeric item.

Element category

The **Element** category contains constructs that represent the global elements defined in the schema.

This example has only one global element (**Person**) which is why the **Element** category has only one child node (the **Person** category).

Doc group

The **Doc** group represents the whole XML instance document.

This includes the **Prolog** of the document and the top-level element of the instance document - **ElementDecl** group of the **Person** element.

In preceding image, the example has only one global element declared in the schema (**Person**). For that reason it is directly included in the **Doc** group.

If more than one global element was declared in the schema, a choice group named **Global** would be defined as a choice selection for the elements because the XML instance document must have only one root element. In that case, this **Global** group would be included in the **Doc** group along with **Prolog**.

Global

The Global group is present in the imported type tree when the input XML Schema contains more than one global element.

Based on the preceding example, if a second global element named "address" is added, the imported type tree would contain the choice group Global.

```
<element name="address" type="ipo:Address"/>
```

The components of the Global group are all globally defined XML Schema elements.

XML schema elements and attributes

This section lists type properties in the resulting type tree that are created from the imported XML Schema elements and attributes.

- [XML schema elements](#)
- [Primitive datatypes](#)
- [Derived datatypes](#)
- [XML schema constraining facets](#)
- [Global elements](#)
- [xsi:type attribute](#)

XML schema elements

The following table lists XML Schema elements with corresponding types created in the resulting type tree.

XML Schema Element	Type Tree Type Class
all	Unordered group named TypeDef
annotation	(The comments in the XML Schema are ignored during the import process.)
any	Text item named anyElem
anyAttribute	Sequence group named anyAttr containing name and value text items
appinfo	(Application information is ignored during the import process.)
attribute	Text item
attributeGroup	Category containing a Comp category and a AttrGroupDecl group
choice	Category containing a Comp category and a TypeDef choice group
complexContent	Sequence Group named TypeDef
complexType	Category containing at a minimum a Comp category and a TypeDef sequence group
documentation	(Text comments are ignored during the import process.)
element	This is a group when the element definition contains a reference to a global type, otherwise, it is a category
extension	Sequence froup named TypeDef
field	Property named Field Path in the Intent > Validate As > Derivation > ID Constraints property group.
group	Category containing a Comp category and a GroupDecl group
import	(Not represented)

```

include
    (Not represented)
key
    Group property under Intent > Validate As > Derivation > ID Constraints (named Key)
keyref
    Group property under Intent > Validate As > Derivation > ID Constraints (named KeyRef)
list
    Category containing a Comp category and a TypeDef sequence group with the range 0-s
notation
    (Non-XML data is ignored during the import process.)
redefine
    (Not represented)
restriction
    Text item
schema
    XSD type category
selector
    Group property under Intent > Validate As > Derivation > ID Constraints (named Selector Path)
sequence
    Sequence group named TypeDef
simpleContent
    Sequence group named TypeDef
simpleType
    Text item
union
    Text item
unique
    Text item

```

Primitive datatypes

The following table lists primitive XML Schema datatypes and the corresponding properties in the resulting type tree.

XML Schema Type	Item Subclass, Interpret as, Presentation	Description
anyURI	Text, Character	
base64Binary	Text, Character	
boolean	Text, Character	Restricted to the values: "0", "1", "true" or "false"
date	Date & Time, Character	Custom date format: {CCYY-MM-DD}
dateTime	Date & Time, Character	Custom date/time format: {CCYY-MM-DD}T{HH24:MM:SS[.0-6]}
decimal	Number, Character, Decimal	Custom separator format: #####[.##]
double	Number, Character, Decimal	Custom separator format: #####[.##]
duration	Text, Character	
float	Number, Character, Decimal	Custom separator format: #####[.##]
gDay	Text, Character	
gMonth	Text, Character	
gMonthDay	Text, Character	
gYear	Date & Time, Character	Custom date format: {CCYY}
gYearMonth	Date & Time, Character	Custom date format: {CCYY-MM}
hexBinary	Text, Character	
NOTATION	Text, Character	
QName	Text, Character	
string	Text, Character	
time	Date & Time, Character	Custom time format: {HH24:MM:SS[.0-6]}

Derived datatypes

The following table lists derived XML Schema datatypes and corresponding properties in the resulting type tree.

XML Schema Type	Item Subclass, Interpret as, Presentation	Restrictions
byte	Number, Character, Integer	Size: <ul style="list-style-type: none"> • Min: 1 • Max: 3 Restriction: Range; Rule: Include <ul style="list-style-type: none"> • Min: -128 • Max: 127
ENTITIES	(Group of text items)	
ENTITY	Text, Character	

XML Schema Type	Item Subclass, Interpret as, Presentation	Restrictions
ID	Text, Character	
IDREF	Text, Character	
IDREFS	(Group of text items)	
int	Number, Character, Integer	<p>Size:</p> <ul style="list-style-type: none"> • Min: 1 • Max: 10 <p>Restriction: Range; Rule: Include</p> <ul style="list-style-type: none"> • Min: -2147483648 • Max: 2147483647
integer	Number, Character, Integer	
language	Text, Character	
long	Number, Character, Integer	<p>Size:</p> <ul style="list-style-type: none"> • Min: 1 • Max: 19 <p>Restriction: Range; Rule: Include</p> <ul style="list-style-type: none"> • Min: -9223372036854775808 • Max: 9223372036854775807
Name	Text, Character	
NCName	Text, Character	
negativeInteger	Number, Character, Integer	<p>Restriction: Range; Rule: Include</p> <ul style="list-style-type: none"> • Max: -1
NMOKEN	Text, Character	
nonNegativeInteger	Number, Character, Integer	<p>Restriction: Range; Rule: Include</p> <ul style="list-style-type: none"> • Min: 0
nonPositiveInteger	Number, Character, Integer	<p>Restriction: Range; Rule: Include</p> <ul style="list-style-type: none"> • Max: 0
normalizedString	Text, Character	
positiveInteger	Number, Character, Integer	<p>Restriction: Range; Rule: Include</p> <ul style="list-style-type: none"> • Min: 1
short	Number, Character, Integer	<p>Size:</p> <ul style="list-style-type: none"> • Min: 1 • Max: 5 <p>Restriction: Range; Rule: Include</p> <ul style="list-style-type: none"> • Min: -32768 • Max: 32767
token	Text, Character	
unsignedByte	Number, Character, Integer	<p>No sign</p> <p>Size:</p> <ul style="list-style-type: none"> • Min: 1 • Max: 3 <p>Restriction: Range; Rule: Include</p> <ul style="list-style-type: none"> • Max: 255
unsignedInt	Number, Character, Integer	<p>No sign</p> <p>Size:</p> <ul style="list-style-type: none"> • Min: 1 • Max: 10 <p>Restriction: Range; Rule: Include</p> <ul style="list-style-type: none"> • Max: 4294967295

XML Schema Type	Item Subclass, Interpret as, Presentation	Restrictions
unsignedLong	Number, Character, Integer	No sign Size: <ul style="list-style-type: none">• Min: 1• Max: 20 Restriction: Range; Rule: Include <ul style="list-style-type: none">• Max: 18446744073709551615
unsignedShort	Number, Character, Integer	No sign Size: <ul style="list-style-type: none">• Min: 1• Max: 5 Restriction: Range; Rule: Include <ul style="list-style-type: none">• Max: 65535

XML schema constraining facets

The following table lists XML Schema constraining facets and the corresponding properties in the resulting type tree.

XML Schema Facet	Item Subclass, Interpret as, Presentation	Restrictions
enumeration	Text, Character	Restrictions: Value; (Rule) Include: --01-01 --07-04 --12-25
fractionDigits	Number, Character, Decimal	Custom separator format: #####[.##] Data language: ASCII
length	Text, Character	Size: <ul style="list-style-type: none">• Min: 8• Max: 8
maxExclusive	Number, Character, Integer	Restriction: Range; Rule: Include Max: NOT 101 (The number 101 is not included and the valid exclude range is from 1 to any value less than 101.)
maxInclusive	Number, Character, Integer	Restriction: Range; Rule: Include <ul style="list-style-type: none">• Max: 100
maxLength	Text, Character	Size: <ul style="list-style-type: none">• Min: 0• Max: 20
minExclusive	Number, Character, Integer	Restriction: Range; Rule: Include Min: NOT 99 (99 is not included as the minimum value. The range therefore extends from any value greater than 99 to the maximum value specified.)
minInclusive	Number, Character, Integer	Restriction: Range; Rule: Include <ul style="list-style-type: none">• Min: 100
minLength	Text, Character	Size: <ul style="list-style-type: none">• Min: 1
pattern	Text, Character	
totalDigits	Number, Character, Decimal	Size: <ul style="list-style-type: none">• Min: 0• Max: 8 Custom separator format: #####[.##] Data language: ASCII
whiteSpace	Text, Character	

Global elements

You can select specific global elements to be generated in the type tree by using the XML Schema Importer wizard. Each global element is presented as a `global_element_name {namespace}` value.

Related tasks

- [Running the XML Schema Importer](#)

xsi:type attribute

An element in an instance document can explicitly assert its type by using the `xsi:type` attribute [1](#).

- [xsi:type support for abstract and concrete data types](#)
- [Use of derived types on base types by using xsi:type attributes](#)

Related tasks

- [Running the XML Schema Importer](#)

xsi:type support for abstract and concrete data types

IBM Sterling Transformation Extender supports the `xsi:type` attribute.

You can use the `xsi:type` attribute for substitution groups, elements that are defined as either abstract or concrete, or elements that are defined by either abstract or concrete types. The support for `xsi:type` creates choices for these types that can receive the `xsi:type` attribute at run time.

For example:

- When a type is defined as abstract, the corresponding imported type tree type is created as a choice of all the types that extend the abstract type.
- When a type is defined as concrete, the corresponding imported type tree type is created as a choice of the concrete type and all the types that extend the concrete type.

In the following Schema fragment, the type `Address` is defined as abstract and the types `US-address` and `UK-address` are simple extensions of the abstract type. An `address` element in an instance document must contain an `xsi:type` attribute to specify a concrete type. The validation library uses the `xsi:type` attribute to determine the real type that is specified and validate the type correctly.

```
<!-- abstract type -->
<x:complexType name="Address" abstract="true">
  <x:sequence>
    <x:element name="street" type="xs:string" />
    <x:element name="city" type="xs:string" />
  </x:sequence>
</x:complexType>
<!-- concrete types -->
<x:complexType name="US-address">
  <x:complexContent>
    <x:extension base="Address">
      <x:sequence>
        <x:element name="zip" type="xs:integer" />
      </x:sequence>
    </x:extension>
  </x:complexContent>
</x:complexType>
<x:complexType name="UK-address">
  <x:complexContent>
    <x:extension base="Address">
      <x:sequence>
        <x:element name="postalCode" type="xs:string" />
      </x:sequence>
    </x:extension>
  </x:complexContent>
</x:complexType>
<!-- abstract type -->
<x:element name="address" type="Address">
```

Related tasks

- [Running the XML Schema Importer](#)

Use of derived types on base types by using xsi:type attributes

You can select the option that enables the support of xsi:type attributes.

Each global type that has a derived type is presented as a *global_type_name {namespace}* value.

Related tasks

- [Running the XML Schema Importer](#)
-

When XSDL hints are required in output

During the XML Schema import process, when you do not accept the default **XSDL Hints Location** setting (**None**), the following choices are listed:

- **All Elements**: Creates XSDL hints placeholders for all types described by the XML Schema.
- **Global Element**: Creates XSDL hints placeholders for all global elements.
- **Root Element**: Do not use.

When the actual value of the XSDL hint is required in the output, you must follow specific instructions.

For example, during the import process you select Global Element for the **XSDL Hints Location** value. In the generated type tree, all global types (represented in the type tree as components of the **XSD Element** category) contain in their **AttrList** group at least two optional text items: **xsi-noNamespaceSchemaLocation** and **xsi-schemaLocation**.

For either of these items, you can create a map rule that sends the XSDL hint to the output. For example, when a map rules sends the **schemaLocation** XSDL hint to the output, the output looks similar to:

```
<ipo-out:purchaseOrders
  xmlns:ipo-out="http://www.example.com/IPO"
  xsi:schemaLocation="http://www.example.com/IPO ipo.xsd">
```

When the <http://www.w3.org/2001/XMLSchema-instance> namespace is not declared in the input XML Schema, you must manually add the definition in the output card.

To add a namespace to the output card:

1. Open the output card.
2. Expand the **Type** setting.
3. Under Schema Type > Name Spaces is a list of the namespaces identified in the input Schema.
4. Right-click the **Name Spaces** setting and choose **Add** from the context menu.
5. On the new line, enter <http://www.w3.org/2001/XMLSchema-instance> in the Setting column and **xsi** in the Value column and click OK.
6. Save, build, and run the map.

The resulting XML document has a header that looks similar to the following text:

```
<?xml version="1.0"?>
<ipo-out:purchaseOrders
  xmlns:ipo-out="http://www.example.com/IPO"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:SchemaLocation="http://www.example.com/IPO ipo.xsd">
```

See the Map Designer documentation for information about how to add XSDL hints to an output card when you are using native XML schema validation instead of type trees and Xerces validation.

Related tasks

- [Running the XML Schema Importer](#)
-

Example using XSDL hints

During the XML Schema import process, when you select a value other than "None" for the **XSDL Hints Location**, extra steps are required if the actual value of the XSDL Hints are required in the output.

1. From the Type Designer Startup window, select Import a type tree and click OK.
The Importer Wizard is displayed.
2. Select XML Schema and click Next.
3. Select the following example file and click Next:
install_dir\examples\xml\ipo.xsd
4. At the prompt to specify a national language, accept the default (Western) and click Next.
5. In the Filename field, enter a filename and path for the type tree (.mtt) you are about to create. For example, enter:
install_dir\examples\xml\ipo_global_elements.mtt
6. Under Validation, select Xerces.
7. For the XSDL Hints Location, select Global Elements.
8. Click Next.
9. When type tree generation is complete, click Finish and open the type tree.

All global types contain at least two optional text items in their **AttrList** group (to which values can be assigned in a map as map rules):
xsi~noNamespaceSchemaLocation and xsi~schemaLocation.

10. Using the Map Designer, open the following example map:
`install_dir\examples\xml\ipo.mms`

11. Update the input and output cards of validationMap with the ipo_global_elements.mtt type tree.

12. In the output card, delete the existing rule: **Output = Input**.

13. Enter new map rules in the output card. The **xsi~schemaLocation** attribute should be equal to <http://www.example.com/IPO ipo.xsd>.

This rule causes the output document to contain the schemaLocation XSDL hint (<http://www.example.com/IPO ipo.xsd>).

```
<ipo-out:purchaseOrders  
    xmlns:ipo-out="http://www.example.com/IPO"  
    xsi:schemaLocation="http://www.example.com/IPO ipo.xsd">
```

See the Map Designer documentation for information about how to add XSDL hints to an output card when you are using native XML schema validation instead of type trees and Xerces validation.

When a namespace is not declared in an input schema, you must manually define it in the output card. See "[To add a namespace to the output card](#)" for instructions.

XML DTD Importer

Use the XML DTD Importer to create type trees that describe the format of XML data.

Many of the type trees that are generated by the Importer Wizard can be immediately used for map development. However, depending on the contents of the interface-specific metadata file, it might be necessary for the generated type tree to be modified using the Type Designer.

You cannot simultaneously have a type tree open in the Type Designer while importing a type tree with the same name.

- [Running the XML DTD Importer](#)
Use the XML DTD Importer to automatically generate XML DTD type trees.
- [DTD elements and attributes in the type tree](#)

Running the XML DTD Importer

Use the XML DTD Importer to automatically generate XML DTD type trees.

About this task

The following procedure describes how to use the XML DTD Importer Wizard to generate a type tree from an XML DTD.

To run the XML DTD Importer:

Procedure

1. [Access](#) the Importer Wizard in the Design Studio application.
2. In the Import window, expand the Transformation Extender folder, select XML DTD from the import source list, and click Next.
The XML DTD Importer window opens.
3. Enter or select the name of the XML DTD file that you want to import, following the Importer Wizard instructions, and then click Next.
4. If applicable, select the national language that describes execution-time data. The default value is Western.
5. Under Validation, select one of the following validation types:
 - Xerces to create a version 8.0 type tree that is validated with the Xerces parser.
 - Classic to create a version 7.5.1 type tree that is validated with classic validation.
6. To create a placeholder for the document type (DOCTYPE) declaration in the resulting type tree, select the Generate DOCTYPE option.
7. Click Next.
8. Enter or select the parent folder of the generated type tree.
9. For File name, do one of the following choices:
 - To create a type tree file, enter a new name.
 - To overwrite an existing type tree file, enter its name if it does not already display in the field.
The default name for the type tree is the XML DTD file name with the .mtt extension.
10. To generate the type tree, click Next.
If you selected a file name for an existing type tree file, you are prompted to continue or stop the import operation in the Overwrite Warning window.
- After the import operation runs, it displays the run results in the XML DTD Importer window. If errors or warnings, or both, occur, they display in that same window. The errors and warnings are also saved to a log file in the product installation directory. The file name is either dtd.log or ttmaker.log, depending on where in the process the errors and warnings occur. You can open this file and analyze it in a text editor after the import operation completes. The import operation deletes and regenerates these log files every time a new XML DTD is imported. If errors occur, the import operation cannot generate the type tree until you resolve the problem. If only warnings occur, the import operation can still generate the type tree. However, to ensure that the reported warnings do not interfere with the requirements necessary for a valid instance document, inspect the generated type tree.
11. After the type tree generation process is complete, click Finish.
If the type tree was generated, you are given the option to open the type tree in the Type Tree editor. The Properties view for the generated type tree contains the imported data format for the generated type tree. You can also click the type tree in the Extender Navigator view or the Navigator view to view the date and time that the import operation generated the type tree, under the Info property in the Properties view.

Related tasks

- [Accessing the Importer Wizard](#)
-

DTD elements and attributes in the type tree

In the following image you can see a pattern of how DTD elements and attributes transform into types in the type tree for both Xerces and classic 7.5.1 validation. In this image you can also see some structural differences in the construction of a classic and Xerces type tree.

WSDL Importer

The Web Services Description Language (WSDL) Importer is a utility for generating type trees from message definitions contained within WSDL documents.

Tuning Importer behavior

The WSDL Importer has a Java™ native interface and uses the JVM options specified in the dtx.ini file to create a Java virtual machine (JVM). For information about configuring JVM options to tune the importer's behavior, see ["Configuring JVM options from the dtx.ini file"](#).

Import options

The WSDL import process can create a classic type tree or a native XML schema (.xsd) file from the WSDL document. You select the type of output in the WSDL Importer wizard.

- When the WSDL document structure requires support for derived types (such as through the xsi:type attribute), create a native XML schema.
- When the WSDL document structure does not require support for derived types, create a classic XML type tree. Classic type trees are smaller than native XML schemas.

Classic type tree processing

The importer reads a WSDL document and assembles lists of port types, operations, services, messages, and bindings. When the parsing is complete, the importer creates a map for each web service operation and adds it to the resulting XML map source file. During this process, the XML Schema Importer is used to create the types defined in the input WSDL document.

After the WSDL Importer imports the WSDL document, it creates a WSDL type tree (.mtt) file and an XML (*_map.xml) file. Use File > Import > Map from XML in the Design Studio to import the XML file to create a map. The import process creates a map (*WSDLname.map.mms*) that uses the new WSDL type tree in the output cards.

Native XML schema processing

When you select native XML schema validation in the wizard, the WSDL Importer imports the WSDL document and creates two XML schemas:

WSDLname.xsd
Contains the XSD and WSDL-based schema types.

WSDLname_soap.xsd
Contains the SOAP envelope, header, and body types.

Use the *WSDLname_soap.xsd* as the card's native type tree.

- [Support for SOAP arrays](#)
- [WSDL document elements](#)
- [WSDL document types](#)
- [Running the WSDL Importer](#)
Use the WSDL Importer to automatically generate type trees or XML schemas.
- [Special symbols](#)
- [Character restrictions](#)
- [WSDL example](#)

Support for SOAP arrays

The WSDL Importer can process SOAP arrays and produce relevant type trees when the type of the array's elements is known. When a SOAP array derived type is found, the value of the arrayType attribute (if present) is used to create a reference to the defined type.

Type tree representation	Type definition
- ElemDecl + AttrList - Wrapper - Group (0:1) - Seq (0:1) valueOf(arrayType) (s)	<x:element name="arrayOfThings"> <x:complexType> <x:complexContent> <x:restriction base="tns:Array"> <x:sequence> <x:element name=="int" maxOccurs="unbounded"/> </x:sequence> <x:attribute ref="tns:arrayType" wsdl:arrayType="xs:int[]"/> </x:restriction> </x:complexContent> </x:complexType> </x:element>

WSDL document elements

A WSDL document typically contains the elements listed in the following table.

Element	Description
message	Message definitions
types	Type definitions. Normally XML Schema.
portType	List of operations available for the port.
binding	Definitions of how operations are bound to transports.
service	Defines the ports offered by the service.

WSDL document types

There are two different types of WSDL documents:

- WSDL Service Interface documents
- WSDL Service Implementation documents.

WSDL Service Interface documents contain **type**, **portType**, and **binding** elements.

The WSDL Service Implementation contains import and service elements, as well as a description of a service that implements a service interface. The **import** element imports a WSDL Service Interface definition.

The WSDL Importer gets its information from the import service interface and not from the service implementations. Both document types can be combined into a single document. If both document types are not combined into a single document, the Service Implementation document will include an import statement to import the Service Interface document. If a Service Implementation document is processed by the importer, the importer must be able to import the Service Interface document to obtain the message definitions.

Running the WSDL Importer

Use the WSDL Importer to automatically generate type trees or XML schemas.

Procedure

To run the WSDL Importer:

1. [Access](#) the Importer Wizard in the Design Studio application.
2. In the Import window, expand the Transformation Extender folder, select WSDL from the import source list, and click Next.
The WSDL Importer window opens.
3. Enter the Uniform Resource Identifier (URI) of the WSDL document that you want to import or select a WSDL document, following the Importer Wizard instructions, and then click Next. When you are importing a WSDL document that references a URI on another machine, access to that other machine might be required. For example, if the WSDL document imports type definitions with a reference to an HTTP-based URL, network access to that referenced URL is required.
4. If applicable, select the national language that describes execution-time data. The default value is Western.
5. Choose to generate either a classic type tree or a native XML schema.
6. Click Next.
7. Select one or more operations that are defined for the specific class to be run in the map, and click Next.
8. Enter or select the parent folder of the generated type tree or XML schema.
9. Accept the default file name or enter a new name.
 - For classic type tree generation, the default file name is *WSDLname.mtt*.
 - For XML schema generation, the default file name is *WSDLname.xsd*.
10. To generate the type tree, click Next.
After the import operation runs, it displays the run results in the WSDL Importer window. If errors or warnings, or both, occur, they display in that same window. The errors and warnings are also saved to a log file in the product installation directory. The file name is *wsdlimp.log*. You can open this file and analyze it in a text editor after the import operation completes. The import operation deletes and regenerates this log file every time a new WSDL is imported. If errors occur, the import operation cannot generate the type tree or XML schema until you resolve the problem. If only warnings occur, the import operation can still generate the type tree or XML schema. However, to ensure that the reported warnings do not interfere with the requirements necessary for a valid instance document, inspect the generated type tree or XML schema.
11. After the generation process is complete, click Finish.
You can open the type tree to display the imported data format in the Properties view. Click the type tree in the Extender Navigator view or the Navigator view to display the time stamp of the import operation and other details in the Properties view.

Related tasks

- [Accessing the Importer Wizard](#)

Special symbols

After importing a **.wsdl** file, you might see the <IGNORE>, <QUOTE>, and <ANYQUOTE> symbols as initiators or terminators in the resulting type tree.

- [ANYQUOTE and QUOTE](#)
- [IGNORE](#)

ANYQUOTE and QUOTE

<ANYQUOTE> and <QUOTE> symbols are used to allow for pairs of double quotes or pairs of single quotes.

This mechanism is needed, for example, to allow for both double-quote pairs and single-quote pairs for the attributes in XML data.

During the data validation process, the <ANYQUOTE> symbol can match both double and single quote characters, and <QUOTE> must match the same quote character (single or double) that was recognized by the last <ANYQUOTE> symbol.

This logic is used to allow for pairs of matching double or single quotes and to prevent the pairs with mixed double and single quotes.

- [ANYQUOTE and QUOTE examples](#)

ANYQUOTE and QUOTE examples

When a text item is defined with initiator <ANYQUOTE> and terminator <QUOTE>, the following data validates successfully.

"John"

The following data also validates successfully:

'John'

However, the following data fails validation:

"John'

In the failed result, the initial double quote is validated by <ANYQUOTE> and the value John validates successfully as the value of the text item. However, the last single quote fails validation because it is represented in the type tree by the <QUOTE> symbol and the previous <ANYQUOTE> symbol validated a double quote character.

If <QUOTE> is encountered during data validation, and no <ANYQUOTE> symbol was encountered before that, both double quote and single quote characters in data will be accepted as valid.

IGNORE

The <IGNORE> symbol is used by the validation engine to ignore a portion of the input data. The syntax is:

<IGNOREcPar1cPar2cPar3>

Option	Description
c	A character used as a separator for Par1, Par2 and Par3 values.
Par1	A string that is expected to appear at the beginning of the data that is to be ignored.
Par2	A string that is mapped on output as a result of the <IGNORE> symbol.
Par3	A string that is expected to appear at the end of ignored data. If it is omitted, the ignored data ends when what follows the <IGNORE> symbol in the type tree is recognized in the data on input.

- [IGNORE examples](#)

IGNORE examples

In the following examples, a comma (,) is used as the separator among Par1, Par2, and Par3 <IGNORE> values (this is the c character in the <IGNORE> syntax).

Example 1

A text item is defined with this initiator: <IGNORE, ab, cdef, gh>John

The following data validates successfully:

abghJohnSmith

The following data is built on output:

cdefJohnSmith

Example 2

A text item is defined with this initiator: <IGNORE,,xsd:,,:>Bill

The following data validates successfully:

```
xs:BillSmith
```

The following data is built on output:

```
xsd:BillSimth
```

Example 3

A text item is defined with initiator <<IGNORE,,,,:>Color> and terminator </Color>.

The following data validates successfully:

```
<ab:Color>Blue</Color>
```

The following data is built on output:

```
<Color>Blue</Color>
```

Example 4

A text item is defined with initiator <<IGNORE,,,,:>Color> and terminator </Color>.

The following data validates successfully:

```
<Color>Blue</Color>
```

Although the colon (:) does not appear anywhere in the input data, the validation process recognizes that no data should be ignored because the **Color** value in the data matches the **Color** value in the initiator, so in that case, the <IGNORE> symbol in the initiator will not ignore any data on input.

On output, the data is the same as on input:

```
<Color>Blue</Color>
```

This is because the second parameter (Par2) in the <IGNORE> symbol was specified as "empty".

Character restrictions

You can customize a character restrictions configuration file that is utilized by the WSDL Importer.

- [Using a configuration file for character restrictions](#)
 - [XML configuration file and schema](#)
 - [Using character restrictions](#)
 - [Understanding character restriction ordering](#)
 - [Multiple character restriction sets](#)
-

Using a configuration file for character restrictions

The WSDL Importer uses a configuration file during type tree generation that you can customize to include specific Exclude character restrictions.

Character restriction lists often contain entity references, such as & (ampersand) or ' (apostrophe) that are special characters in XML. If you use the same character restrictions across multiple type trees, using a configuration file to specify these character restrictions eliminates the need to create an individual restriction list for each type tree.

XML configuration file and schema

The XML-based configuration file (xmlsimp.xml) and XML Schema (xmlsimp.xsd) are installed with the Design Studio. These files work together to specify Exclude character restrictions for items in a type tree that represent values for simple elements and attributes in XML documents.

- [XML schema](#)
 - [XML configuration file](#)
 - [Customizing the configuration file](#)
-

XML schema

The xmlsimp.xsd schema file defines a valid structure for the xmlsimp.xml configuration file. This schema file, located in the Design Studio installation directory, must not be modified or moved because it is used by the importers to validate the xmlsimp.xml configuration file during the import process. The absence of this schema file will cause the import process to fail; modifying the file can cause unexpected results, such as the creation of an invalid type tree.

XML configuration file

The `xmlsimp.xml` configuration file, which specifies character restrictions for item types in a type tree, is also located in the Design Studio installation directory and must not be moved. This file can be encoded using any encoding scheme supported by the IBM alphaWorks XML4J 3.2.1. (For a full list of supported encodings, consult XML4J documentation.)

The default content of the `xmlsimp.xml` configuration file specifies the five entity references supported by XML (apostrophe, quotation mark, greater-than, less-than, ampersand) as Exclude characters, as shown:

```
<!-- The default character restrictions -->
<character_restrictions name="default_cs">
  <elements>
    <character_restriction exclude="&apos;" reference_string="&#39;"/>
    <character_restriction exclude="&apos;" reference_string="&#39;"/>
    <character_restriction exclude="&quot;" reference_string="&#34;"/>
    <character_restriction exclude="&quot;" reference_string="&#34;"/>
    <character_restriction exclude="&gt;" reference_string="&gt;"/>
    <character_restriction exclude="&gt;" reference_string="&#62;"/>
    <character_restriction exclude="&lt;" reference_string="&lt;"/>
    <character_restriction exclude="&lt;" reference_string="&#60;"/>
    <character_restriction exclude="&amp;" reference_string="&amp;#amp;"/>
    <character_restriction exclude="&amp;" reference_string="&#38;"/>
    <character_restriction exclude="&apos;" reference_string="&apos;"/>
    <character_restriction exclude="&quot;" reference_string="&quot;"/>
  </elements>
  <attributes>
    <character_restriction exclude="&apos;" reference_string="&#39;"/>
    <character_restriction exclude="&apos;" reference_string="&#39;"/>
    <character_restriction exclude="&quot;" reference_string="&#34;"/>
    <character_restriction exclude="&quot;" reference_string="&#34;"/>
    <character_restriction exclude="&gt;" reference_string="&gt;"/>
    <character_restriction exclude="&gt;" reference_string="&#62;"/>
    <character_restriction exclude="&lt;" reference_string="&lt;"/>
    <character_restriction exclude="&lt;" reference_string="&#60;"/>
    <character_restriction exclude="&amp;" reference_string="&amp;#amp;"/>
    <character_restriction exclude="&amp;" reference_string="&#38;"/>
  </attributes>
</character_restrictions>
```

Customizing the configuration file

You can edit the `xmlsimp.xml` configuration file to suit your requirements. For example, if you process Latin-1 characters, you might add the eacute (é) character to your Exclude list as shown:

```
<character_restrictions name="latin1">
  <elements>
    <character_restriction exclude="&apos;" reference_string="&#39;"/>
    <character_restriction exclude="&apos;" reference_string="&#39;"/>
    <character_restriction exclude="&quot;" reference_string="&#34;"/>
    <character_restriction exclude="&quot;" reference_string="&#34;"/>
    <character_restriction exclude="&gt;" reference_string="&gt;"/>
    <character_restriction exclude="&gt;" reference_string="&#62;"/>
    <character_restriction exclude="&lt;" reference_string="&lt;"/>
    <character_restriction exclude="&lt;" reference_string="&#60;"/>
    <character_restriction exclude="é" reference_string="&#eacute;"/>
    <character_restriction exclude="é" reference_string="&#233;"/>
    <character_restriction exclude="é" reference_string="&#xE9;"/>
    <character_restriction exclude="é" reference_string="é'"/>
    <character_restriction exclude="&amp;" reference_string="&amp;#amp;"/>
    <character_restriction exclude="&amp;" reference_string="&#38;"/>
    <character_restriction exclude="&apos;" reference_string="&apos;"/>
    <character_restriction exclude="&quot;" reference_string="&quot;"/>
    <character_restriction exclude="&gt;" reference_string="&gt;"/>
  </elements>
  <attributes>
    <character_restriction exclude="&apos;" reference_string="&#39;"/>
    <character_restriction exclude="&apos;" reference_string="&#39;"/>
    <character_restriction exclude="&quot;" reference_string="&#34;"/>
    <character_restriction exclude="&quot;" reference_string="&#34;"/>
    <character_restriction exclude="&gt;" reference_string="&gt;"/>
    <character_restriction exclude="&gt;" reference_string="&#62;"/>
    <character_restriction exclude="&lt;" reference_string="&lt;"/>
    <character_restriction exclude="&lt;" reference_string="&#60;"/>
    <character_restriction exclude="&amp;" reference_string="&amp;#amp;"/>
    <character_restriction exclude="&amp;" reference_string="&#38;"/>
  </attributes>
</character_restrictions>
```

You must specify the proper encoding in the XML prolog in the configuration file. For example, if the character é is encoded in the configuration file as Latin-1 (ISO-8859-1) byte value 0xE9, the XML prolog of the configuration file must specify that the ISO-8859-1 encoding is used, instead of the default UTF-8.

Using character restrictions

Importers that utilize the configuration file build Exclude character restrictions for item types in a type tree in the same order as the configuration file. When the configuration file is processed by an importer, duplicate entries are eliminated, but the remaining entries are NOT reordered by the importer in any way.

For this reason it is important to understand how the ordering of character restrictions in the configuration file affects data processing at map execution time.

Understanding character restriction ordering

When input data is assigned to an item (A) that has Exclude character restrictions defined and item A is mapped to an item (B) on output, where item B also has Exclude character restrictions defined, the following process occurs:

- The input data is first validated for item A according to the rules specified in "[Validation of Input Data](#)", and the *internal* data is constructed.
- The internal data is passed to item B on output and the process described in "[Construction of Output Data](#)" takes place, producing the final output data.
- [Validation of input data](#)
- [Construction of output data](#)
- [Summary of the ordering process](#)

Validation of input data

When a text item within a type tree has Exclude character restrictions specified, a particular process takes place during input validation. For each character in the input data (starting from the first character), the program checks the **Reference String** column for reference strings that begin with the same character-checking the longest reference strings first. When a reference string is found, its occurrence in the input data is replaced with the paired Exclude value.

Construction of output data

When output data is being built from the output card for an item in a type tree that has Exclude character restrictions specified, the program looks for each character of data (starting from the first character) assigned to the item in the **Exclude** column. The program searches the **Exclude** column in the order that the restrictions are specified. When an Exclude character is found, it is replaced with the associated reference string, which is then moved to output.

Summary of the ordering process

- During input validation, the program looks in the **Reference String** column for the longest reference string beginning with the selected character.
- On output, the program looks for the first available match in the **Exclude** column.

Multiple character restriction sets

It is possible to specify multiple character restriction sets in a single configuration file. Each character restriction set is represented by a **<character_restrictions>** element in the configuration file. This element has the **name** attribute which identifies each character restriction set.

Another XML element in the configuration file, **<use_character_restrictions>**, specifies which character restriction set to use during the subsequent import. The value of this element should match the name of the character restriction set that you want the importer to use for the generated type trees.

If multiple **<character_restrictions>** elements are given the same **name** attribute value, and if that **name** is specified as the value for the **<use_character_restrictions>** element, the **<character_restrictions>** element which was specified first in the configuration file will be used.

If the **<use_character_restrictions>** element has a value that does not match the **name** attribute value of any **<character_restrictions>** element, the result of the import process will be the same as when an empty **<character_restrictions>** element is used.

WSDL example

If you installed the examples, you can find WSDL example files in the following directory: *install_dir\examples\web_services*.

Return codes and error messages

- [COBOL Copybook return codes and messages](#)
- [XML parsing error messages](#)

COBOL Copybook return codes and messages

If the COBOL copybook input file specified to the Importer Wizard contains incorrect or invalid COBOL copybook statements, the Importer Wizard maps will attempt to skip the erroneous statements and continue on with the next valid statement. Invalid statements are reported in the COBOL Copybook importer error and warning report.

Trying to import a file that contains a large number of errors or is not a valid COBOL copybook definition might result in the failure of the Importer Wizard to create a type tree. When this situation occurs, the maps executed by the Importer Wizard might report warning and error messages when you run the COBOL Copybook importer wizard.

The following is a list of return codes and messages along with hints or instructions for trying to resolve the problem:

Return Code	Message
N/A	Unable to rename ... An attempt might have been made to write to a read-only directory, an output file directory might be missing, or there might not be sufficient disk space on the output drive.
1	User aborted map If you cancel execution before the map completes, this message is displayed.
2	Not enough memory to execute map Close other running applications and re-try.
3	Could not open map One of the map executable files (.mmc) cannot be found. Check that the following four files are present in the install directory: UnBlock.mmc , NormalizeCopyBook.mmc , TranslatFile.mmc , BuildTTMakerScript.mmc
4	Could not read map
5	Could not read inputs
6	Invalid map handle
7	Invalid card number was specified Your map might be corrupt.
8	One or more inputs was invalid Check to make sure a proper copybook input file was specified.
9	Target not available This might be a security/authorization problem. Verify that you have authorization for read access for the input copybook file and write access for the specified output directory and the product installation directory.
10	Internal error
11	Could not build one or more outputs An attempt might have been made to write to a read-only directory, an output file directory might be missing, or there might not be sufficient disk space on the output drive.
12	Source not available An incorrect file name or path for the copybook input file might have been specified.
13	Could not open work files There might not be sufficient disk space on the drive where the install directory exists to create necessary importer work files.
15	Map must be recompiled You might have an outdated version of the importer maps versus the installed version of the Command Server. Re-install the correct version of the importer.
16	Disk write error An attempt might have been made to write to a read-only directory or file. Also, check the amount of available disk space.
17	Disk read error An attempt might have been made to access a file on a shared resource that is not accessible. Also, check the amount of available disk space.
18	Page usage count error An internal paging problem exists.
19	Internal calling error
20	Reopen file failed A map did not run correctly.
21	Input valid but unknown data found The input specified is likely not a valid copybook. Check to make sure a proper copybook input file was specified.
22	Page size too small

28	<p>Input type contains errors Invalid copybook statement(s) were found in the specified input. The COBOL Copybook Importer error report file should contain details on the problem statement(s).</p>
50	<p>Type exists and is not overwritten A duplicate definition was found in the .mts file. The COBOL Copybook Importer uses the first definition found for a given field or group name and ignores or warns about any duplicate names.</p>

XML parsing error messages

The following error messages pertain to the XML parsing that takes place when using the XML Schema Importer or XML DTD Importer.

Value	Message
-1	Input XML data is invalid.
-2	Error allocating memory.
-3	Invalid input parameters.
-4	Error initializing XML4C library.
-5	Error instantiating SAX2Parser.
-6	Error instantiating MemBufInputSource.
-7	DTD metadata location not in 'root_element DTD_URI' format.
-8	Unexpected parser error.
-9	Could not find type to assign XML data.
-10	Could not find type to assign attribute XML data.
-11	URL of the DTD against which the input instance is validated, is malformed

Type Tree Maker

Introduction

The Type Tree Maker is an application that is installed with the Design Studio. The Type Tree Maker reads-in an XML document file (.mts file extension) containing type tree commands and uses these commands to create or update a type tree.

- Example files
- Command files
- Accessing the Importer Wizard
- [Creating document files](#)
- [Example files](#)
- [Command files](#)
- [Accessing the Importer Wizard](#)

Creating document files

About this task

About this task

The document file that the Type Tree Maker uses may be produced in one of the following ways:

- Using the Type Designer to export all or part of a type tree. For more information on exporting a type tree or a portion of a type tree, see the Type Designer documentation.
- Executing an importer map (using the importer wizard) that generates a Type Tree Maker document file. For information about the importer wizard, see [Type Tree Maker Importer](#).
- Creating a map that generates a document (script) file. The document (script) file must conform to the file structure and syntax explained in the [Type Tree Maker Document File documentation](#).

Example files

The default directory `install_dir\examples\dsgnstud\ttmaker` contains example files including a type tree that contains definitions of a document file and a map that converts a data dictionary definition to a document file. These examples are explained in the [Examples](#) documentation.

Command files

The Type Tree Maker creates command files (`.mts`). A Type Tree Maker command file is represented by an acorn icon.

Accessing the Importer Wizard

About this task

The following procedure describes how to access the Importer Wizard.

To access the Importer Wizard:

Procedure

1. Open the Design Studio application.
2. Navigate to the Importer Wizard through either one of the following ways:
 - Click File > Import .
 - In the Extender Navigator view or Navigator view, right-click the location in a project in which you want to create the new type tree, and select Import. The Import window opens.
3. Expand the Transformation Extender folder, select Type Tree Maker from the import source list, and click Next.
4. Follow the Importer Wizard instructions.

Using the Type Tree Maker

- [Creating an XML document file](#)
- [Creating a Type Tree](#)
- [Type Tree Maker Importer](#)
- [Starting the Type Tree Maker from a command line](#)

Creating an XML document file

About this task

The Type Tree Maker uses document files in an XML format (.mts file extension) to create type trees. The following steps describe how to use the Type Designer to export an example type tree (xreftbl.mtt) to an XML document file (.mts).

To create an XML document file (.mts) using the Type Designer:

Procedure

1. Open a project in the Design Studio.
2. If not already imported into your work area, import the file system for the Export Type Tree Maker Example. The files are located at `install_dir\examples\dsgnstud\ttmaker\export`.
3. Open the xreftbl.mtt type tree. In the type tree editor, right-click on the root type Data and select Export from the context menu. The Export tree to file dialog opens.
4. Enter a new or accept the default file name and path and click Save. The exported .mts file is saved as Data.mts.

Results

After the export, when you execute the .mts file, the Type Tree Maker converts it into a type tree.

Creating a Type Tree

About this task

Use the Type Tree Maker to create the type tree.

To create a type tree using the Type Tree Maker:

Procedure

1. From the Start menu, click Programs > Transformation Extender > Design Studio > Type Tree Maker . The Open Type Tree Maker Command File window opens.
2. Navigate to the *install_dir\examples\dsgnstud\ttmaker\export* folder to select Xreftbl.mts. You created this file in the earlier section.
3. Click Open.
The IBM Transformation Extender Type Tree Maker generates a type tree and presents it in the Type Tree Maker window.
4. Click Close to exit the Type Tree Maker application.
The Type Tree Maker has created a type tree or modified an existing type tree, based on the information in the selected command file. The name of the type tree file is the name specified in the NEWTREE or OPENTREE command in the command file. View the type tree by opening it in the Type Designer.

Results

[Examples](#) describes the examples that are installed with the Type Tree Maker. These examples are for two uses of the Type Tree Maker and show command files and the types of commands they might contain.

Type Tree Maker Importer

The Type Tree Maker Importer is a facility that automatically generates type trees from a script (.mts file extension) that contains Type Tree Maker commands.

- [Running the Type Tree Maker Importer](#)
-

Running the Type Tree Maker Importer

About this task

This example uses the acme_tables.mtt file installed in *install_dir\examples\dsgnstud\ttmaker\import* directory. The acme_tables.mtt file was exported in the Design Studio to create the acme_tables.mts file. See [Creating an XML Document File](#) for details about creating .mts files.

To run the Type Tree Maker Importer:

Procedure

1. [Access the Importer Wizard](#) in the Design Studio application.
2. In the Import window, select Type Tree Maker from the import source list, and click Next. The Type Tree Maker Importer window opens.
3. Enter or select the name of the Type Tree Maker (.mts) file that you want to import, in the File name field, following the Importer Wizard instructions.
4. To generate the type tree, click Next. If the destination type tree file exists, you are prompted to continue or stop the import operation in the Overwrite Warning window.
After the import operation runs, it displays the run results in the Type Tree Maker Importer window. If errors or warnings, or both, occur, they appear in that same window. The errors and warnings are also saved to a log file in the product installation directory. The file name is ttmaker.log. You can open this file and analyze it in a text editor after the import operation has completed. The import operation deletes and regenerates this log file every time a new Type Tree Maker (.mts) file is imported. If errors occur, the import operation cannot generate the type tree until you resolve the problem. If only warnings occur, the import operation can still generate the type tree; however, you should inspect it to ensure that the reported warnings do not interfere with the requirements necessary for a valid instance document.
5. After the type tree generation process is complete, click Finish.
A BR tag was used here in the original source. If the type tree was generated, you are given the option to open the type tree in the Type Tree editor. The Properties view for the generated type tree contains the imported data format for the generated type tree. You can also click the type tree in the Extender Navigator view or the Navigator view to view the date and time that the import operation generated the type tree, under the Info property in the Properties view.

Starting the Type Tree Maker from a command line

When you start the Type Tree Maker from a command line, use the options listed in the following table.

The following is the syntax of **ttmakr64**:

ttmakr64 <full_path_of_.mts_file> [-A] [-B] [-H] [-N]

Command Line Option

Use

- | | |
|----|---|
| -A | Automatically analyze trees that are created or modified. Related to the ANALYZE processing instruction described in Processing instructions . |
| -B | Batch mode. Automatically closes the Type Tree Maker window when finished. Related to the AUTOCLOSE processing instruction described in Processing instructions . |
| -H | Hide the tree window. Displays a partial Type Tree Maker window for faster command file processing. Related to the HIDETREE processing instruction described in Processing instructions . |
| -N | Display no Type Tree Maker window. |

Type Tree Maker document file

The document file used by the Type Tree Maker to create a type tree is an XML-based document with a specific structure. This script file, which has the file extension **.mts**, contains commands and command elements that must be specified in a particular way and in a specific sequence.

The following file (**ttmaker60.mts**) is provided with the product, located in the **examples\dsgnstud\ttmaker** directory.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE TTMAKER SYSTEM "ttmaker60.dtd">
<?ANALYZE?><TTMAKER Version="n.n">
<NEWTREE Filename="install_dir\examples\dsgnstud\ttmaker\export\element.mtt">
<ROOT SimpleTypeName="Element" SetUpProperties="DEFAULT" SetUpComponents=
"DELETE" OrderSubtypes="ASCENDING"><Sequence partition="NO"><Implicit/>
</Sequence>
<CharTextWestern><Size Min="0" Max="S"/>
<Western CharSet="NATIVE"/>
<ValueRestrictions IgnoreCase="NO" Rule="INCLUDE"></ValueRestrictions>
</CharTextWestern>
</ROOT>
<ITEM SimpleTypeName="Acc't#Qual'rCd" CategoryOrItemParent="Element"
Description="Acct No. Qualifier Code
(896 V3050)" partition="NO" OrderSubtypes="ASCENDING"><CharTextWestern>
<Size Min="0" Max="2"/>
<PadText Justify="LEFT" ApplyPad="ANYCONTEXT"><PadValue>&lt;SP&gt;</PadValue>
<PaddedToFixedSize Length="2"/>
</PadText>
<Western CharSet="NATIVE"/>
<ValueRestrictions IgnoreCase="NO" Rule="INCLUDE"><Value Description=
"Time Deposit">01</Value>
<Value Description="Stock">05</Value>
<Value Description="Bond">06</Value>
<Value Description="Life Insurance Face Value">07</Value>
<Value Description="Retirement Acct - Vested Interest">08</Value>
<Value Description="Demand Deposit">DA</Value>
<Value Description="Return Items on Demand Deposit Acct">RD</Value>
<Value Description="Return Items on Savings Acct">RS</Value>
<Value Description="Savings">SG</Value>
<Value Description="Mutually Defined">ZZ</Value>
</ValueRestrictions>
</CharTextWestern>
</ITEM>
</NEWTREE>
</TTMAKER>
```

When you use the Type Designer to import and create a type tree from **ttmaker60.mts**, you can see in the type tree how the structure of the document file is represented.

- [File structure](#)
- [Composition of the .mts file](#)

File structure

A Type Tree Maker document (script) file begins with an XML prolog and is followed by a **TTMAKER** element.

The **Prolog** identifies the XML version and the Document Type Definition (DTD) of the document file.

The **TTMAKER** element contains a series of lines where each line in the **TTMAKER** element must be either a command or comment.

- [Commands and comments](#)
- [Prolog](#)
- [TTMAKER element](#)
- [XML tag notation](#)

Commands and comments

Commands and comments begin in the first character position of a line.

- A command line begins with a command initiator.
 - Command initiators are XML start-tags.
- A command line ends with a command terminator
 - A command terminator consists of the appropriate XML end-tag, followed by a carriage return/linefeed <NL>.
- Comments begin with the symbols <!-- and end with the symbols --><NL>.

Characters used as markup delimiters by XML may occur within elements contained in **TTMAKER**.

Prolog

The **Prolog** consists of an XML declaration, followed by a DTD declaration, followed by a series of processing instructions.

The XML declaration is either

```
<?<SP>XML<SP>version="1.0"<SP>encoding="UTF-8"?>  
or  
<?<SP>XML<SP>version="1.0"<SP>encoding="Shift_JIS"?>  
The DTD declaration is  
<!DOCTYPE TTMAKER<SP>SYSTEM<SP>"ttmaker60.dtd">  
• Processing instructions
```

Processing instructions

Processing instructions act on a tree or the Type Tree Maker window. One or more of the following processing instructions can be used:

Processing Instruction

Effect on Type Tree or Window

```
<?AUTOCLOSE?>  
Automatically closes the Type Tree Maker window after the document file has been processed. The command line option is -B.  
<?ANALYZE?>  
Performs a logic analysis on type trees referenced in the document file before saving the type tree. The command line option is -A.  
<?HIDETREE?>  
Hides the type tree while processing the document file. Very large trees can be created or modified faster if the HIDETREE instruction is used. The command line option is -H.  
<?REPLACE?>  
Replaces an existing type when a new type command is issued and the type tree already has an existing type with the same name. If not used, the existing type is not replaced and an error message is issued.
```

The processing instructions listed provide similar results to using the command line options.

TTMAKER element

The **TTMAKER** element has a start tag `<TTMAKER Version="6.0">` and end tag `</TTMAKER>`. For example:

```
<TTMAKER Version="6.0">... </TTMAKER>
```

The **TTMAKER Element** consists of one or more **NEWTREE** and **OPENTREE** elements.

The **NEWTREE** element begins with a start tag `<NEWTREE Filename="install_dir\examples\dsgnstud\ttmaker\export\ttmaker60.mtt">` and end tag `</NEWTREE>`. For example:

```
<NEWTREE Filename="install_dir\examples\dsgnstud\ttmaker\export\ttmaker60.mtt"></NEWTREE>
```

The **TTMAKER Choice Expression** consists of a choice of the **NEWTREE** and **OPENTREE** commands:

The **NEWTREE** and **OPENTREE** elements instruct the Type Tree Maker to create a new tree or open an existing one. A **Choice** expression determines the sequence of actions to be performed at a particular location in the type tree. Each **Choice** is a sequence of commands that acts on a particular type. The class of the active type determines the valid command sequence.

For example, if the active type is a group, you can add components and new **GROUP** types. If the active type is an item, you can add restrictions and new **ITEM** types.

A **Choice** expression identifies a sequence of commands. It is not an XML element, so it has no start or end tag.

XML tag notation

Each element in the document file has a start tag and an end tag. **NEWTREE** and **OPENTREE** elements contain other elements. Elements are organized as follows:

Element	Description
Command elements	Actions that initialize type trees, types, and operate on active types
Choice elements	Elements within a command that specify a choice of other elements
Sequence elements	Elements within a command that specify a sequence of other elements
Text elements	Character or integer text items

Each type of element has a start tag and an end tag. Command elements are followed by a `<NL>`. Choice, sequence, and text elements are components of a command. They are not followed by a `<NL>`.

A start tag is the name of the element enclosed in angle brackets. For example: `<CATEGORY>` is the start tag for the **CATEGORY** command.

An end tag uses the same element name and includes a slash (/) inside the left angle bracket. For example, `</CATEGORY>` is the end tag for the **CATEGORY** command.

Some elements have attributes. An attribute list, when defined, appears after the name of the element between the angle brackets of the start tag. For example:

```
<delimited location="INFIX">
```

Composition of the .mts file

This section contains descriptions for Type Tree Maker commands and command elements that make up the Type Tree Maker document (script) file (.mts).

In a Type Tree Maker document (script) file (.mts), command components must be on the same line as the command in which they are contained. In the following examples, components may appear on separate lines for documentation purposes only.

- [Commands](#)
- [Command elements](#)

Commands

Commands in the context of the .mts file are referring to those tags within level one that serve as a command. For example, **NEWTREE** is a command that tells the Type Tree Maker to create a new type tree. Each command and related component is described in this documentation.

- [AddComponent](#)
- [AddRestriction](#)
- [ASCTYPE_XML](#)
- [BinaryDatetime](#)
- [BinaryNumber](#)
- [BinaryText](#)
- [CATEGORY](#)
- [CharDateTime](#)
- [CharNumber](#)
- [CharRestrictions](#)
- [CharTextJapanese](#)
- [CharTextWestern](#)
- [Choice](#)
- [ChoiceComponent](#)
- [GROUP](#)
- [ITEM](#)
- [NEWTREE](#)
- [PROPAGATE](#)
- [Range](#)
- [Range command](#)
- [Range command element](#)
- [RangeRestrictions](#)
- [Sequence](#)
- [SequenceComponent](#)
- [SyntaxItem](#)
- [Unordered](#)
- [UnorderedComponent](#)
- [Value](#)
- [ValueRestrictions](#)

AddComponent

The **AddComponent** command adds a component to the group specified as the active type.

The component is added as the last component in the component list of the active type. If the active type is an item, the **AddComponent** command is ignored and an error message is written to the error file.

See the Type Designer documentation for information on specifying component rules.

Components of AddComponent

A group view displays the components of the **AddComponent** command.

The **AddComponent** element has as an attribute the complete type name of the type to which the component is to be added. It then has either a "[SequenceComponent](#)", "[ChoiceComponent](#)", or "[UnorderedComponent](#)" command element.

Example

```
<AddComponent CompleteTypeName="ACMEREC Table ACME">
<SequenceComponent>
<RelativeTypeName>ACMEREC Row</RelativeTypeName>
<Range Min="0" Max="S"/>
</SequenceComponent>
</AddComponent>
```

AddRestriction

The **AddRestriction** command adds a restriction to the item type specified as the active type.

Components of AddRestriction

A group view displays the components of the **AddRestriction** command.

The **AddRestriction** element has as an attribute the complete type name of the type to which the restriction is to be added. The **AddRestriction** command then has a "[ValueRestrictions](#)", "[RangeRestrictions](#)", or "[CharRestrictions](#)" command.

Example

```
<AddRestrictions CompleteTypeName="Acc't#Qual'rCd Element Data">
<ValueRestrictions>
<Value Description="Life Insurance">07</Value>
<Value Description="Retirement Acct">08</Value>
</ValueRestrictions>
</AddRestrictions>
```

ASCTYPE_XML

The **ASCTYPE_XML** command specifies that an element has XML specific attributes.

Example

```
<xs:element name="ASCTYPE_XML">
<xs:complexType>
<xs:choice>
<xs:element ref="XML_DOCUMENT"/>
<xs:element ref="XML_COMMENT"/>
<xs:element ref="XML_PI"/>
<xs:element ref="XML_PCDATA"/>
<xs:element ref="XML_ELEMENT"/>
<xs:element ref="XML_ANYELEMENT"/>
<xs:element ref="XML_ATTRIBUTE"/>
<xs:element ref="XML_ATTRIBUTELIST"/>
<xs:element ref="XML_CONTENT"/>
<!--
NOTE: The following five XML types are present in the type
tree but they never appear in the mapping. They are only
used to represent the corresponding XML Schema constructs.
They never appear in the type tree generated from a DTD
document, only from XML Schema document.
-->
<xs:element ref="XML_SIMPLETYPE"/>
<xs:element ref="XML_COMPLEXTYPE"/>
<xs:element ref="XML_ATTRIBUTEGROUP"/>
<xs:element ref="XML_GROUP"/>
<xs:element ref="XML_GLOBAL"/>
</xs:choice>
</xs:complexType>
</xs:element>
```

Each of the above declared elements corresponds with their XML Schema counterparts, providing the same level of information.

BinaryDatetime

The **BinaryDatetime** command defines a binary date-time item.

Components of BinaryDatetime

A group view displays the components of the **BinaryDatetime** command.

In addition to its Presentation attribute (**PACKED** or **BCD**) and its **BinaryDateTimeFormatString** element, the **BinaryDatetime** command can have either "[ValueRestrictions](#)" or "[RangeRestrictions](#)" command.

Example

```
<BinaryDatetime Presentation="PACKED">
<BinaryDateTimeFormatString>YYMMDD</BinaryDateTimeFormatString>
</BinaryDatetime>
```

BinaryNumber

The **BinaryNumber** command defines a binary numeric item.

Components of BinaryNumber

A group view displays the components of the **BinaryNumber** command.

The **BinaryNumber** command must have either a BCD, BinFloat, BinInt, or Packed element and either a "[ValueRestrictions](#)" or "[RangeRestrictions](#)" command.

Example

```
<BinaryNumber>
<BinInt Length="4" Sign="YES" ByteOrder="NATIVE"/>
</BinaryNumber>
```

BinaryText

The **BinaryText** command defines a binary text item.

Components of BinaryText

A group view displays the components of the **BinaryText** command.

The **BinaryText** command can have a Size, **ForNONE**, or "[ValueRestrictions](#)" element.

Example

```
<BinaryText>
<Size Min="1" Max="4"/>
<NONE RequiredOnInput="YES"><NULL></NONE>
</BinaryText>
```

CATEGORY

The **CATEGORY** command adds or replaces a category type as a subtype of the active type.

If the active type is a group or item, the **CATEGORY** command is ignored and an error message is written to the error file.

Components of CATEGORY

The following group view displays the components of the **CATEGORY** command.

The **CATEGORY** command must have at least one TypeSyntax, a Sequence, Choice, or Unordered group subclass command, and one of the item subclass command elements.

Example

```
<CATEGORY SimpleTypeName="Field" CategoryParent="Data">
<Sequence partition="NO">
<Implicit/>
</Sequence>
<CharTextWestern>
<Size Min="0" Max="S"/>
<Western CharSet="NATIVE"/>
<ValueRestrictions IgnoreCase="NO" Rule="INCLUDE"/>
</CharTextWestern>
</CATEGORY>
```

CharDateTime

The **CharDateTime** command defines a character date & time item.

Components of CharDateTime

A group view displays the components of the **CharDateTime** command.

In addition to its required CharDateTimeFormatString command element, the **CharDateTime** command can have PadText, Zero and **ForNONE** command elements, as well as either a "[ValueRestrictions](#)" or "[RangeRestrictions](#)" command.

Example

```
<CharDatetime>
<CharDateTimeFormatString>{YYMMDD}{HH24MM}</CharDateTimeFormatString>
<Western CharSet="NATIVE"/>
<NONE>*****</NONE>
<RangeRestrictions Rule="INCLUDE"/>
</CharDatetime>
```

CharNumber

The **CharNumber** command defines a character number item.

Components of CharNumber

A group view displays the components of the **CharNumber** command.

The **CharNumber** command has a required **CharNumber_1 Choice Expression**, which is either a CharInt, Decimal, or Zoned command element and can have Western, Zero, and **ForNONE** command elements and either a "[ValueRestrictions](#)" or "[RangeRestrictions](#)" command.

Example

```
<CharNumber>
<CharInt>
<TotalDigits Min="0" Max="6"/>
<IntegerFormatString>{#';'####}</IntegerFormatString>
</CharInt>
<Western CharSet="ASCII"/>
<Zero>*****</Zero>
</CharNumber>
```

CharRestrictions

The **CharRestrictions** command adds a character restriction to an item.

Components of CharRestrictions

A group view displays the components of the **CharRestrictions** command.

The **CharRestrictions** command has attributes for indicating whether or not the case of restrictions should be ignored and for specifying a syntax item for a release character and can have either a "[IncludeChar](#)" or "[ExcludeChar](#)" command element.

Example

```
<CharRestrictions IgnoreCase="NO">
<ExcludeChar>
<ExcludeValue>$</ExcludeValue>
<RefString>%DOLLAR%</RefString>
</ExcludeChar>
<ExcludeChar>
<ExcludeValue>&#amp;</ExcludeValue>
<RefString>%AMPERSAND%</RefString>
</ExcludeChar>
</CharRestrictions>
```

CharTextJapanese

The **CharTextJapanese** command defines a character text item.

Components of CharTextJapanese

A group view displays the components of the **CharTextJapanese** command.

The **CharTextJapanese** command can have "[Size](#)", "[PadText](#)", **ForNONE**, and "[Japanese](#)" command elements and a "[ValueRestrictions](#)" command.

Example

```
<CharTextJapanese>
<Size Min="0" Max="15"/>
<Japanese CharSet="NATIVE"/>
</CharTextJapanese>
```

CharTextWestern

The **CharTextWestern** command defines a character text item.

Components of CharTextWestern

A group view displays the components of the **CharTextWestern** command.

The **CharTextWestern** command can have "[Size](#)", "[PadText](#)", **ForNONE**, and "[Western](#)" command elements, as well as a "[ValueRestrictions](#)", "[RangeRestrictions](#)" or "[CharRestrictions](#)" command.

Example

```
<CharTextWestern>
<Size Min="0" Max="15"/>
```

```
<Western CharSet="NATIVE"/>
<ValueRestrictions IgnoreCase="NO" Rule="INCLUDE"/>
</CharTextWestern>
```

Choice

The **Choice** command defines a character text item.

Components of choice

A group view displays the components of the **Choice** command.

The **Choice** command can have a [WhiteSpace](#), [FloatingComponent](#) and [ChoiceComponent](#) command.

ChoiceComponent

The **ChoiceComponent** command defines a character text item.

Components of ChoiceComponent

A group view displays the components of the **ChoiceComponent** command.

The **ChoiceComponent** command can have **RelativeTypeName** and **Rule** command elements.

Component Description

RelativeTypeName: Specifies the relative type name of the syntax item used as the separator. A relative type name is a series of simple type names separated by spaces.

Rule: Specifies the component rule (optional) associated with the component. A rule can be up to 32K bytes.

GROUP

The **GROUP** command adds a new group as a subtype of the active type.

If the active type is an item, this command is ignored and an error message is written to the error file.

Components of GROUP

A group view displays the components of the command.

The **GROUP** command can have "[TypeSyntax](#)", a "[Sequence](#)", "[Choice](#)", or "[Unordered](#)" group subclass command.

Example

```
<GROUP>
<SimpleTypeName>MyGroup</SimpleTypeName>
<setUpProperties>INHERIT </setUpProperties>
<terminator>
<WLiteral><NL></WLiteral>
</terminator>
</GROUP>
```

ITEM

The **ITEM** command adds a new item to the active type.

If the active type is a group, this command is ignored and an error message is written to the error file.

Components of ITEM

A group view displays the components of the **ITEM** command.

The **ITEM** command can have "[CharTextWestern](#)", "[CharTextJapanese](#)", "[CharNumber](#)", "[CharDateTime](#)", "[BinaryText](#)", "[BinaryDatetime](#)", "[BinaryNumber](#)", and "[SyntaxItem](#)" commands.

Example

```
<ITEM>
<SimpleTypeName>MyItem</SimpleTypeName>
<iprop_char_text>
<size>0</Min><Max>S</Max></size>
</iprop_char_text>
</ITEM>
```

NEWTREE

The **NEWTREE** command creates a new type tree.

Components of NEWTREE

A group view displays the components of the **NEWTREE** command.

The **NEWTREE** command can have a **ROOT** command element and an **OPENTREE** command.

Example

```
<NEWTREE>
<FileName>install_dir\mytree.mtt</FileName>
<ROOT>
<SimpleTypeName>Sales</SimpleTypeName>
<Description>Tree for Sales</Description>
</ROOT>
</NEWTREE>
```

PROPAGATE

The **PROPAGATE** command allows propagation and deletion of subtypes of the active type.

Keyword Description

INITIATOR: Propagates all initiator properties of the active type to all types in the sub-tree of the active type.

TERMINATOR: Propagates all terminator properties of the active type to all types in the sub-tree of the active type.

RELEASE: Propagates all release properties of the active type to all types in the sub-tree of the active type.

PARTITION: Propagates the partition property of the active type to all group or item types in the sub-tree of the active type.

DESCRIPTION: Propagates the description property of the active type to all types in the sub-tree of the active type.

GROUPFORMAT: Propagates all group properties of the active type to all group and/or category types in the sub-tree of the active type. This keyword is ignored for items.

ITEMFORMAT: Propagates all item properties of the active type to all item and/or category types in the sub-tree of the active type. This keyword is ignored for groups.

DELETESUBTYPES: Deletes subtypes of the active type.

Example:

```
<PROPAGATE>TERMINATOR</PROPAGATE>
```

Range

Range can be a command and a command element.

Range command

The **Range** command defines a range restriction for an item. For example, <Range>x, y

Range command element

The **Range** command element defines a range for a component. For example, [<Range>x, y]

Components of Range

A group view displays the components of the **Range** command/command element.

Component Description

Min: Specifies the integer value of the minimum range.

Max: Specifies the value of the maximum range: either an integer or the value (**S**) for an unspecified maximum range.

Example

```
<Range>
<Min>2</Min>
```

```
<Max>S</Max>
</Range>
```

RangeRestrictions

The **RangeRestrictions** command specifies that an item will have range restrictions.

Components of RangeRestrictions

A group view displays the components of the **RangeRestrictions** command.

The **RangeRestrictions** command can have "[ValueRestrictions](#)" and "[Range](#)" commands.

Sequence

The **Sequence** command defines a group as a sequence group.

Components of Sequence

A group view displays the components of the **Sequence** command.

The **Sequence** command can have an Implicit or Explicit command element and a **SequenceComponent** command.

SequenceComponent

The **SequenceComponent** command defines a component as a sequence group.

Components of SequenceComponent

A group view displays the components of the **SequenceComponent** command.

The **SequenceComponent** command can have a **RelativeTypeName**, Range, or **Rule** command.

Component Description

RelativeTypeName: Specifies the relative type name of the syntax item used as the separator. A relative type name is a series of simple type names separated by spaces.

Rule: Specifies the component rule (optional) associated with the component. A rule can be up to 32K bytes.

SyntaxItem

The **SyntaxItem** command defines a syntax item.

Components of SyntaxItem

A group view displays the components of the **SyntaxItem** command.

The **SyntaxItem** command can have a Size, Western, or Japanese command element and a **ValueRestriction** command.

Unordered

The **Unordered** command specifies a value restriction for an item.

Components of Unordered

A group view displays the components of the **Unordered** command.

The **Unordered** command can have a "[Delimited](#)" or "[WhiteSpace](#)" command elements and either a **FloatingComponent** or "UnorderedComponent" command.

UnorderedComponent

The **UnorderedComponent** command defines a component for an unordered group.

Components of UnorderedComponent

A group view displays the components of the **UnorderedComponent** command.

The **UnorderedComponent** command can have a **RelativeTypeName**, Range, or Rule command.

Component Description

RelativeTypeName: Specifies the relative type name of the syntax item used as the separator. A relative type name is a series of simple type names separated by spaces.

Rule: Specifies the component rule (optional) associated with the component. A rule can be up to 32K bytes.

Value

The **Value** command specifies a value restriction for an item.

Components of Value

A group view displays the components of the **Value** command.

ValueRestrictions

The **ValueRestrictions** command specifies that an item will have value restrictions.

Components of ValueRestrictions

A group view displays the components of the **ValueRestrictions** command.

The **ValueRestrictions** command can have an **IgnoreCase** or **Rule** attribute and a **Value** command.

IgnoreCase

Yes: Not case sensitive

No: Case sensitive

Rule: Specifies the component rule (optional) associated with the component. A rule can be up to 32K bytes.

S

Command elements

This documentation contains descriptions of command elements. Command elements consist of other elements and expressions. Elements have XML start and end tags. Expressions are implied groups within a command that are not enclosed in XML tags.

- [BCD](#)
- [BinFloat](#)
- [BinInt](#)
- [CharInt](#)
- [Decimal](#)
- [Decimal format Strings](#)
- [Leading-Sign](#)
- [Whole#](#)
- [Fraction](#)
- [Trailing-Sign](#)
- [Sign Sub-string formats](#)
- [Delimited](#)
- [Components of delimited](#)
- [DelimiterLiteral](#)
- [empty](#)
- [ExcludeChar](#)
- [ExcludeRange](#)
- [ExcludeValue](#)
- [Explicit](#)
- [FractionItem](#)
- [Implicit](#)
- [IncludeChar](#)
- [INITIATOR](#)
- [Japanese](#)
- [Literal](#)
- [Modify](#)
- [OneByteLiteral](#)
- [Packed](#)

- [PaddedToFixedSize](#)
 - [PadNumber](#)
 - [PadText](#)
 - [RELEASE](#)
 - [ROOT](#)
 - [Size](#)
 - [TERMINATOR](#)
 - [ThousandsItem](#)
 - [TotalDigits](#)
 - [TTMAKER](#)
 - [TypeSyntax](#)
 - [Variable](#)
 - [Western](#)
 - [WhiteSpace](#)
 - [Zero](#)
 - [Zoned](#)
-

BCD

The **BCD** command element specifies BCD properties for an item type.

Components of BCD

A group view displays the components of the **BCD** command element.

The **BCD** command element can have a **Size** command element.

Example:

```
<BCD>
<size>
<Min></Min>
<Max>10</Max>
</size>
</BCD>
```

BinFloat

The **BinFloat** command element specifies binary float properties for a item type.

Components of BinFloat

A group view displays the components of the **BinFloat** command element.

The **BinFloat** command element can have a **Length** attribute.

Component Name	Attributes	Description
Length	1, 2, 4	Specifies a length restricted to 1, 2, or 4 bytes.

Example

```
<BinFloat>
<Length>10</Length>
</BinFloat>
```

BinInt

The **BinInt** command element specifies binary integer properties for an item type.

Components of BinInt

A group view displays the components of the **BinInt** command element.

The **BinInt** command element can have a **Length**, **Sign**, or **ByteOrder** attribute.

Component Name	Attributes	Description
Length	1, 2, 4	Specifies a length restricted to 1, 2, or 4 bytes.
Sign	Yes	Default setting. Specifies the sign attribute value of the number.
	No	Specifies the sign attribute value of the number.
ByteOrder	<ul style="list-style-type: none"> BIGENDIAN LITTLEENDIAN NATIVE 	Specifies the byte order attribute value of the number.

Example

```
<BinInt>
<Length>10</Length>
<ByteOrder>BIGENDIAN</ByteOrder>
</BinInt>
```

CharInt

The **CharInt** command element specifies character integer number properties.

Components of CharInt

A group view displays the components of the **CharInt** command element.

The **CharInt** command element can have "[TotalDigits](#)" , "[PadNumber](#)" , **IntegerFormatString** and "[ThousandsItem](#)" command elements.

IntegerFormatString: Specifies the syntax of the character integer item.

Integer format strings

The form of an integer format string is similar to that for decimals, except that the Fraction sub-string is not included:

```
"{" + Leading-sign(1) + Whole# + Trailing-sign(1) + "}"
```

Example

```
{L+}
{L-`(`###T-`)}'
```

Decimal

The **Decimal** command element specifies character decimal number properties.

Components of Decimal

A group view displays the components of the **Decimal** command element.

The **Decimal** command element can have **TotalDigits**, **PadNumber**, **DecimalFormatString**, **ThousandsItem**, and **FractionItem** command elements.

DecimalFormatString: Specifies the syntax of the character decimal item.

Decimal format Strings

A decimal format string is enclosed in braces { }. It has an optional "[Leading-Sign](#)" , a required Whole#, a required Fraction, and an optional Trailing-Sign. The following is an example of a decimal format string:

```
{" + Leading-Sign(1) +Whole# + Fraction + Trailing-Sign(1) + "}"
```

Leading-Sign

The following table describes the optional **Leading-Sign** format string.

Sub-string Values Description:

"L" + Positive + Negative + Zero(1)

A leading sign is required for positive numbers and negative numbers.

"L"+ "[" + Positive + "]" + Negative + Zero(1)

A leading sign is optional for positive numbers and required for negative numbers.

"L" + Positive + "[" + Negative + "]" + Zero(1)

A leading sign is required for positive numbers and optional for negative numbers.

"L" + Positive + Zero(1)

A leading sign is required for positive numbers and there is no leading sign for negative numbers.

"L" + Negative + Zero(1)

A leading sign is required for negative numbers and there is no leading sign for positive numbers.

Example

```
{L+}
```

Whole#

The following table describes the required **Whole#** format string.

Sub-string Values	Description
min-digits(1) + "#" + value(1) + "###" + max-digits(1) -or- min-digits(1) + "#" + "[" + value(1) + "]" + "###" + max-digits(1)	Specifies the thousands separator and the range of whole number digits. If min-digits is not used, the default is zero. If max-digits is not used, the default is S . If a ThousandsItem is specified, a Value must be present as the default for the syntax item.
"V" + implied-places	Specifies the decimal to be implicit and implied places specifies where the intended decimal separator is to be placed.
value + min-digits(1) + "##" + max-digits(1)	Specifies the value of the decimal separator to be required and the range of fraction digits. If min-digits is not used, default is zero. If max-digits is not used, default is S .
"[" + value + min-digits(1) + "##" + max-digits(1) + "]"	Specifies the value of the decimal separator to be optional if there is no fractional portion of the number. It also specifies the range of fraction digits. If min-digits is not used, default is zero. If max-digits is not used, default is S .

Examples

####[`.'##] is interpreted as:

- There is no thousands separator.
- The decimal separator has the value of a period (.) .
- The decimal separator and fraction are optional.
- There are no restrictions on the whole number digits, or on fractional digits.

Value Description:

L-#####V2

An implicit decimal, implied to two places, with a required leading minus sign for negative numbers.

L-#####[`.##]

ANSI decimal.

L-1####[,1##]

EDIFACT decimal.

L-'N'####'##

SWIFT decimal.

Fraction

The following table describes the required **Fraction** format string.

Sub-string Values Description

"V" + implied-places

Specifies the decimal to be implicit and implied places specifies where the intended decimal separator is to be placed.

value + min-digits(1) + "##" + max-digits(1)

Specifies the value of the decimal separator to be required and the range of fraction digits. If min-digits is not used, default is zero. If max-digits is not used, default is **S**.

"[" + value + min-digits(1) + "##" + max-digits(1) + "]"

Specifies the value of the decimal separator to be optional if there is no fractional portion of the number. It also specifies the range of fraction digits. If min-digits is not used, default is zero. If max-digits is not used, default is **S**.

Trailing-Sign

The following table describes the optional **Trailing-Sign** format string.

Sub-string Values Description

"T" + Positive + Negative + Zero(1)

A trailing sign is required for positive numbers and negative numbers.

"T" + "[" + Positive + "]" + Negative + Zero(1)

A trailing sign is optional for positive numbers and required for negative numbers.

"T" + Positive + "[" + Negative + "]" + Zero(1)

A trailing sign is required for positive numbers and optional for negative numbers.

"T" + Positive + Zero(1)

A trailing sign is required for positive numbers and there is no trailing sign for negative numbers.

"T" + Negative + Zero(1)

A trailing sign is required for negative numbers and there is no trailing sign for positive numbers.

Sign Sub-string formats

The following table describes the sub-strings:

Sub-string name	Sub-string Values	Meaning
Positive	"+" + value(1)	
Negative	"-" + value(1)	
Value	A text-string enclosed in single-quotes. Value has a release character of "/" if the text uses a single-quote or "/" in the format string.	The sign value of the previous sign-indicator. If Value is not used, the default is "+" for positive numbers and "-" for negative numbers.
Zero	"Z" + value	Specifies the required leading sign value if the number is zero. If Zero is not used, there is no sign associated with a zero.
	"[" + "Z" + value + "]"	Specifies the optional leading sign value if the number is zero. If Zero is not used, there is no sign associated with a zero.

Delimited

The **Delimited** command element is a sequence element that specifies delimited properties of a group type.

Delimited consists of a location attribute and a **Delimiter Choice**.

Components of delimited

A group view displays the components of the Delimited command element.

The **Delimited** command element can have a **location** attribute, and "[DelimiterLiteral](#)" and "[Variable](#)" command elements.

Location attribute value:

PREFIX: Specifies before each component.

INFIX: Default setting. Specifies between each component.

POSTFIX: Specifies after each component.

Examples

```
<delimited location="PREFIX">
<Variable>
<RelativeTypeName>Record Separator</RelativeTypeName>
<DefaultSyntaxValue><CR></DefaultSyntaxValue>
</Variable>
<delimited>
<delimited>
<WLiteral WesternCharSet="ASCII">
<WesternLiteral>#</ WesternLiteral >
</WLliteral >
</delimited>
```

DelimiterLiteral

The **DelimiterLiteral** command element specifies the properties of a western literal.

Components of DelimiterLiteral

A group view displays the components of the DelimiterLiteral command element.

The **DelimiterLiteral** command element can have Western and Japanese and command elements.

empty

A group view displays the components of the **empty** command element.

The **empty** command element can have **IgnoreCase** or **Required** attributes, Western or Japanese commands and a **LiteralValue** command.

IgnoreCase

Yes: Not case sensitive.

No: Case sensitive.

Required

OnInput: Only required on input.

OnOutput: Only required on output.

Always: Always required.

Never: Never required.

ExcludeChar

A group window displays the components of the **ExcludeChar** command element.

The **ExcludeChar** command element can have an **ExcludeValue** or **RefString** command element.

ExcludeRange

The **ExcludeRange** command element defines the range restriction to be excluded for an item.

Components of ExcludeRange

A group view displays the components of the **ExcludeRange** command element.

The **ExcludeRange** command element can have a **Min** or **Max** command elements.

ExcludeValue

The **ExcludeValue** command element defines the values to be excluded for an item.

Components of ExcludeValue

A group view displays the components of the **ExcludeValue** command element.

The **ExcludeValue** command element can have a **Description** attribute. Enter the values to be excluded from the restriction list in the item view.

Explicit

The **Explicit** command element specifies the attributes and syntax for an explicit group.

Components of Explicit

A group view displays the components of the command element.

The **Explicit** command element can have a **Track** attribute and a Delimited command.

Track

Content
 Track only content.

Places
 Track existing components even if they have only a placeholder.

Example

```
<Explicit>
<delimited>
<WLiteral WesternCharSet="ASCII">
<WesternLiteral>#</ WesternLiteral >
</WLliteral >
</delimited>
</Explicit>
```

FractionItem

A group window displays the components of the **FractionItem** command element.

The **FractionItem** command element can have a **Find** attribute and **RelativeTypeName** command element.

Find

- **Yes:** Specifies to find the item separator.
- **No:** Default setting. Specifies not to find the item separator.

RelativeTypeName: Specifies the relative type name of the syntax item used as the separator. A relative type name is a series of simple type names separated by spaces.

Example

```
<FractionItem find="YES">
<RelativeTypeName>My Tag</RelativeTypeName>
</FractionItem>
```

Implicit

The **Implicit** command element specifies the attributes and syntax for an implicit group.

Components of Implicit

A group view displays the components of the **Implicit** command element.

The **Implicit** command element can have a **Delimited** or **WhiteSpace** command and a **FloatingComponent** command.

IncludeChar

A group window displays the components of the **IncludeChar** command element.

The **IncludeChar** command element can have an **IncludeFirst** or **IncludeAfter** command element.

INITIATOR

The **INITIATOR** command element properties of the initiator of a type.

Components of INITIATOR

A group view displays the components of the **INITIATOR** command element.

The **INITIATOR** command element can have a **Literal** or **Variable** command element.

Example

```
<initiator>
<WLiteral>
<WesternLiteral>:35B</WesternLiteral>:
</WLliteral>
</initiator>
```

Japanese

The **Japanese** command element specifies the Japanese character set.

Components of Japanese

A group view displays the components of the **Japanese** command element.

The following data languages can be defined in the **Japanese** component.

Literal

A group view displays the components of the **Literal** command element.

The **Literal** command element can have an **IgnoreCase** attribute, a Western or Japanese command element, and a **LiteralValue** command.

IgnoreCase

- **Yes:** Not case-sensitive.
- **No:** Case-sensitive.

Modify

A group view displays the components of the **Modify** command element.

The **Modify** command element can have a **PROPAGATE**, **AddComponent**, or **AddRestriction** command.

OneByteLiteral

A group window displays the components of the **OneByteLiteral** command element.

The **OneByteLiteral** command element can have a **LiteralValue** or **Western** command element.

Packed

The **Packed** command element specifies binary packed properties for an item type.

Components of Packed

A group view displays the components of the **Packed** command element.

The **Packed** command element can have a **Length**, **ImplicitDecimalPlaces**, or **TrailingSign** attribute.

Length: Specifies the length of the packed number.

ImplicitDecimalPlaces: Specifies the implicit decimal places of the packed number.

TrailingSign

- **Yes:** Specifies the presence of a Components sign.
- **No:** Specifies the absence of a Components sign.

Example

```
<Packed>
<Length>10</Length>
<ImplicitDecimalPlaces>2</ImplicitDecimalPlaces>
</Packed>
```

PaddedToFixedSize

A group view displays the components of the **PaddedToFixedSize** command element.

PadNumber

The **PadNumber** command element specifies the one-byte pad character for number text items.

Components of PadNumber

A group view displays the components of the **PadNumber** command element.

The **PadNumber** command element can have **Justify**, **ApplyPad**, or **Fill** attributes, and either a **PadValue** or "**PaddedToFixedSize**" command.

Justify:

- **LEFT:** Default setting. Specifies left justification.
- **RIGHT:** Specifies right justification.

ApplyPad:

- **ANYCONTEXT:** Specifies to pad the number text in any context.
- **FIXEDGROUP:** Default setting. Specifies to pad the number text only in a fixed context.

Fill:

- **BEFORE:** Specifies to pad the number before the sign.
- **AFTER:** Default setting. Specifies to pad the number after the sign.

PadValue: Specifies the value of the one-byte pad character.

Example

```
<PadNumber fill="AFTER">
<PadValue><></PadValue>
<PaddedToFixedSize Length="50"/>
</PadNumber>
```

PadText

The **PadText** command element specifies the one-byte pad character for character text items.

Components of PadText

A group view displays the components of the **PadText** command element.

The **PadText** command element can have a **Justify** or **ApplyPad** attributes and a **PadValue** or "[PaddedToFixedSize](#)" command element.

Justify:

- **LEFT**: Default setting. Specifies left justification.
- **RIGHT**: Specifies right justification.

ApplyPad:

- **ANYCONTEXT**: Specifies to pad the text in any context.
- **FIXEDGROUP**: Default setting. Specifies to pad the text only in a fixed context.

PadValue: Specifies the value of the one-byte pad character.

Example

```
<PadText>
<PadValue><SP></PadValue>
<PaddedToMinContent/>
</PadText>
```

RELEASE

The **RELEASE** command element specifies the release character for the active type.

Components of RELEASE

A group view displays the components of the **RELEASE** command element.

The **RELEASE** command element can have a "[OneByteLiteral](#)" or "[Variable](#)" command element.

Example

```
<Release>
<ReleaseLiteral>
<OneByte>?</OneByte>
</ReleaseLiteral>
</Release>
```

ROOT

The **ROOT** command element specifies the properties for the root of the type tree.

Components of ROOT

A group view displays the components of the **ROOT** command element.

The **ROOT** command element can have a **SimpleTypeName**, **Description**, **SetUpProperties**, **SetUpComponents**, or **OrderSubtypes** attributes. The **ROOT** command element can also have a Sequence, Choice, or Unordered group subclass command and one of the item subclass command elements.

Component	Description
SimpleTypeName	Specifies the simple type name of the category added or replaced.
Description	Describes the category added or replaced.
SetUpProperties	INHERIT : If a property is not used, property is inherited. DEFAULT : Default setting. If a property is not used, properties that apply to the root of a newly created type tree are used.
SetUpComponents	INHERIT : Inherit all components. DEFAULT : Default setting. Do not inherit components.
OrderSubTypes	ASCENDING : Default setting. Specifies that subtypes of the category replaced or added in ascending order. DESCENDING : Specifies that subtypes of the category replaced or added in descending order. ADDFIRST : Specifies that subtypes of the category replaced or added to the beginning of the list. ADDLAST : Specifies that subtypes of the category replaced or added to the end of the list.

Example

```
<Root>
<Description>Sales Automation </ Description>
</Root>
```

Size

The **Size** command element specifies the size properties of an item.

Components of Size

A group view displays the components of the **Size** command element.

The **Size** command element can have a **Min** or **Max** attributes.

Component Description:

- **Min**: Specifies the integer value of the minimum size.
- **Max**: Specifies the value of the maximum size: either an integer or the value (**S**) for an unspecified maximum size.

Example

```
<size>
<Min>2</Min>
<Max>S</Max>
</size>
```

TERMINATOR

The **TERMINATOR** command element specifies the terminator of the active type.

Components of TERMINATOR

A group view displays the components of the **TERMINATOR** command element.

The **TERMINATOR** command element can have a Literal or Variable command elements.

Example

```
<terminator>
<Literal>
<WesternLiteral><NL></WesternLiteral>:
</Literal>
</terminator>
```

ThousandsItem

The **ThousandsItem** command element specifies the thousands separator of a character decimal number or a character integer as an item.

Components of ThousandsItem

A group view displays the components of the **ThousandsItem** command element.

The **ThousandsItem** command element can have a **Find** attribute and a **RelativeTypeName** command element.

Component Name	Description
Find	Yes : Specifies to find the item separator. No : Default setting. Specifies not to find the item separator.
RelativeTypeName	Specifies the relative type name of the syntax item used as the separator. A relative type name is a series of simple type names separated by spaces.

Example

```
<ThousandsItem>
<RelativeTypeName>My Decimal</RelativeTypeName>
</ThousandsItem>
```

TotalDigits

The **TotalDigits** command element specifies the digits properties for a character number.

Components of TotalDigits

A group view displays the components of the **TotalDigits** command element.

The **TotalDigits** command element can have a **Min** or **Max** attributes.

Component Name	Attributes	Description
TotalDigits	Min Max	Specifies the minimum and maximum digits of the character decimal number. If not used, the default is a minimum of 0, maximum of unspecified (S).

Example

```
<TotalDigits>
<Min>2</Min>
<Max>15</Max>
</TotalDigits>
```

TTMAKER

A group view displays the components of the **TTMAKER** command element.

The **TTMAKER** command element can have a **Version** attribute and a [NEWTREE](#) or [OPENTREE](#) commands.

TypeSyntax

A group view displays the components of the command element.

The **TypeSyntax** command element can have an [INITIATOR](#), [TERMINATOR](#), [RELEASE](#), or [empty](#) command elements.

Variable

The **Variable** command element specifies properties of a variable syntax object.

Components of Variable

A group view displays the components of the **Variable** command element.

The **Variable** command element can have a **Find** attribute and a **RelativeTypeName** or **DefaultSyntaxValue** command elements.

Component	Description
Find	YES: Specifies to find the value of the variable delimiter. NO: Default setting. Specifies not to find the item delimiter. Use the last value found or the default value of the variable delimiter.
RelativeTypeName	Specifies the relative type name of the syntax item used as the separator. A relative type name is a series of simple type names separated by spaces.
DefaultSyntaxValue	Specifies the literal used as a default whenever a value for the syntax item has not been set in the data. If unprintable, enter as symbolic values. If entering a symbolic value in a map rule to be interpreted by the Type Tree Maker, separate the first character from the rest. For example, "<" + "CR>" or "<CR>"

Example

```
<Variable find="YES">
<RelativeTypeName>My Tag</RelativeTypeName>
<DefaultSyntaxValue>:35B</DefaultSyntaxValue>
</Variable>
```

Western

The **Western** command element specifies western character set properties.

Components of western

A group view displays the components of the **Western** command element.

The following data languages can be defined as a character set of a **Western** literal:

WhiteSpace

A group view displays the components of the **WhiteSpace** command element.

The **WhiteSpace** command element can have a **BuildAs** attribute and a Western command element.

Component Description:

BuildAs: Enter the characters that will replace white spaces.

Zero

The **Zero** command element specifies a special value for zero.

Components of Zero

A group view displays the components of the **Zero** command element.

The **Zero** command element can have a **RequiredOnInput** attribute.

Attribute Description

RequiredOnInput: The special value is required on input.

Zoned

The **Zoned** command element specifies character zoned number properties.

Components of zoned

A group view displays the components of the **Zoned** command element.

The **Zoned** command element can have an **ImplicitDecimalPlaces** or **Sign** attribute and either a **TotalDigits** or **PadText** command element.

Component	Description
ImplicitDecimalPlaces	Specifies the number of implied decimal places.
Sign	Yes: Indicates the zoned number has a sign. No: Indicates the zoned number does not have a sign.

Example

```
<zoned>
<ImplicitDecimalPlaces>0</ImplicitDecimalPlaces>
</zoned>
```

Type Tree Maker examples

Type Tree Maker examples are provided with the Design Studio. These examples show how to use the Type Tree Maker and Type Tree Maker document files.

When you install the Design Studio, two example folders are created. One example folder is created at: *install_dir \examples\dsgnstud\ttmaker*.

The ttmaker directory contains the following folders:

Folder Description:

export: Exporting item restrictions to create a cross-reference file.

import: Importing metadata to create a type tree.

The second examples folder is created at: *install_dir \ttmaker*.

The ttmaker directory at this location contains the following folder:

dtd: The document type definition (DTD) defining the structure of the Type Tree Maker document file.

- [Export example](#)
- [Import example](#)
- [DTD example](#)

Export example

The export folder contains files that can be used to export an item's restriction list and then process the resulting document file to create a cross-reference file. [Generating a Cross-Reference Table from a Restriction List](#) describes this procedure in detail. The following briefly describes the procedure:

- In the Type Designer, select the type and use the Export command to create an **.mts** file for that type.
 - A map to create a cross-reference table uses the **.mts** file exported from the Type Designer. Each row in the resulting output contains two items from the restriction list: the restriction and its description.
 - The file can be used as a lookup file for some other mapping task.
 - [Installed export example files](#)
 - [Files created](#)
 - [Generating a cross-reference table from a restriction list](#)
-

Installed export example files

The following files are installed in the export folder:

Table 1.

File	Use
export.mms	Map file that uses the .mts file as input to create the xref.txt file.
element.mtt	Type tree that contains the item with the restriction list.
xreftbl.mtt	Type tree that defines the cross-reference table to be produced.
readme.txt	Readme file that briefly describes the Export example.
ttmaker60.mtt	Type tree that contains the types that define the structure and syntax of the Type Tree Maker document file.

Files created

When you work through this example, the following files are created:

Table 1.

File	How the File is Created
element.mts	Type Tree Maker command file created by exporting the sub-tree from the Element type tree.
xref.txt	The cross-reference table created by running the map.

Generating a cross-reference table from a restriction list

About this task

The following section explains how a Type Tree Maker document file containing a restriction list is mapped to a cross-reference table.

To generate a cross-reference table from a restriction list:

1. Open a project in the Extender Studio.
2. If not already imported into your work area, import the file system for the Export Type Tree Maker Example. The files are located at *install_dir\examples\dsgnstud\ttmaker\export*.
3. Open the **element.mtt** type tree.
4. Double-click the item type **Acc't#Qual'rCd**.
5. The item view opens. The restriction list appears in the **Include** column.
6. Close the item view when you finish looking at the restrictions.
7. To export the subtree with this restriction list, right-click **Acc't#Qual'rCd** and select Export from the context menu. The Export tree to file dialog opens.
8. Enter a new or accept the default file name and path and click Save. The exported **.mts** file is saved. This file contains commands with information that will be used to create the cross-reference file.
9. Open the example map source **export.mms**.
This map source file contains two maps: an executable map **exprstrc** and a functional map **It'sARow**.
10. Build and run the **exprstrc** map.
11. View run results.

Results

The cross-reference file now contains the necessary information from the item restriction list.

Import example

The import folder is installed in the *install_dir\examples\dsgnstud\ttmaker* folder. It contains files that can be used to create a Type Tree Maker document file from a file that contains metadata. Metadata is data about data; it defines the syntax, structure, and semantic rules of data. A type tree is a graphical representation of metadata. The document file that is used by the Type Tree Maker is metadata in a textual form.

The type tree that defines the document file is a schema. It defines data about metadata. The type tree that defines the input metadata is a schema too; it defines the metadata used to generate the document file.

A map whose purpose is to update rows in a database table with data from another data file might then use the resulting type tree.

Briefly, this procedure consists of the following:

- Define the map that generates a Type Tree Maker document file from the input metadata.
 - The Type Tree Maker generates a type tree from the document file. The type tree defines several tables, their data fields and rows.
 - This type tree could then be used by another map (which is not supplied as part of this example). The map could read or update rows in database tables with the contents of some other input.
- [Installed import example files](#)
 - [Files created](#)
 - [MyImport.mtt type tree](#)
 - [Mapping from metadata to a TTMAKER document](#)

Installed import example files

The following files are installed in the import folder:

Table 1.

File	Use
createtable.mms	Map file that contains the executable map TableImport, which generates the output file acmetbls.mts . This .mts file is the input to the Type Tree Maker. This is described in the section, " Files Created ".
acmemeta.mtt	Type tree that defines the schema for the table, row, and field types to be created. This tree contains the type MetaFile, which defines the input to the map.
myimport.mtt	Type tree that contains the definition of the Type Tree Maker document file to be created.
acmedef.txt	Data for the MetaFile input.
readme.txt	Readme file that briefly describes the Import example.

Files created

The following files are created when working through this example:

Table 1.

Files Created	How the File is Created
acme_tables.mts	Type Tree Maker document file created by running the map.
acme_tables.mtt	Type tree generated by the Type Tree Maker from the acme_tables.mts file.

MyImport.mtt type tree

The **myImport.mtt** is a customized version of **ttmaker60.mtt**. The **ttmaker60.mtt** type tree is generic; it can be used for any Type Tree Maker document file. A customized version helps to visualize the solution to a specific problem. In **ttmaker60.mtt**, a type tree consists of either a NEWTREE or OPENTREE element, which in turn consists of a series of **OPENTREE Choice Expression(s)**.

The **myImport.mtt** type tree replaces the generic **OPENTREE Choice Expression** with specific components for constructing a specific type tree. The **ttmaker60.mtt** type tree was made more specific for this example, as follows:

ttmaker60.mtt was saved as **myImport.mtt**. The layout of the type tree to be created was designed.

Specific types were added as subtypes of the ITEM, GROUP, and CATEGORY Element types under the root **XML**.

Each specific type was modeled after its parent type. For example, the FieldType ITEM Element is identical to the generic ITEM Element.

After all the subtypes were made, the components of NEWTREE were changed to incorporate the specific types for this example.

Now, you are ready to make the map using the modified MyImport type tree.

Mapping from metadata to a TTMAKER document

In the **createtable.mms** source file, **tblimp** is the executable map. It takes as its input the **Acme** metadata file and produces a type tree command file that will create a new type tree file defining this data. The TableImport map generates the **.mts** file.

The input card for the map uses the definitions of the metadata contained in the **Acmemeta.mtt** type tree file and uses the actual metadata contained in **ACMEDEF.TXT**:

The **acmemeta.mtt** type tree describes the **Acme** metadata:

The type of the output card for the map is Document XML from the myimport.mtt type tree.

The **TableImport** executable map contains rules that create the **.mts** file from the input metadata.

After you build and run the map, view the Run Results.

When you run the Type Tree Maker with the **acme_tables.mts** file as input, the Type Tree Maker window shows the type tree being built. When the Type Tree Maker completes, the Type Tree Maker window and a completion message is displayed.

The **acme_tables.mtt** type tree contains types for all the fields, rows, and tables defined in **acmedef.txt**.

This type tree can then be used to describe the input or output data for some other map.

DTD example

The following **.dtd** file is installed in the *install_dir\ttmaker\dtd* folder:

Table 1.

File	Use
ttmaker60.dtd	The document type definition (DTD) defining the structure for the Type Tree Maker document file.

Return codes and error messages

When the Type Tree Maker finds a problem with a command, it returns either an error message or a warning message, depending on the severity of the problem. These error and warning messages, viewable from the Type Tree Maker, are also stored in the *install_dir\ttmaker.log* file.

The **ttmaker.log** file record contains the line number of the command that caused the error or warning message, followed by a message. An example of a warning message is Warning 34 at line 3: Overwrote existing type (and possible subtree).

- [Type Tree generation error messages](#)
- [Type tree generation logic warning messages](#)
- [Errors and warnings](#)

Type Tree generation error messages

The error and warning messages are listed in the following tables. The hints show some likely causes for the error message.

A general hint is to always check the spelling and syntax of the commands.

Table 1.

Message	Description
Command line too long	Correct the command line. The maximum size of a command line is 4096 bytes.
Invalid syntax in command line	Correct the command.
Cannot create type tree	A directory may not exist. All directory names in the path must conform to 8.3 rules: a maximum of eight characters for file name and a three-character extension.
Cannot set root type	The definition of the root type is incorrect.
Cannot set description	The type was not created properly.
Cannot set current type (Type does not exist.)	The type was not created properly.
Cannot create type (already exists?)	The type cannot fit under the active type (for example: a group cannot be placed under an item), the active type name is too long, or the number of types exceeds the maximum number of types allowed in a type tree.
Invalid reserved word	Check the command line and correct.
Memory failure	Close applications and run Type Tree Maker again.
[] mismatch	Check the command line syntax and correct.
Cannot open type tree	The type tree file does not exist.
Cannot set type information	The type was not created properly.
Cannot set character attributes	The active type is not a character item.
Cannot set binary attributes	The active type is not a binary item.
Cannot set delimited format	The active type is not a delimited group.
Cannot set implied format	The active type is not an implied group.
Cannot add component	The active type is an item that cannot have a component list.
User aborted	The user pressed Cancel.
Cannot set initiator	The type was not created properly.
Cannot set terminator	The type was not created properly.
Cannot set release character	The type was not created properly.
Cannot delete restriction list	The active type is not an item.
Cannot delete component list	The active type is an item that cannot have a component list.
Cannot add restriction	The active type is not an item. Only items can have restrictions.
Cannot propagate	The type was not created properly.
Cannot get type information	The type was not created properly.
Cannot get initiator	The type was not created properly.
Cannot get terminator	The type was not created properly.
Cannot get release character	The type was not created properly.
Cannot get delimited format	The active type is not a delimited group.
Cannot get implied format	The active type is not an implied group.
Cannot get character attributes	The active type is not a character item.

Message	Description
Cannot get binary attributes	The active type is not a binary item.
Cannot replace type - class of type is incorrect	The active type does not match the class.
Cannot get type name	The type was not created properly or the path name was specified incorrectly.
Cannot get type information to replace type	The type was not created properly.
Hex value is not valid	The valid Hex characters are 0-9 and A-F.
Cannot save type tree specified in last NEWTREE= command	The path name is invalid.
Component rule is invalid	Check the syntax of the component rule.
Cannot delete subtype	The type does not exist.
Analysis errors occurred	The tree did not analyze correctly.
Invalid file name specified	Check for invalid characters or the size of the file name.
Invalid type name	Check the type name for reserved characters or reserved words.
Overwrote existing type (and possible subtree)	The type already existed in the tree and was overwritten.
Separator option has Find set but no default	For the Separator option, you must specify Default when you specify Find .
Analysis warnings occurred	The Type Tree Analyzer produced warnings.

Type tree generation logic warning messages

The following messages are generated if you use the Type Tree Maker to create a type tree.

Table 1.

Return Code	Message
L202	RESTRICTION list deleted: TYPE is not an ITEM - TYPE: 'type name' Hint: Type class was changed from an item to a group or category, so the restriction list was deleted. If this was not your intent, change it back to the way it was.
L207	Root of TYPE Tree is not a CATEGORY - TYPE: 'type name'
L208	DELIMITER location not specified - TYPE: 'type name'
L209	SYNTAX ERROR in COMPONENT RULE - COMPONENT # of TYPE: 'type name'
L210	TYPE created from an ITEM or a GROUP is a CATEGORY - TYPE: 'type name'
L211	TYPE created from an ITEM is a GROUP - TYPE: 'type name'
L212	TYPE created from a GROUP is an ITEM - TYPE: 'type name'
L213	Floating Component TYPE deleted because TYPE has an Explicit format - TYPE: 'type name' (warning)
L214	Invalid ITEM presentation. Presentation set to defaults - TYPE: 'type name' (warning)
L215	DELIMITER length greater than maximum allowed in TYPE: 'type name' Hint: Delimiters, initiators, and terminators are limited to 120 bytes.
L216	The Padded To length for item - 'type name' was less than the maximum content size.
L217	Fixed TYPE - 'type name' must include white space or have a terminator.
L218	RELEASE CHARACTER for TYPE - 'type name' must have a value.
L219	INITIATOR for TYPE - 'type name' must have a value.
L220	TERMINATOR for TYPE - 'type name' must have a value.
L221	Name for TYPE - 'type name' contains reserved characters. Please rename.
L222	DELIMITER for TYPE - 'type name' had no value. The group format has been set to Implicit with No syntax. (warning)
L223	No syntax item specified as DELIMITER for TYPE 'type name'
L224	No syntax item specified as RELEASE CHARACTER for TYPE 'type name'
L225	No syntax item specified as INITIATOR for TYPE 'type name'
L226	No syntax item specified as TERMINATOR for TYPE 'type name'
L227	No syntax item specified as 1000's separator for TYPE 'type name'
L228	No syntax item specified as decimal separator for TYPE 'type name'
L229	1000's separator for TYPE 'type name' has no value
L230	Decimal separator for TYPE 'type name' has no value
L231	1000's separator type neither inherited nor local TYPE 'type name'
L232	Default 1000's separators not specified TYPE 'type name'
L233	Default 1000's separator not in restriction list TYPE 'type name'
L234	1000's separator type is not a SYNTAX ITEM TYPE 'type name'
L235	1000's separator type has no restriction list TYPE 'type name'
L236	Decimal separator type neither inherited nor local TYPE 'type name'
L237	Default decimal separator not specified - TYPE 'type name'
L238	Default decimal separator not in restriction list TYPE 'type name'
L239	Decimal separator type is not a SYNTAX ITEM TYPE 'type name'
L240	Decimal separator type has no restriction list TYPE 'type name'

Errors and warnings

When the Type Tree Maker finds a problem with a command, it creates either an error message or a warning message, depending on the severity of the problem. These error and warning messages that can be viewed from the Type Tree Maker window are also stored in the file **ttmaker.log** in your product installation directory. When the Type Tree Maker finishes running, view the errors and warnings in the **ttmaker.log** file.

The **ttmaker.log** file record contains the line number of the command that caused the error or warning message, followed by a message. An example of a warning message is:

```
Warning 34 at line 3: Overwrote existing type (and possible subtree)
```

See [Return Codes and Error Messages](#) for a list of error and warning messages.

A general hint is to always check the spelling and syntax of the commands.

Integration Flow Designer

Integration Flow Designer overview

The Integration Flow Designer is a component of the Design Studio that provides a graphical facility to combine collections of maps and run them as a single unit.

The Integration Flow Designer has an easy-to-use graphical interface. You can create system diagrams, what-if scenarios, generate process control information, and see how the systems will behave at execution time. Additionally, the automation of the Integration Flow Designer helps reduce the number of errors that often result from manual operations.

The term for a collection of maps is system. Using the Integration Flow Designer, you create a system file by combining map components and then saving them as a single file known as a system definition file (.msd).

The Integration Flow Designer provides both local and client functionality. Use the Command Server or Launcher to execute a system's process control information on both Windows and UNIX operating systems.

- [When to Use the Integration Flow Designer](#)

When to Use the Integration Flow Designer

Use the Integration Flow Designer when you have multiple maps to manage within your enterprise. Just one click can build or port as many maps as you have defined in any system.

Note: You must use the Integration Flow Designer to generate systems if you plan to use the Launcher. The Integration Flow Designer is the client definition facility for the Launcher.

You do not need the Integration Flow Designer if you are using a Command Server. However, you will find it to be very useful as a client facility for managing maps that will be run by a Command Server. The Integration Flow Designer generates process control information for Command Servers in the form of command files. Generating these command files manually is tedious and error-prone. Using the Integration Flow Designer eliminates possible manual errors.

Using the Integration Flow Designer

When you use the Integration Flow Designer to create systems from collections of maps, you will typically have maps already created and ready to use. However, using the Integration Flow Designer, you can create placeholder maps called pseudo maps. You can add input and output cards to these pseudo maps and create source maps as you move forward with your design.

For each system, you can select a server and choose the execution mode: **Launcher** or **Command**. If you are using the Launcher, you must generate a Launcher system file (.msl). The .msl file is the only file that the Launcher recognizes. If you are using the Command Server, you can generate text command files.

When you select a server for the system, the system is called an executable system. For each executable system, you can create and run deploy scripts that automate the process of generating and deploying a system to a different server.

After you generate an .msl file, you are ready to run the system using the Launcher.

Stages of using the Integration Flow Designer

The three main stages of using the Integration Flow Designer are discussed.

- [Stage 1: defining systems](#)
- [Stage 2: verifying component relationships](#)
- [Stage 3: preparing systems to run](#)

Stage 1: defining systems

Use the Integration Flow Designer at design time to model what-if scenarios and visualize the results.

In the first stage, you use the Integration Flow Designer to construct system definition files. A system definition file is a model of a system of maps that you want to execute. System definition files are maintained in various system windows that you open and view. The diagrams include map and subsystem components and their execution properties. Data flow relationships are automatically derived and displayed. The system definitions are stored in system definition files (.msd) based on how you decide to organize them. The Navigator helps you to graphically interact with system definition files and the various systems that they contain.

The Integration Flow Designer includes a Doc Links feature that helps you document your design. You can link any text or Windows-registered document type to both map and system components. Then you can view and edit this document by referencing it through the system component to which it is linked. For example, you can link a Microsoft Word document to a map component that specifies design information for this map. You can link a document from a preferred project management tool to a map or a system to assist in managing the project plan. In addition to Doc Links, the Integration Flow Designer enables you to enter free-form text directly to the system definition file.

While you are defining systems, you can use the Integration Flow Designer to open the Map Designer and/or the Database Interface Designer using a map you select in a system. You can also open the Type Designer using a type tree associated with a map input or output card that you select.

Stage 2: verifying component relationships

The Integration Flow Designer displays a system graphically, which allows you to visualize your designs and verify that relationships between components are configured properly.

You can view the contents of maps and subsystems by double-clicking the icons in the system window. This allows you to view the details associated with a component's sources or targets. Visual cues, such as tool tips for specific sources and targets, help you to see key pieces of setup information. Displaying information graphically reduces the need to hunt for information buried within a system definition; however, a find-and-replace operation can also be used.

Stage 3: preparing systems to run

After you design your systems and verify component relationships, you use the Integration Flow Designer to prepare them to run. The Integration Flow Designer assists you in this process by building maps and porting them to designated servers. You can easily build and port entire systems, or specific groups of maps or individual maps.

The Integration Flow Designer also generates process control information based on the execution mode you specify for a system and the execution settings you have specified for the various components within the system. The Integration Flow Designer produces command files for systems that will be run by a Command Server and Launcher system files (.msl) for systems that will be run by a Launcher.

The Integration Flow Designer includes an Analyzer that checks your system definitions for logical consistency to ensure they can be executed. The Analyzer detects any inconsistencies you may have introduced in the definition while experimenting with system models.

Systems that will be run by a Launcher can be designed to run independently on one or more servers or as a distributed system across multiple servers. Event coordination among multiple servers is analyzed at design time before deployment. During execution, data from one system component to another can flow seamlessly without the need for a global controller.

Integration Flow Designer basics

- [Filename extensions](#)
- [Starting the Integration Flow Designer](#)
- [Converting files from previous versions](#)

Filename extensions

The Integration Flow Designer views a map as an indivisible unit recognizing both source and compiled maps. For design purposes, it also enables you to define placeholders for maps that have not yet been implemented (pseudo maps). The Integration Flow Designer generates or uses the following files:

Filename Extension	Description
.msd	System definition file generated by the Integration Flow Designer
.msl	Launcher system file, a system definition file generated for the Launcher
.mmc	Compiled map generated by the Map Designer
.mtt	Type tree file generated by the Type Designer

Note: Pseudo maps and external systems do not have filename extensions.

Starting the Integration Flow Designer

About this task

This procedure assumes that you have installed the Design Studio.

To start the IFD

Procedure

Select Integration Flow Designer from the Design Studio program menu. The Startup window is displayed with the following options:
The Startup window is displayed with the following options:

- Open an existing system definition file
Select this option to browse for a specific .msd file.
- Create a new system definition file
Select this option to create a new .msd file.
- Open a recently used system definition file
Select this option to choose one or more .msd files from the displayed file list. You can also double-click on a file from this list to open it.

To disable the Startup window

Procedure

To disable the Startup window, enable the Do not show this at startup check box. You can always access this dialog from the Help menu by choosing Startup Window.

Converting files from previous versions

When upgrading your version of the Design Studio, some file conversions must take place. Always create backups of your files before converting to a new version.

Integration Flow Designer user interface

The Integration Flow Designer interface includes a navigation pane (Navigator), floating toolbox, and main window for each open system. The menu bar and toolbar are customizable.

- [Navigator](#)
- [Toolbox](#)
- [Menu Commands](#)
- [Customizing your work environment](#)

Navigator

About this task

The Navigator is the navigation pane for the Integration Flow Designer. The Navigator displays a list of open system definition files (.msd) and the systems they contain.

To view and position the Navigator:

Procedure

1. From the View menu, choose Navigator.
The Navigator is displayed.
2. To reposition the Navigator, do one of the following:
 - a. Double-click the border of the Navigator (gripper) to dock or float the navigation pane.
 - b. Right-click on the gripper to view a context menu for positioning the Navigator.

Results

There are two tabs that represent the List view and the Composition view.

- [Navigator icons](#)
- [Navigator: List view](#)
- [Navigator: Composition view](#)
- [Using the Navigator](#)
- [Displaying components from the navigator](#)
- [Dragging components](#)

- [System windows](#)

Navigator icons

In addition to the alphabetical and compositional structure showing the contents of each system file, information about entries in the Navigator is conveyed visually with icons.

The following table lists and briefly describes some of the icons that appear in the Navigator.

Icon	Description
	system definition file (.msd)
	system or subsystem (relative to positioning)
	system component
	log file
	document link
	deployment definition

Navigator: List view

The List view displays open system definition files. This hierarchy includes:

- Defined Servers
- Systems
- Subsystems
- Components within a system
- Cards within a map component
- Doc Links (within a map component)
- Log Files (within a map component)
- Deploy Scripts

A subsystem is displayed under the system that contains it, a map component is displayed under the system that contains it, and so forth. Components at the same level are listed alphabetically.

Click any item to display its components in the system window. Right-click on any icon to view a context menu for that component.

Navigator: Composition view

The Composition view presents open system definition files sorted alphabetically. Each system definition file can be expanded to present all executable systems that it contains. An *executable system* is one that has an assigned server. Systems that do not have an assigned server do not appear in Composition view.

Click any system to display its components in the system window. Right-click on any icon to view a context menu for that component.

Using the Navigator

Use the Navigator to open and interact with one or more system definition files. To view the systems in the Navigator, expand or compress the node using + or -.

Once a system definition file is selected, you can use File menu commands, toolbar tools, or a context menu to close, save or copy the file.

- [Drag-and-drop functionality](#)
- [Clipboard functionality in the Integration Flow Designer](#)
- [System and subsystem components](#)
- [Focus options](#)
- [Resizing/moving options](#)

Drag-and-drop functionality

The following drag-and-drop operations are available in the Integration Flow Designer.

- Servers can be copied/moved across system files or can be copied within the same system file.
- Components can be copied/moved across system files or copied within the same system file.
- Systems can be copied/moved across system files or copied within the same system.
- Doc Links can be copied/moved across system files or within the same system file.

Clipboard functionality in the Integration Flow Designer

- Servers can be copied into the clipboard and/or pasted across system files and within system files.
- Components (systems or maps) can be copied and/or pasted across system files and within the same system file.
- Doc Links can be copied and/or pasted across system files and within the same system file.

System and subsystem components

- The symbols next to the component names in the Navigator indicate the type of component. Even though both systems and subsystems have the same symbol, it is their position in the Navigator that determines the reference. For example, system symbols always appear directly under a system definition file symbol. A subsystem symbol can appear anywhere below a system symbol. Right-clicking on these symbols yields a different context menu depending on where the symbol is located.
- A question mark is displayed on the subsystem component icon when unresolved references exist. An unresolved reference is when the subsystem references a system that does not exist.

Focus options

The following focus options are available when using the Integration Flow Designer.

- When you double-click a component in the Navigator, it selects the component, opens the system window containing that component (if it is not already open), and refocuses the system window on the selected component.
- Place the cursor on a system, subsystem, or map component symbol in the Navigator and double-click or press `Enter` to synchronize the focus of the Navigator and the active system window.

Resizing/moving options

- Use the Navigator gripper to dock or float the Navigator.
- Right-click on a non-client area in the Navigator to view a context menu that enables options to undock, float, or hide the Navigator window.
- To re-dock the Navigator, drag-and-drop the Navigator to the left or right edge of the main window of the Integration Flow Designer.
- The Navigator can be sized horizontally when docked, or it can be sized as a normal window when undocked.

Displaying components from the navigator

About this task

From the Navigator you can display a specific system, subsystem component or map component. When you double-click a component in the Navigator, the corresponding system window opens and becomes the active window.

Dragging components

In the Integration Flow Designer, you can perform drag-and-drop operations on servers, components, and document links within the Navigator. To move items, simply click-and-hold and drag the item to the desired location. To copy an item, you must also press `CTRL` while dragging the item.

You can only move or copy items to items of similar nodes.

System windows

The Integration Flow Designer main window displays any open system windows. Each system window is composed of a work area, a title bar, drop-down lists to specify the system server and execution mode, and a platform indicator displaying the type of platform associated with the assigned server. Specify the server and type of platform in the Add Server or Edit Server dialog by selecting New or Edit from the Server menu.

Multiple system windows can be open at the same time so you can view and work on several system definition files. The size and position of each system window is saved in its system definition file; when reopened, each system window is presented in its last state.

System Window Element Description

Title bar

The title bar of each window, as shown in the preceding image, displays the system name and the path of the containing system definition file (.msd).

Because you can have multiple system definition files open concurrently, a system window's title bar helps you to identify which system is being viewed. The same system name can be used throughout the various system definition files that you have open. The combination of the system name and the system definition file that contains the system is unique.

Work area

The work area is the main viewing area of the system window and is where you construct the system definition file, which is the graphical representation of the components that compose the system. It can also show links that represent data flow between the components. Components can be source maps, compiled maps, pseudo maps, or other systems (also known as subsystems).

Property Bar

The property bar is displayed below the title bar to show the assigned **Server**, the **Execution Mode**, and the **Platform** of the assigned server.

Server

From the Server field, you can select a server on which the system will run from drop-down list. This drop-down list contains the list of default servers as well as each server that is defined within the currently selected system definition file

Execution Mode

Assign the execution mode to define the type of Command or Launcher that will run the system. The Execution Mode drop-down list includes Command and Launcher . Selecting Command specifies that this system will be run using a Command Server. Selecting Launcher specifies that this system will be run using a Launcher.

Platform

This field shows the platform type associated with the assigned server. Specify the platform type in the Add Server or Edit Server dialogs which appear by selecting New or Edit from the Server menu.

Toolbox

When enabled from the View menu, the toolbox is displayed automatically when a system window is opened. It provides the tools needed to create or modify a system definition file.

The toolbox location is saved when the Integration Flow Designer is closed. The next time you open the Integration Flow Designer, the toolbox is displayed at the previously saved location.

To use each tool, you must click the button and then click in the system window. Each tool button is described in the following table.

Tool Name	Use This Tool To...
Select	Draw a box around one or more components in the system definition file to select them. You can then perform operations on the selected group.
Add Text	Add a text block to your system definition diagram.
Override	Override sources and targets. The source or target of the first card selected will override the source or target of the second card selected.
Add Pseudo Link	Draw links between system components. Pseudo links serve a documentation purpose and are represented by a red dotted line.
Add Source Map	Add source map components.
Add Compiled Map	Add compiled map components.
Add Pseudo Map	Add pseudo map components.
Add Subsystem	Add subsystem components.
Add External System	Add an external system placeholder.

Menu Commands

You can perform actions in the IFD through menu commands, tools, and shortcut keys. Not all menu commands have tool icons and not all tool icons are represented in the command menus. When working with systems, you can activate commands several ways:

- Select functionality from the menu bar
- Click the tools on the toolbar
- Right-click an icon in the Navigator
- Double-click objects in the Navigator
- Right-click in the system window or component in the system window
- Click the execution settings button on any map icon in the system window

Note: IFD commands are available as listed above; however, most procedural information in this guide uses the menu access as a default method. Please use the access method most convenient for you.

- [Menu Bar](#)
- [File menu](#)
- [Edit menu](#)
- [View Menu \(Alt+V\)](#)
- [System menu](#)
- [Tools menu](#)
- [Window Menu \(Alt+W\)](#)
- [Help menu](#)

Menu Bar

The IFD menu bar provides the common menu structure for commands generally used with Windows programs as well as commands specific to the IFD.

File menu

The File menu provides the options that are generally available in Windows applications.

Using the keyboard, press **Alt F** to view the File menu.

Command	Key stroke	Description
New	Alt+F, N or Ctrl+N	Creates a new system definition file and opens a system window
Open	Alt+F, O or Ctrl+O	Displays the Open dialog for opening an existing system definition file
Close	Alt+F, C	Closes the active system definition file
Save	Alt+F, S or Ctrl+S	Saves the active system definition file
Save As	Alt+F, A	Saves the active system definition file with a new file name
Source Control	Alt+F, L	Displays a submenu with Source Control options
System Definition File Differences	Alt+F, D	Begins the compare system definition file operation
Print	Alt+F, P or Ctrl+P	Prints a copy of the system definition diagram shown in the active system window
Print Preview	Alt+F, V	Displays the active system window in preview mode
Print Setup	Alt+F, R	Changes the printer and printing options
Recently Used File List	Alt+F, 1 Alt+F, 2 ...	Presents up to eight of the most recently used system definition files
Exit	Alt+F, X	Quits the application; prompts to save system files

Edit menu

The Edit menu provides the editing commands that are generally available in Windows' applications. It also contains the **Find** and **Replace** commands, which are useful for adapting systems to new server environments.

Using the keyboard, press **Alt E** to view the Edit menu.

Command	Key Stroke	Description
Undo	Alt+E, U or Ctrl+Z	Reverses the last action, such as cut, copy, paste, and delete operations and the visual positioning of a component within the system window
Delete	Alt+E, D or Del	Deletes the selected component or components
Cut	Alt+E, T or Ctrl+X	Cuts the selection and places a copy on the clipboard
Copy	Alt+E, C or Ctrl+C	Copies the selection to the clipboard
Paste	Alt+E, P or Ctrl+V	Inserts the contents of the clipboard into the active system window
Select All	Alt+E, A or Ctrl+A	Selects all components in the active system window
Find	Alt+E, F or Ctrl+F	Displays the Find dialog for locating system component names in the Navigator
Replace	Alt+E, R or Ctrl+R	Displays the Replace dialog for replacing system component names in the Navigator
More Find/Replace	Alt+E, M	Displays the More Find/Replace dialog for locating and editing system component names and paths in an entire system definition file

View Menu (Alt+V)

The View menu offers you choices of viewing specific items.

Using the keyboard, press **Alt V** to view the View menu.

Command	Key	Description
Toolbars	Alt+V, T	Displays the Customize dialog for toolbars
Status Bar	Alt+V, S	Shows or hides the status bar in the system window
Navigator	Alt+V, N	Shows or hides the Navigator
Toolbox	Alt+V, X	Shows or hides the toolbox
Grid	Alt+V, G	Toggles the display of the grid in the active system definition diagram
Build Results	Alt+V, B	Presents the latest build results
Deploy Results	Alt+V, D	Presents the latest deploy results

System menu

The System menu contains commands that initiate actions for a selected system.

Using the keyboard, press **Alt S** to view the System menu.

Command	Key	Description
New	Alt+S, N	Creates a new system and displays its window
Delete	Alt+S, D	Deletes the active system.
Copy	Alt+S, C	Displays the Copy System dialog to copy the active system as a new system in the same or a different system definition file (.msd)

Command	Key	Description
Rename	Alt+S, R	Displays the Rename System dialog to rename the active system
Build Maps	Alt+S, B	Builds all maps that are referenced by source map components and subsystem components contained in the active system window
Analyze	Alt+S, A	Analyzes the active system.
Generate	Alt+S, G	Generates an Launcher system file (if the system's execution mode is set to Launcher) or generates a command file for a Command Server (if the execution mode is set to Command)
Deploy	none	Displays a submenu for defining deploy scripts. Selecting Definitions from the submenu displays the Define Deploy Scripts dialog to manage deploy scripts for the active system. Selecting a specific deploy script name from the submenu runs the selected deploy script.
Unresolved Settings	Alt+S, U	Processes unresolved settings for the active system definition file

Tools menu

The Tools menu contains commands that initiate actions for defining Integration Flow Designer tools.

Using the keyboard, press **Alt L** to view the Tools menu.

Command	Key	Description
Short Cuts	Alt+L, S	Displays the Shortcut Keys dialog for changing the shortcut key assignments
Options	Alt+L, O	Displays the Options dialog for changing program options
Align to Grid	Ctrl+G	Aligns each selected component to its nearest respective grid intersection point

Window Menu (Alt+W)

The Window menu provides viewing control of the open system windows.

Using the keyboard, press **Alt W** to view the Window menu.

Command	Key	Description
New Window	Alt+W, N	Opens a new system window with the same contents as the active system window. The new system window has a similar name followed by a numeral (1 for the first duplicate, and so on).
Split	Alt+W, S	Splits the active system window into four panes, useful for working on components that are spread out. When the split grid is displayed, click your mouse to position the junction of the boundaries. After the grid is set, click and drag either the horizontal divider, or the vertical divider to a new position.
Cascade	Alt+W, C	Arranges the system windows so that they overlap in a descending pattern
Tile Horizontally	Alt+W, H	Arranges system windows as horizontal non-overlapping tiles
Tile Vertically	Alt+W, V	Arranges system windows as vertical non-overlapping tiles
Next	Alt+W, X or Ctrl+F6	Activates the next system window
Previous	Alt+W, P or Ctrl+Shift+F6	Activates the previous system window
Arrange Icons	Alt+W, A	Arranges system window icons in the main window
Close All	Alt+W, L	Closes all open system windows
Open Windows List	Alt+W, 1 Alt+W, 2 „,	Displays a list of windows that are currently open. Select the one you want to view. A check box indicates which system window is currently active. When ten system windows are open, a More Windows command is displayed to access a dialog to activate any open system window.

Help menu

The Help menu offers information about the IFD version.

Using the keyboard, press **Alt H** to view the Help menu.

Customizing your work environment

You can customize the Integration Flow Designer interface to suit your preferences while working in the application. For example, if you do not want receive confirmation messages upon deletion; you have the ability to turn off that setting. Or, you can create a shortcut for a keystroke you use often.

- [Shortcut keys](#)
- [Options](#)
- [Finding and replacing in the Integration Flow Designer](#)

Shortcut keys

From the Shortcut Keys dialog, you can configure shortcut keys for Integration Flow Designer functions.

Under **Select a macro** is a list of the Integration Flow Designer menu selections. When you click on an item in the list, its description and currently assigned shortcut is displayed. You can create or remove shortcuts as needed.

Refer to Design Studio Introduction documentation for more information about configuring shortcut keys.

Options

From the Options dialog, you can customize options related to viewing, saving, and confirmation messages. To access the Options dialog, select Tools > Options .

Click on each item in the list to view available options.

- [Maps](#)
- [Miscellaneous](#)
- [Navigator](#)
- [Subsystem](#)
- [Confirmations](#)
- [Import and export](#)

Maps

The Maps option specifies how cards are displayed when a map component is expanded.

Miscellaneous

The only Miscellaneous option available is **Backup On Save**, which is enabled by default. When enabled, the Integration Flow Designer automatically creates a backup copy of a system definition file (.msd) when it is saved and places it in the same directory.

Navigator

The Navigator option allows you to do the following:

- Select the font for the text in the Navigator
- Select the style of lines that connect components in the treeview format in the Navigator. Options include **None**, **Solid**, or **Dotted**.
- Enable or disable tooltips to appear when you mouse-over the name of a component that is not fully displayed.

Subsystem

The Subsystems option allows you to specify how cards are displayed when a subsystem component is expanded.

Confirmations

The Confirmations option allows you to enable or disable confirmation messages from appearing upon copy, move, or delete operations. All options are enabled by default, which is the recommended setting.

Import and export

The Import and Export option allows you to enable logging and set up preferences for import and export operations. See [Import and Export Options](#) for more information.

Finding and replacing in the Integration Flow Designer

The Integration Flow Designer supports standard search facilities. There are two find commands:

- **Find** searches for the selected name in the Navigator.
- **More Find/Replace** searches an entire system definition file.

You can perform find and replace operations in the Navigator using the **Find** and **Replace** options separately; or you can perform a complete find and replace operation in a system definition file (.msd) using the **More Find/Replace** option.

- [Finding and replacing component attributes](#)
 - [Using the more find and replace option](#)
 - [Text in a system definition diagram](#)
 - [Printing a system definition diagram](#)
 - [Using the grid](#)
-

Finding and replacing component attributes

A system that you define can include a complex arrangement of components. Subsystems can contain nested subsystems, so executable maps are represented at various locations within the Navigator.

Using the More Find/Replace operation, you can easily locate and modify specific attributes associated with components defined in systems. Use the Find operation to locate card names, path names, or component names within a system; use the Replace operation to change the names. You can also use the Find and Replace operations to change specific settings across a set of components in order to adapt existing systems to new server environments.

Note: You can find and replace component names and path names, however you can only find card names. The replace operation cannot be used to replace card names. The goal of the Find operation is to locate specific attributes associated with system components based on the search criteria that you define. For example, by using the Find and Replace operations, you can change the paths that specify where executable maps are located on target servers. From the More/Find Replace dialog, there are two types of Find options:

- To list only those attributes that match a specific value based on your search criteria, enter a value in the Find What field and click Find.
- To find all the attributes associated with the components in a system that match your specified search criteria, click Find All without entering a value into the Find What field.

Note: The text on the Find All button changes to "Find" when a value is entered in the Find What field.

You can change the display width of a column by moving your cursor on the border between column titles and then clicking and dragging the border to the desired width.

- [Find and replace operations](#)
-

Find and replace operations

About this task

To begin a find operation

Procedure

1. Select the system window from where you want to perform a find operation.
2. Select **Edit > Find**.
The Find dialog is displayed.
3. Enter text in the Find What field, and click Find.
4. To list only those attributes that match a specific value based on your search criteria, enter a value in the Find What field and click Find.

To begin a replace operation

Procedure

1. From the Edit menu, choose Replace.
The Replace dialog is displayed.
2. After you have an entry in the Find what field, enter the new name that is replacing the existing name in the Replace with field.
3. Click Find Next to begin the operation.
4. Click Replace or Replace All to replace the names.

Sorting by column

What to do next

You can sort the results listed in the More Find/Replace dialog by clicking the column headings. When you click a column heading, the list is sorted in alphabetic order. Clicking again reverses the sort order.

Using the more find and replace option

About this task

Use the More Find/Replace option to search an entire system definition file.

- [To find component names](#)
- [To find specific component names](#)
- [To find card names](#)
- [To find path names](#)

- [To find specific path names](#)
- [To replace](#)

To find component names

Procedure

1. From the Edit menu, select More Find/Replace.
2. Enter the component name in the Find What field.
3. Select Component Names from the Search Criteria drop-down list.
4. Click Find.
Results appear under the Component Name heading on the Find tab.
5. To go directly to a resulting item, select it from the list and click Go To.

To find specific component names

Procedure

1. From the Edit menu, select More Find/Replace.
2. Select Component Names from the Search Criteria drop-down list.
3. Click Settings to specify any further search criteria you require.
4. When the Component Name Settings dialog is displayed, enable the components that you would like to find, and click OK.
5. From the Find/Replace dialog, enable Match Case and or Include Referenced Systems as optional search criteria.
6. Click Find to display the search results.
Results appear under the Component Name heading on the Find tab.
7. To go directly to a resulting item, select it from the list and click Go To.

To find card names

Procedure

1. From the Edit menu, select More Find/Replace.
2. Enter the card name in the Find What field.
3. Select Card Names from the Search Criteria drop-down list.
4. Click Find.
The default is to find all card names associated with the system. You can add further settings to find specific card names.
5. To go directly to a resulting item, select it from the list and click Go To.
This allows you to view the component in the associated system definition diagram.

To find path names

Procedure

1. From the Edit menu, select More Find/Replace.
2. Enter the path name in the Find What field.
3. Select Path Names from the Search Criteria drop-down list.
4. Click Find. The default is to find all path names associated with the system. You can adjust the settings to find specific path names.
Results appear under the Component Name heading on the Find tab.
5. To go directly to a resulting item, select it from the list and click Go To.
This allows you to view the component in the associated system definition diagram.

To find specific path names

About this task

Procedure

1. Enter the path name in the Find What field.
2. Select Path Names from the Search Criteria drop-down list.
The Path Name Settings dialog is displayed.
3. Enable or disable path name search options as desired and click OK.
4. From the More Find/Replace dialog, click Find to display the search results.

Results appear under the Component Name heading on the Find tab.

5. To go directly to a resulting item, select it from the list and click Go To. This allows you to view the component in the associated system definition diagram.

To replace

Procedure

1. From the Edit menu, select More Find/Replace.
The More Find/Replace dialog is displayed.
 2. From the Find tab, begin a find operation.
 3. When the results appear in the list, select the check box for each item that you want to change.
 4. Go to the Replace tab.
 5. In the Replace field, enter what is being replaced.
 6. In the With field, enter the replacement text, or select a historical entry from the drop-down list. If you do not enter a value in the With field, the replace operation will substitute a blank string for the string you entered in the Replace field.
 7. Click Find.
- The replace operation completes based on your selection criteria.

Text in a system definition diagram

About this task

You can easily add text to a system definition diagram in order to help document your design.

To add text to a system definition diagram

Procedure

1. From the toolbox, click on the Add Text tool.
2. In the system window, click on the location where you would like to enter text.
3. Enter text. You can move or delete the text by selecting it and choosing the appropriate action from the menu or toolbar.

Printing a system definition diagram

About this task

In the Integration Flow Designer you can print a snapshot of the active system window.

To print system details:

Procedure

1. From the Navigator, select the system you want to print.
2. In the system window, arrange the system components to display how you would like them to print. For example, double-click map source files to expand or hide map components.
3. For best results, select File > Print Preview to view what will print, and then click Print. Otherwise, select File > Print .
The Print dialog is displayed.
4. Select a printer and click OK.

Using the grid

About this task

Components in a system diagram are positioned on a grid that can be viewed by selecting Grid from the View menu or by clicking the grid icon on the tool bar.

- [To view the grid](#)
- [To cause new components to be aligned to a grid point](#)
- [To align existing components to grid points](#)

To view the grid

Procedure

- From the toolbar, click the Grid tool to view or hide the grid.
 - You can also cause subsequent actions that add or move components to position the components on the nearest grid point by enabling the snap to grid mode. Enable or disable snap-to-grid mode by selecting the Snap To Grid Mode tool on the toolbar. When snap-to-grid mode is enabled, existing components are not repositioned.
-

To cause new components to be aligned to a grid point

Procedure

- From the tool bar, click the Snap To Grid Mode tool to enable snap-to-grid mode.
 - When each new component is added, it will be positioned to its nearest respective grid point.
-

To align existing components to grid points

Procedure

1. Select the components to be aligned to the grid.
 2. From the Tools menu, choose Align to Grid or press `Ctrl G`.
Each selected component will move to its nearest respective grid point.
-

How the Integration Flow Designer works

When you use the Integration Flow Designer, you are creating visual process flows and from them creating executable systems that are processed by the Launcher or Command Server.

The Integration Flow Designer enables you to easily manage collections of related maps by graphically organizing them into logical collections of systems. The Integration Flow Designer turns a system into an executable file that is run by the Launcher or Command Server.

- [Theory of operation](#)
-

Theory of operation

You use the Type Designer and Map Designer to produce maps. A map transforms data content from source formats to target formats according to the rules you specify in the map. An executable map is stored in a source map file and built into a compiled map file. Command or Launchers run compiled maps.

The Integration Flow Designer views a map as an indivisible unit, recognizing both source and compiled maps. For design purposes, it also enables you to define placeholders, or pseudo maps, for maps that have not yet been implemented. Consider the Integration Flow Designer as one layer up from the Map Designer.

- [Map components](#)
 - [Command Server or Launcher](#)
 - [Map component execution settings](#)
 - [Systems](#)
-

Map components

A map component is an Integration Flow Designer object that represents an executable map with a defined set of interfaces based on sources, targets, and a surrounding layer of execution settings used at run time.

There are three classes of map components.

A map component references an executable map; it does not consume it. That is, the executable map exists externally and is reusable such that another map component can also reference it. You define an executable map once and then use it as many times as you want in the Integration Flow Designer. Since the Integration Flow Designer treats every map component as a unique instance, you can define multiple map components that reference the same executable map.

The Integration Flow Designer enables you to work from the bottom up. You can roll existing executable maps up into systems. The Integration Flow Designer also enables you to work from the top down. You can start at the top defining systems of maps that have not yet been implemented and you work your way down later to implement these maps. You can also do a bit of both: reference existing executable maps and maps that will be implemented later within the same system.

A map component is an executable unit. The Integration Flow Designer maintains the execution settings that you specify separately with each map component. The referenced compiled map files and source map files are static and are not modified directly with this information. The settings are eventually used on designated servers to execute the maps.

Command Server or Launcher

The Integration Flow Designer manages collections of maps at design time while a Command Server or Launcher manages the execution of maps at run time. The Integration Flow Designer treats a map as an indivisible unit at design time while a Command Server or Launcher treats maps as indivisible units at run time.

The Command Server manages the execution of one system at a time. The Launcher manages the execution of as many systems as you have defined concurrently.

- [Command Servers](#)
 - [Launchers](#)
-

Command Servers

A Command Server executes maps based on commands that you supply using a predefined command-line syntax. You start a Command Server, providing it with one or more commands, and the Command Server terminates when it has completed the last command. The commands for a Command Server can be produced by the Integration Flow Designer from the execution settings that you specify for map components. You can also define the commands manually.

You can use a Command Server to execute one map or multiple maps. A Command Server executes multiple maps in a single-threaded sequential fashion, a scheme that is sufficient for many applications. A Command Server expects all the sources of a map to exist when the map is executed.

Note: Refer to the Command Server documentation for more information.

Launchers

The Integration Flow Designer is the tool you use to create and generate systems for the Launcher to run. A Launcher executes maps based on a sophisticated event-driven model. Maps are launched only when specific events occur and only when required resources to run the map are available. You can specify event triggers, source and target resources, and other settings for a map from the Integration Flow Designer.

For example, on Microsoft Windows systems, the Launcher is a service that runs maps in a multithreaded fashion. The Launcher is started and allowed to run continuously to monitor events and run maps accordingly.

Note: For more information about the Launcher, see the *Launcher* documentation

Map component execution settings

You maintain the execution settings for map components by using the Integration Flow Designer. These settings are eventually embedded into the process control information, which drives the corresponding Command or Launcher. Some execution settings that you specify are used by both Event and Command Servers at execution time, including:

- The location of the map on the server to be executed
 - The source settings for input data
 - The target settings for output data
 - The same map settings available in the Map Designer
 - Retry options that apply if resources (such as work files or log files) that the map requires to execute are in use. Each source and target of the map also has retry options
 - [Launcher settings](#)
 - [Source and target settings](#)
 - [Map settings](#)
-

Launcher settings

Some map component settings are specific to an Launcher, such as time event triggers and input event triggers. Examples of an input event trigger include when a file's timestamp changes, or when a messaging system receives a message.

Note: you are using cluster support for the Launcher, the system definition file (.msd) is a key factor in the behavior of file triggers. To prevent trigger files from being processed more than once, set the OnSuccess card setting to Delete, or configure a keep directory. See the Launcher documentation for details about the keep directory and the OnSuccess setting.

Source and target settings

A source is used to read input data from a file, query a database, retrieve a message from a messaging system, or receive the results of a call to an application. A target is used to write output data to a file, insert or update data into a database, send a message to a messaging system, or call an application to send data to it.

A map component that references an executable map has a default set of sources and targets defined within the referenced source or compiled map file. The sources and targets are the interface points of the map component used within a system. The Integration Flow Designer displays sources and targets of a map component.

You can use the Integration Flow Designer to change a map component's sources or targets by changing its source or target settings. This capability enables you to reuse the functionality of a map and couple it to desired source and target pipelines.

Retry and rollback settings are also available for each source and each target. The retry setting specifies how many attempts will be made to recover from a source or target fault condition that might occur. These attempts are spaced over a specified interval of time. The rollback setting specifies whether source or target data should be rolled back to its original state if the map fails as a transaction unit.

Map settings

Map settings are initially defined for a map in the Map Designer. You can use the Integration Flow Designer to override these map settings. Map settings include definitions for audit log creation, paging values, trace file creation, validation settings, and work file configuration.

Systems

Systems of maps are executable units. The Integration Flow Designer enables you to use a system as a component of another system. This type of system is a subsystem component. A system can contain two types of components: map components and subsystem components. A subsystem component is a reference to another system that you have already defined. As with map components, the Integration Flow Designer displays subsystem components graphically.

A system can consist of map components and subsystem components. Subsystems can also be nested within subsystems. The Integration Flow Designer Navigator displays systems and the contained subsystems in a hierarchical and alphabetical list.

A map component and a subsystem component are conceptually similar. The difference between them is that they operate at different layers. As with a map component, a subsystem is an Integration Flow Designer object. A subsystem does the following:

- Encapsulates an executable system and includes a surrounding layer of execution settings used at run time
- Can include a subsystem component with a defined set of interfaces based on the sources and targets of the various maps that it contains

At design time, you decide which subsystem components you want to add to a system. A subsystem component can reference another system you have defined within a system definition file (.msd).

A subsystem component can also reference a design document that you have produced by using a preferred tool (a type is registered with the Windows platform). This facility helps you to keep track of design-related and implementation-related information at a system level.

As with map components, it is important to recognize that while a subsystem references another system, it does not consume it. The system being referenced is reusable. Another subsystem can also reference the same system.

Systems and the subsystems that reference them have execution settings. The execution settings include the server that will execute the system and the type of Command or Launcher that will run the system.

The following table lists some similarities of maps and systems of maps:

Feature	Map	Systems of Maps
Executable	✓	✓
Contains execution settings	✓	✓
Reusable	✓	✓
Include sources and targets that serve as component interfaces	✓	✓

- [Single-server systems](#)
- [Distributed systems](#)

Single-server systems

You can configure a system to execute on a single server; all maps contained within the system are executed on the designated server. Although all maps are executed on this single server, the actual sources or targets of these maps may cause data to be gathered from and sent to other servers.

A Command Server manages the execution of one system at a time. A Launcher manages the execution of as many systems as you have defined concurrently.

Distributed systems

You can also configure a system so its subsystem components are distributed across a set of servers. Each server has a Command or Launcher that runs the corresponding maps.

Data flows across servers between the distributed system components as indicated in the following example of a distributed system. This data flow is possible because maps can directly communicate data across servers when they execute.

There are four basic source and target types: file, database, message, and application. Within these four types, wide ranges of options are available to retrieve data from a remote location and to send data to a remote location, similar to the following scenarios:

- A map source or target can be a file that is in a shared location, allowing multiple servers to have direct access to it.
- Using the application adapter architecture, you can use any connectivity method to retrieve data from a remote location and to send data to a remote location. For example, you can use sockets, remote procedure calls, or directly call a local application that connects indirectly to another server to pass data. Databases and messages within messaging systems can also be accessed remotely.

The primary benefit of a distributed system is that you decide which subsystem components run on specific servers so that you can balance the overall execution load of the system across the servers in your enterprise. The data flow between servers is managed automatically as maps are executed. You do not need a global manager to

coordinate this data flow.

You can easily add more servers to a distributed system at a later time. The Integration Flow Designer enables you to rearchitect the distribution of subsystem components across a new set of servers.

System definition files

A system definition file contains the definitions for one or more systems and servers.

- [Systems](#)
- [System definition files](#)
- [System definition files and map source files](#)
- [About relocating system definition files](#)
- [Working with system definition files](#)
- [Importing and exporting system definitions](#)

Systems

A *system* is a collection of maps that you organize into a unit. A system can include components that reference executable maps and components that reference other systems. Systems of maps are executable units. The systems you define using the Integration Flow Designer are executed by Command or Launchers.

System definition files

A *system definition file* contains the definitions for one or more systems and the servers on which those systems are to execute.

System definition files and map source files

The following table notes the parallels between a system definition file (.msd) and a map source file (.mms).

system definition file (.msd)...	map source file (.mms)...
Normally contains one or more executable systems. Because a system contains a collection of maps, you can view a system definition file as a level up from a map source file	Normally contains one or more executable maps.
Maintains systems in source format for design purposes. Systems are not executed directly from a system definition file. You analyze and generate process control information for a particular system to execute it	Maintains maps in source format. Maps are not executed directly from a map source file. You build a compiled map to execute it

About relocating system definition files

If you want to copy system files (.msd) from one machine to another, be aware of the dependencies or references to other source map files (.mms), compiled map files (.mmc), and text or Windows-based design documents that you have produced.

The server map location is used to resolve paths specified as relative to the map location, such as the audit log, trace file, work file, and source and target files.

Whenever an .msd file is copied or moved to a different location or a different machine, the set paths for the contained source files (such as map source, compiled map, or subsystem component) change. When you open the system, the source files cannot be found. In this case, you must change the paths to reflect the new file location. You can change paths by either performing a find-and-replace operation or by editing the component.

Working with system definition files

About this task

This section describes the basic procedures for working with .msd files, including opening, creating, closing, and so forth.

- [To create a new system definition file](#)
- [To open a system definition file](#)
- [To display the systems contained in a system definition file](#)
 - To display systems in a system definition file, choose one of the following options:
 - [To close a system definition file](#)
 - [To save a system definition file](#)
 - [To create a system](#)
 - [To copy a system to another system](#)
 - [To rename a system](#)
 - [To delete a system](#)

- [Assigning a server to a system](#)
 - [Modifying a server definition](#)
-

To create a new system definition file

Procedure

- From the File menu, choose New.
 - A new system definition file icon named SystemFile **X** (where **X** is a sequential number) is displayed in the Navigator. A new system window opens with the default name System1.
-

To open a system definition file

Procedure

1. From the File menu, choose Open.
The Open dialog is displayed.
 2. Select an .msd file and click Open.
-

To display the systems contained in a system definition file

To display systems in a system definition file, choose one of the following options:

Procedure

- In the Navigator, click on a system definition file name.
Systems assigned a server of **None** are not executable.
 - In the Navigator, click + to expand the system and view its components.
 - Double-click on the system node or select the system node and press **Enter** to open the system window.
-

To close a system definition file

Procedure

1. In the Navigator, select a system to close.
 2. From the File menu, choose Close.
If you made changes to a system since it was last saved, a confirmation dialog is displayed giving you the option to save changes.
-

To save a system definition file

Procedure

1. In the Navigator, select a system definition file name.
 2. From the File menu, choose Save.
If this is a new system definition file, the Save As dialog is displayed so you can enter a file name.
 3. Enter the name of the file to be saved and click Save.
-

To create a system

Procedure

1. In the Navigator, select the system definition file that is to contain the new system.
 2. From the System menu, choose New.
The New System dialog is displayed with a default name of System**X**, where **X** is a sequential number.
 3. Accept the default name or enter a unique name for the system, and click OK.
A system window is displayed with the new system name and the path name of the system definition file displayed in the title bar.
-

To copy a system to another system

Procedure

1. In the Navigator, select the system that you want to copy.
 2. From the System menu, choose Copy.
The Copy System dialog is displayed.
 3. Enter a unique name and click OK.
The new system name is displayed in the Navigator.
-

To rename a system

Procedure

1. In the Navigator, select the system that you want to rename.
 2. From the System menu, choose Rename.
 3. When the Rename System dialog is displayed, enter a unique name and click OK.
-

To delete a system

Procedure

1. In the Navigator, select the system that you want to delete.
 2. From the System menu, choose Delete.
The Delete System dialog is displayed.
 3. Click OK to delete the system.
-

Importing and exporting system definitions

The Integration Flow Designer permits the import and export of system definition files as XML. When you import or export a system definition file (.msd), the logging option is enabled by default and both the server and system definitions are included.

There are options available for password encryption and for overwriting existing definitions.

- [Exporting system definitions](#)
 - [Importing system definitions](#)
 - [Import and export options](#)
 - [Audit log](#)
-

Exporting system definitions

About this task

For information about the export command line utility, refer to the *Utility Commands* documentation.

The following items are included in the definition when exporting an .msd file:

- deployment scripts
- components (maps, subsystems, external)
- pseudo links
- text components
- component positions within the system window

However, the state of a system window, such as the size or grid setting, is not exported.

Password encryption is an optional setting. To change default settings or for additional information about import and export options, see [Import and Export Options](#).

To export a system definition:

Procedure

1. From the Navigator, select a system to export.
The Save As dialog is displayed.
2. Accept the default or enter a new file name and click Save.
The system definition is saved as an .xml file.

Results

The following XML example displays a simple system definition with one map component:

```

<?xml version="1.0" encoding="UTF-8"?>
<MSDFile FilePath="" ServerCount="0" SystemCount="1" Version="1">
  <System ComponentsCount="1" ExecutionMode="Command" Name="cmndsrvr"
    ScriptCount="1"
    ServerName="Local Server" Version="1">
    <DeployScripts Name="cmndsrvr" Version="1">
      <BuildAndTransferMaps>
      </BuildAndTransferMaps>
      <GenerateAndTransferMslFile FileName="install_dir\systems\evntsvr.msl"/>
      <TransferAdditionalFiles>
      </TransferAdditionalFiles>
    </DeployScripts>
  <MapComponents ComponentName="cmndsrvr" ExecutableMapName="cmndsrvr"
    Expanded="No" LogFileName="install_dir\examples\general\ifd\cmndsrvr\cmndsrvr.log"
    MapCompiledFile="install_dir\examples\general\ifd\cmndsrvr\cmndsrvr.mmc"
    MapSourceFile="install_dir\examples\general\ifd\cmndsrvr\cmndsrvr.mms"
    Type="Source Map" Versions="1">
    <EventServerSettings ConcurrentInstances="0"
      MapServerLocations="install_dir\examples\general\ifd\cmndsrvr\cmndsrvr.mmc"
      Mapdelay="0" Pendinghigh="0" Pendinglow="0" Version="1">
      <Pending_Hour>0</Pending_Hour>
      <EventServer_TimeEvent BeginAfter-Hour="0" BeginAfter-Minute="0"
        BeginAfter-Second="0" SkipIfBusy="Yes" Switch="OFF">
        <Trigger_Once></Trigger_Once>
      </EventServer_TimeEvent>
    </EventServerSettings>
    <ComponentPosition X="281" Y="129"/>
  </MapComponents>
  </System>
</MSDFile>

```

Importing system definitions

About this task

Note: For information about the import command line utility, see the *Utility Commands* documentation.
To change the default settings or for additional information about import and export options, see [Import and Export Options](#).

To import a system definition:

Procedure

- From the Integration Flow Designer menu, select **File** > **Import**.
- The Open dialog is displayed, prompting you to select an .xml file.
- Choose a file and click **Open**.
The Save As dialog is displayed.
- Enter a name for the system and click **Save**.
The system is displayed in the Navigator.

Import and export options

The Integration Flow Designer settings for importing and exporting system and server definition files are located under **Tools** > **Options** > **Import and Export**. Each option is described below.

Enable Logging: Enables or disables logging for the import and export process.

Import

Overwrite Servers: Enable this option to overwrite any server definitions already present in the .msd file. If the option is disabled, the import process fails when a duplicate server name is detected and an error is logged.

Overwrite Systems: Enable this option to overwrite existing system names already present in the .msd file. If the option is disabled, the import process fails when a duplicate system name is detected and an error is logged.

Referenced Servers: Enable this option to import any servers defined within the system to be imported.

Referenced Systems: Enable this option to import any referenced systems defined within the system to be imported.

Export

Encrypt Password: Enable this option to encrypt the server password.

Referenced Systems: Enable this option to include any referenced systems associated with the system definition file to be exported.

Referenced Servers: Enable this option to include any referenced servers defined within the system to be exported.

Audit log

About this task

When the logging option is enabled, the log file is generated by default into the directory of the map that is being imported or exported.

The name of the file begins with the map name, followed by **imexport.log**. For example, **cmndsrvr imexport.log**.

To view the import/export audit log

Procedure

- Select View > Import\Export Results .
- The import/export log is displayed.

Results

The following is an example of an export log:

```
Map cmndsrvr exported successfully.  
System cmndsrvr exported successfully.  
15:23:33, March 02, 2004  
Output File: install_dir\examples\general\ifd\cmndsrvr\cmndsrvr.xml.  
Map cmndsrvr exported successfully.  
System cmndsrvr exported successfully.
```

Map components

A map component is an Integration Flow Designer object that represents an executable map with a defined set of interfaces based on sources, targets, and a surrounding layer of execution settings used at run time.

The Integration Flow Designer treats each map component as a unique object. Consequently, the name, execution settings, corresponding source and target interfaces, and document links are maintained separately for each map component that you add to a system.

Source map components

Source maps reference executable maps defined within a map source file.

Compiled map components

Compiled maps reference executable maps in compiled file format.

Pseudo map components

Pseudo maps reference executable maps that have not yet been implemented.

- [Source map components](#)
- [Compiled map components](#)
- [Pseudo map components](#)

Source map components

About this task

A source map component references an executable map that is defined in a map source file. You can add map components to single-server systems only. Single-server systems run on one server rather than multiple servers and cannot have a distributed server assigned.

A distributed system consists of subsystems that are assigned to run on different servers.

To add a source map component to a system:

Procedure

1. In the system window, place the focus on a system to add a source map component.
 2. From the toolbox, click the Add Source Map tool.
 3. Click on the location in the system window where you want to position the map.
The Add Source Map(s) dialog is displayed.
 4. Enter the name of the map file or click the browse button to locate the file.
The executable map is displayed in the Add Source Map(s) dialog. Note that you can click browse again and open another map file. You can repeat this step as many times as needed.
 5. Enable the check box next to each executable map you want to reference with map components.
After you close the Add Source Map(s) dialog, one map component will be added for every executable map you selected.
 6. Click Select All to add source map components that reference all executable maps displayed in the list.
 7. Click OK.
Each new source map component is displayed in the system window.
- [Modifying a map component name](#)

Modifying a map component name

About this task

You can modify the names of source map components, compiled map components, and pseudo map components.

To modify a map component name:

Procedure

1. Right-click the map component icon and select Edit Map Component from the context menu.
2. Enter the new name in the Component Name field (replacing the old name).
3. Click OK to save changes.

Compiled map components

About this task

A compiled source map component references an executable map within a compiled map file. You can use compiled maps when source maps are not available. You can add compiled map components to single-server systems only.

Use the following steps to add a compiled map component to a system.

Procedure

1. In the system window, place the focus on a system to add a source map component.
2. From the toolbox, click the Add Compiled Map tool.
3. In the system window, click where you want to insert the compiled map.
The Add Compiled Map(s) dialog is displayed.
4. Enter the name of the map file.
The executable map is displayed in the Add Compiled Map(s) dialog. Note that you can click browse again and open another map file. You can repeat this step as many times as needed.
5. Enable the check box next to each executable map you want to reference with map components.
After you close the Add Compiled Map(s) dialog, one map component will be added for every executable map you enabled.
6. Click Select All to add compiled map components that reference all executable maps displayed in the list.
7. Click OK.
Each new compiled map component is displayed in the system window.

Pseudo map components

About this task

Pseudo maps are placeholders for executable maps that have not yet been implemented. You can add pseudo map components to single-server systems.

To add a pseudo map component to a system:

Procedure

1. In the system window, place the focus on a system.
 2. From the toolbox, click on the pseudo map.
 3. In the system window, click where you want to insert the pseudo map.
The Add Pseudo Map(s) dialog is displayed.
 4. Enter a name for the pseudo map component and click Add.
The component name displays in the Component Name column in the Add Pseudo Map(s) dialog.
 5. Continue to add as many names for pseudo maps that you want to include as components in your system definition diagram.
A check mark automatically appears next to the component name to indicate that it will be added to the system file. You can also do the following:
 - Change the component name by clicking Modify Name.
 - Add a document link by clicking Browse Link.Component names must be unique within a system. A warning message appears if you try to add a duplicate component name.
 6. Click OK.
- [Defining sources and targets for a pseudo map](#)
 - [Editing input and output cards](#)
 - [Creating a source map from a pseudo map](#)

Defining sources and targets for a pseudo map

About this task

After a pseudo map is added to a system file, you can define sources and targets by adding input and output cards.

To add an input card:

Procedure

1. Right-click the pseudo map icon and select Add Input Card from the context menu.
The Add Input Card dialog is displayed.
 2. In the CardName field, enter a unique name for the input card. This is the only required field, however you can specify other values as needed. If you choose to specify the **TypeTree** and **Type** values, they must reference a valid type in an existing tree. See the Map Designer documentation for more information.
 3. Click OK to add an input card.
-

Editing input and output cards

About this task

The settings for an input card or an output card can be modified any time after the card has been added.

- [To edit an input or output card](#)
 - [To delete an input or output card](#)
-

To edit an input or output card

Procedure

1. Right-click the input or output card icon and select Edit from the context menu.
 2. Make your changes to the card settings and click OK to save changes.
-

To delete an input or output card

Procedure

1. Right-click the input or output card icon and select Delete from the context menu.
 2. Click Yes when a dialog is displayed to confirm the deletion.
-

Creating a source map from a pseudo map

Before you begin

Before creating a source map, you must first add at least one output card with a valid type and type tree to the pseudo map. This procedure assumes that you have already added the input and output requirements.

About this task

After the pseudo map component is defined, you can convert it into a source map component. Pseudo maps are the only components in the Integration Flow Designer that allow you to create source maps and add input and output cards.

During source map creation, you can either create a new map source file (.mms) or select an existing map source file. Within the map source file, you can either create a new executable map or overwrite an existing map.

Procedure

1. Right-click the pseudo map icon and select Create Source Map from the context menu.
The Create Source Map dialog is displayed.
2. In the Map Source File field, do one of the following:
 - If you want to create a new .mms file, accept the default map source file or enter a new name.
 - If you selected an existing .mms click GRAPHIC (browse) next to the Map Source File field and choose the file.
3. In the Executable Map Name field, do one of the following:
 - If you want to create a new .mms file, either accept the default executable map name (which is the same name as the pseudo map), or type a new map name.
 - If you selected an existing .mms file and you want to overwrite one of the existing executable maps, click the browse button next to the Executable Map Name field to display the list of executable maps within that file. Select the executable map and click OK.
 - If you selected an existing .mms file and you want to create a new executable map, type the new name in the Executable Map Name field.

4. To edit the input or output cards, select the card from the list and click Edit.
5. To create the .mms file, click Create.
6. When a message is displayed to confirm the map was created successfully, click OK.
The pseudo map component is converted to a source map component (the icon color changes).

Viewing sources and targets of a map component

About this task

When you are working with systems in the Integration Flow Designer, you can view and modify sources and targets (inputs and outputs).

To display sources and targets of a map component:

Procedure

You can display map component sources and targets (inputs and outputs) by doing one of the following:

- Double-click a map component.
- Select the map component and click the Expand/Contract button on the toolbar.
The expanded view of a map component displays (or hides) the input and output cards with corresponding card names and numbers
If the audit log file option is enabled for the map, the audit log file is presented as an output (MapSettings > MapAudit > Switch = ON).

Results

From the system window, after you expand the contents of a map component, you can right-click on a card to view a context menu. From the context menu, you have the option to Open Type Designer or View Card Details.

- If you choose Open Type Designer, the selected card opens in the Type Designer.
- If you choose View Card Details, the View window is displayed for the card.

When a map component is expanded, you can right-click to view a context menu. From the toolbox, you can also use the Override tool to override the source or target of a card with the source or target of another card. Each input and output card is represented by an icon that represents its adapter class.

- [Card tooltips](#)

Card tooltips

About this task

The Integration Flow Designer provides card tooltips that display information about the source of data for an input card or the target of data for an output card. The type of information displayed depends on the adapter type.

To view card tooltips, move the mouse cursor over a card and wait approximately two seconds.

Resource type
Tooltip information

File

Displays the file name specified for the card

Database Adapter

Displays the type of database adapter, connectivity information (connect string, data source name, and so on) and table(s) affected

Messaging Adapter

Displays the type of messaging adapter and the **AdapterCommand** defined for the card

Application Adapter

Displays the type of adapter and the **AdapterCommand** defined for the card

- [Viewing multiple tooltips](#)

Viewing multiple tooltips

About this task

When working in a system definition diagram, it is often useful to view the card tooltips of several of the components simultaneously. You can do this by moving the cursor over each card tooltip you want to display. These card tooltips will continue to be displayed until you click anywhere in the system window or move the cursor over any open area in the system window and wait for several seconds. Performing either of these actions clears the card tooltips and restores the card names.

Working with map components

Three basic characteristics of map components include the component name, execution settings (including source and target interfaces), and document links.

If a map component contains a trigger, the appropriate symbol is displayed on the execution settings button.

The Integration Flow Designer treats each map component as a unique object. The name, execution settings, corresponding source and target interfaces, and document links are maintained separately with each map component that you add to a system.

- [Map names](#)
 - [Execution settings](#)
 - [Overriding map settings](#)
 - [Overriding card settings](#)
-

Map names

You choose a map component name when adding the component to a system. It is common to name a map component the same as the executable map that it references. You can also rename a map component within a system. Duplicate component names are not accepted within a system.

Execution settings

This section presents the basic concepts of map component execution settings and describes the available settings according to the Command or Launcher that runs the map.

Map components are executable units. Using the Integration Flow Designer, you can specify the execution settings for a map component, which in turn are used by the Command Server or Launcher to run the map. Because the Integration Flow Designer treats each component as a unique instance, these execution settings are maintained separately for each map component.

Some of the execution settings that you specify are used by both Launchers and Command Servers at execution time. Settings that apply to both engines include the location of the map on the server to be executed, source settings for input data, target settings for output data, and the same set of map settings available in the Map Designer.

- [Viewing execution settings](#)
 - [MapServerLocation](#)
 - [MapSettings](#)
 - [MapDelay](#)
 - [PendingExpiration](#)
 - [Max Concurrent Map Instances setting](#)
 - [TimeEvent](#)
 - [Pending Instance Thresholds setting](#)
 - [Input and output settings](#)
-

Viewing execution settings

About this task

You can define execution settings for each map component in your system. There are two ways to access execution settings.

To view execution settings:

Procedure

In the system window, do one of the following:

- Right-click the map component icon and select Edit Launcher Settings (or Edit Command Server Settings if the execution mode is Command) from the context menu.
- Click the execution settings button on the map component icon.
The Launcher Settings dialog is displayed (or Command Server Settings dialog if the execution mode is Command).

MapServerLocation

The MapServerLocation setting identifies the path of the executable map file on the server. The default value for the MapServerLocation setting is the path of the selected map.

You can define a resource alias for the MapServerLocation in the Resource Registry. Click Tools > Options > Miscellaneous to enter or navigate to a resource configuration (.mrc) file. When you specify resource alias as the MapServerLocation, the path dynamically resolves at run time. See the Resource Registry documentation for details about resource aliases.

MapSettings

MapSettings specify properties for map execution that are not specific to a particular input or output. These include settings for auditing a map, tracing data content, performance settings, validation settings, and map-specific warnings and errors.

Because maps executed with a Command Server can only override those MapSettings that have corresponding execution commands, some MapSettings are not configurable in the Integration Flow Designer. The following table displays a list of which Launcher and Command Server settings are available in the Integration Flow Designer.

MapSetting	Command Server Settings	Launcher Settings
MapAudit		
Data Log	✓	✓
Execution Log	✓	✓
MapSettings Log		✓
CardSettings Log		✓
MapTrace	✓	✓
WorkSpace	✓	✓
Century	✓	✓
Validation	✓	✓
Retry	✓	✓
Warnings	✓	✓
DocumentVerification	✓	✓

See the Map Designer documentation for information about configuring map settings.

MapDelay

The MapDelay setting defines the amount of time in milliseconds that the Launcher waits, after all event triggers are satisfied, before running the map. The default value for the MapDelay setting is **0**.

PendingExpiration

The PendingExpiration setting specifies the amount of time that a map can stay in an initiation pending state before expiring (that is, being dropped and forgotten). The default value for the PendingExpiration setting is **Hour**.

The PendingExpiration > Value setting specifies how many units in the PendingExpiration setting value (**Hours**, **Minutes**, or **Seconds**) to use for measuring the waiting period. The default value for the **Value** setting is **0**.

If the map associated with a given watch has multiple triggers, and at least one event trigger occurs, the map is placed in an *initiation pending* state until all triggers occur. A *watch* pertains to the map and the set of events that initiate it. Maps in an initiation pending state are reevaluated each time another event trigger occurs. The initiation pending state occurs when any of the following conditions are true:

- The instance of the map is defined as single-threaded and one occurrence of the map is already running.
- The map requires an event trigger that has not yet occurred.
- The maximum number of concurrent maps are already running.

If all preceding conditions are eliminated, then the initiation pending state can be removed from the map; if the PendingExpiration period has not elapsed, then the map can run.

After a map instance is initiated, a `Source not available` message might appear if that source is used by an application unknown to the Launcher and one of the following conditions apply:

- One of the previous source event triggers of a pending map no longer exists
- all event triggers occur and another input is not an event trigger, and that source does not exist

The message `Target not available` may appear if a target message, database, or application adapter does not exist.

Max Concurrent Map Instances setting

The Max Concurrent Map Instances setting, when non-zero, specifies the maximum number of concurrent mappings for the map. This setting overrides the dtx.ini MaxThreads, WatchMaxThreads, and DisableMaxThreads Launcher settings for the map where it is set. This map can consume up to its maximum setting, even if the MaxThreads value for all maps has been reached.

TimeEvent

The TimeEvent settings initiate map components of a system that is time-based. Each map component is run based on a time event, a source event, or a combination of multiple events. If a map component is not assigned an event trigger, the Launcher will not initiate that map to run. The TimeEvent settings are described in the following table.

Note: When you analyze a Launcher system using the Integration Flow Designer, a warning is registered for any source or compiled map component that has no associated event trigger.

TimeEvent Setting	Description
Switch	The Switch setting enables or disables TimeEvent options. The default value for the Switch setting is OFF (Disables TimeEvent options).
BeginAfter	The BeginAfter settings specify the period of time in hours, minutes, or seconds to wait (from the time the Launcher starts) before enabling the time trigger.
	<ul style="list-style-type: none"> • Hour - The Hour setting is used to specify how many hours to wait to begin the map. The default value for the Hour setting is 0. • Minute - The Minute setting is used to specify how many minutes to wait to begin the map. The default value for the Minute setting is 0. • Second - The Second setting is used to specify how many seconds to wait to begin the map. The default value for the Second setting is 0.
Trigger	Use the Trigger setting to define the type of time event trigger. You can trigger a map based on a singular time event or cycle, according to a specified time increment. The default value is Once .
Trigger > Interval	Use the Interval setting to describe the recurring period to use for the trigger. The Interval setting is available only if the value of the Trigger setting is Every . The default value for the Interval setting is 1 Day(s) .
Interval > At	Use the At settings to set a specific time for the trigger to start after the start of the interval defined in Interval settings. The At settings are only available if the value of the Trigger setting is Every . The time periods specified in the Hour , Minute , and Second settings describe a time based on a 24-hour clock set to system time. <ul style="list-style-type: none"> • Hour - The Hour setting specifies the hour in which to run the map. The default value for the Hour setting is 0. • Minute - The Minute setting specifies the minute in which to run the map. The default value for the Minute setting is 0. • Second - The Second setting specifies the second in which to run the map. The default value for the Second setting is 0.
OmitDays	Use the OmitDays setting to set the days to omit.
	<ul style="list-style-type: none"> • Monday - The Monday setting is used to omit Mondays from the time trigger schedule. The default value for the Monday setting is No. • Tuesday through Sunday - These settings work the same way as the Monday setting.
OmitHours	Use the OmitHours setting to set the hours to omit.
	<ul style="list-style-type: none"> • 0 - The 0 setting is used to omit the 0 hour (midnight to 1:00 am). The default value for the 0 setting is No. • 1 through 23 - These settings work the same way as the 0 setting (above).
SkipIfBusy	Use the SkipIfBusy setting to ignore the time event trigger if it occurs when a watch instance is already in process. The default value is Yes .
EventCoordination	Use the EventCoordination setting to control the coordination of time event and input event triggers. The EventCoordination setting is available only if the map has both a time event trigger and one or more input event triggers; and the SourceEvent setting for one of the input cards is ON . The input event trigger can have wildcards. The default value for the EventCoordination setting is FirstAvailable .

Pending Instance Thresholds setting

The **Pending Instance Thresholds** setting enables a listener to pause or resume for a specific watch.

Setting	Description
High	Value (no limit) for the watch to pause the event(s) for that watch. The default value is 0 .
Low	Value (no limit) for the watch to resume the event(s) for that watch. The default value is 0 .
Note: If the InitPendingHigh and InitPendingLow settings in the Launcher section of the dtx.ini file are also set, the Pending Instance Thresholds Launcher settings will take precedence.	

Input and output settings

Because maps executed with a Command Server can only override those input and output settings that have corresponding execution commands, some input and output settings are not configurable in the Integration Flow Designer.

Note: If you use cluster support for the Launcher, the system definition file (.msd) is a key factor in the behavior of file triggers. To prevent trigger files from being processed more than once, set the OnSuccess card setting to Delete, or configure a keep directory. See the Launcher documentation for details about the keep directory and the OnSuccess setting.

Input settings

The following table lists input settings.

Table 1. Table 1. Availability of input card settings in Launcher and Command server

Input Setting	Available in Command Server Settings?	Available in Launcher Settings?
FetchAs	No	Yes

Input Setting	Available in Command Server Settings?	Available in Launcher Settings?
WorkArea	Yes	Yes
Backup	No	Yes
SourceEvent	No	Adapter-dependent
ErrorDirectory	No	Yes
KeepDirectory	No	Yes
Source		
Command	No	Yes
OnSuccess	No	Adapter-dependent
OnFailure	Yes	Yes
Retry	Yes	Yes
Scope	No	Yes
Warnings	No	Yes
FetchUnit	No	Yes

Note: See the Map Designer documentation for information on how to configure these settings.

Output settings

The following table lists output Launcher and Command Server settings.

Table 2. Availability of output card settings in Launcher and Command server

Output Setting	Available in Command Server Settings?	Available in Launcher Settings?
Backup		Yes
Target		
Command	Yes	Yes
OnSuccess		Adapter-dependent
OnFailure	Yes	Yes
Retry	Yes	Adapter-dependent
Warnings		Yes
Scope		Yes

Overriding map settings

About this task

Map settings specify properties for map executions that are not specific to a particular input or output. Map settings defined during the design phase are saved in the map source file (.mms) and the compiled map file (.mmc). Using the Integration Flow Designer, you can override these map settings. Any time you change a map setting from the Integration Flow Designer, it is considered an override. This override value is used during the creation of system files for the Launcher or command files for the Command Server.

When you place an override on a map card, the override propagates to the uppermost system level where that card is visible. When an override is placed on a subsystem card for the corresponding map card, the higher-level subsystem card override takes precedence. The override at the highest level always takes precedence.

To override a map setting:

Procedure

1. In the system window, right-click on a map component and select Edit Launcher settings.
The Launcher Settings dialog is displayed. Select a map setting and choose a new value.
2. Click OK to save changes.
The new map settings will be used during execution.

Overriding card settings

About this task

Card settings, which are specific to input and output cards, define the data object the card represents and how that data is retrieved or routed. A map component that references an executable map has card settings that are defined in the map source file (.mms) and the compiled map file (.mmc). Using the Integration Flow Designer, you can specify execution settings that override the card settings.

Note: You cannot override an input card's Source setting with the static file adapter.

To override card settings:

This procedure assumes that **Launcher** is the execution mode for this system.

Procedure

1. In the system window, double-click on a map component to view the inputs and outputs.

2. On the map component icon, click the execution settings button.
The Launcher Settings dialog is displayed.
3. Select an input or output setting and choose a new value.
4. Click OK to save changes.
The new card settings will be used during execution.

Subsystem components

Systems can contain any combination of map components and subsystem components. A subsystem component is an Integration Flow Designer object that represents or references another system that you have already defined. The concept of a subsystem component is that it enables you to reuse a system. As such, a subsystem is an executable unit for which you can specify execution settings for using the Integration Flow Designer.

The Integration Flow Designer treats each subsystem component as a unique instance. Consequently, the name, execution settings, source and target interfaces, and document links are maintained separately with each subsystem component that you add to a system definition file.

The Integration Flow Designer displays a subsystem component graphically in the system window, as shown in the following diagram.

Subsystem components, which are displayed graphically in the system window, have four basic characteristics: name, execution settings (including sources and targets), and optional document links.

- [Subsystem component name](#)
- [Execution settings](#)
- [Inputs and outputs](#)
- [Working with subsystems](#)

Subsystem component name

When you add a subsystem component to a system definition file, you are selecting a system from a list of systems that you have already defined. By default, the name of the subsystem component is the name of the referenced system.

After a subsystem component is added to a system, you can rename it. The name can be any length and can contain spaces. A subsystem component name must be unique because the Integration Flow Designer does not accept duplicate component names within a system file.

Execution settings

About this task

A subsystem component has a defined set of interfaces that are based on the sources and targets (inputs and outputs) of the various maps that it contains. The inputs and outputs of a subsystem are derived according to the following rules.

- An input of a subsystem is a source that is not internally produced by one of its components.
- An output of a subsystem is a target that is internally produced by one of its components, and is not deleted.

To view execution settings for a subsystem

Procedure

From the system window, do one of the following:

- Click the execution settings button on the subsystem icon.
- Right-click on the subsystem icon and select Edit Execution Settings from the context menu.
The System Launcher Settings dialog is displayed.
- [ServerName](#)
- [ExecutionType](#)

ServerName

The **ServerName** setting specifies the type of server on which the system is deployed. The values include **Local Server**, **Distributed**, and **None**. You can also define additional servers that appear in the drop-down list. The default value for the **ServerName** setting is **Local Server**. Each server option is described in the following table.

Option	Description
Local Server	A system is going to run on the local workstation, usually where the Integration Flow Designer is installed. You might want to assign Local Server to systems to use your workstation as a test environment to verify execution. Or, you may plan to use Local Server as a part of your production environment for system execution.
Distributed	A system is composed of subsystems that are assigned to run on different servers in a coordinated manner to distribute the processing. Components in a distributed system must be subsystems, not maps.
None	

No server selected. Because there is no server assigned, the system is not executable.

Systems that have an assigned server value of **None** do not appear in the **Component** view in the Navigator. These nonexecutable systems can be used as subsystems within a system that has an assigned server.

A subsystem with an assigned server value of **None** derives its server from its parent system that has a server assigned. Subsystems always display in the **Composition** view unless the parent system also has a server value of **None**.

ExecutionType

The **ExecutionType** setting specifies the execution mode for the system. Choices include **Launcher** or **Command**. The default setting is **Launcher**, which indicates that the system is deployed for execution by the Launcher. If **Command** is selected, the system is deployed for execution by the Command Server.

Inputs and outputs

A subsystem component has a set of sources and targets that are based on the sources and targets of the various maps that it contains. The sources and targets (inputs and outputs) are the interface points of the subsystem component used within a system.

In general, a system input is a component input that is not internally produced and a system output is a component output that is not deleted. When a system is presented as a component of another system, the subsystem inputs and outputs serve as interface points, allowing it to be connected to other components in the view.

Double-clicking a subsystem icon expands the view to show inputs and outputs. Double-clicking an input or output card displays the execution settings for the subsystem component.

- [Default settings](#)
- [Saving map card positions](#)
- [Overriding card source or target](#)

Default settings

The default settings for inputs and outputs are based on the inputs and outputs of the various maps contained within the subsystem.

In the system diagram above, the following applies:

- A system input card correlates to a specific input card of a map component that it contains.
- A system output card correlates to a specific output card of a map component it contains.

If any of the inputs can trigger the map, then you will see an input event trigger displayed on the map card icon.

Note: The time trigger icon and input event trigger icon are not displayed on the execution settings button of a subsystem component icon.
The following convention is used to identify the subsystem card:



Saving map card positions

When you initially expand a subsystem component, the input and output cards are cascaded with first card on top by default. Similar to map components, you can move subsystem input or output cards to specific positions and the card positions are retained when you save the file.

The following pertains to card positioning:

- Moving the subsystem component in contracted view to a new position resets the card position to the default.
- If the subsystem component is expanded and it is moved to a new location, the card positions are not changed.

Overriding card source or target

In a subsystem, you can override the source and target of a card with the source or target of another card by using the override function. This works the same as with map component input or output cards.

Working with subsystems

About this task

The following procedure details adding a subsystem to a system.

To add a subsystem component to a system

Procedure

1. Double-click an open .msd file and ensure that it is focused in the system window.
2. From the toolbox, click the Add System tool and then click the location in the client area of the system window where you want to add the subsystem.
The Add System(s) dialog box is displayed.
3. Choose an .msd file and click OK.
In the Add System(s) dialog box, the .msd and a list of systems that it contains is displayed.
4. Do one of the following:
 - Click Select All to reference all systems listed in the new subsystem component.
 - Select only the systems in the list that you want to reference in the new subsystem component.
5. To change the name of a system in the list, click to select it and click Modify Name.
6. To link a document to this subsystem, click Browse Link and navigate to the file.
7. Click OK to save changes.
8. When a message is displayed to confirm that the subsystem was generated, click OK. Recursion is not allowed. The subsystem that you add to a parent system cannot contain the parent system as a component. The Integration Flow Designer alerts you if you try to add a subsystem that contains the system you are defining.

Overview of server definitions and execution modes

The distribution of systems among your enterprise servers plays a key role in the behavior of these systems at execution time. You can define servers using the Integration Flow Designer. These server definitions represent servers within your enterprise that execute systems. After a server is defined, you can assign it to a system. You also assign an execution mode to a system to specify the type of Command or Event Server that will run this system. The server and execution mode that you assign to a specific system provides the Integration Flow Designer information it needs to perform key operations. For example, building maps depends on the target server platform. Also, generating appropriate process control files for execution purposes depends on the type of Command or Launcher that will run the system.

- [Server implementation examples](#)
- [Server definitions](#)
- [Working with system definition files](#)
- [Importing and exporting server definitions](#)
- [System execution modes](#)

Server implementation examples

Use the Integration Flow Designer to define systems of varying complexity. The simplest case involves one system that runs on one server. The following sections describe various examples of systems with increasing complexity.

Multiple independent systems running on one server

You can define multiple systems that have no data dependencies among their components and are all executed on the same server.

Although all systems in this configuration are executed on the same server, data can still be gathered from and sent to remote server locations. This behavior is possible because you can set the sources and/or targets of maps contained in the systems to directly cause data to flow from and to remote servers when these maps are executed.

You can also have multiple independent systems running on multiple servers.

Multiple dependent systems running on one server

You can use the Integration Flow Designer to define systems that have data dependencies among their components and are all executed on the same server.

Data sources and targets can exist on remote servers.

Multiple dependent systems running on multiple servers

You can use the Integration Flow Designer to define distributed systems. A distributed system consists of only subsystem components, not map components directly. These subsystem components are distributed across more than one server. Data flows across servers from one subsystem to another based on the sources and targets of maps that you specify to cause this flow.

Complex combinations of systems

You can define varying combinations of the previously described systems to yield very complex systems. For example, you might have a system that is distributed across servers as well as completely independent systems running on the same servers. The design of your systems depends on your requirements.

Server definitions

The Integration Flow Designer includes three predefined server definitions: **Distributed**, **Local Server**, and **None**.

In addition to the predefined servers, you can create custom server definitions for each system. When you create your own server definitions, they appear on the **Server** drop-down list in the system window.

A new system is automatically assigned the default server of **None** and is considered "non executable". Nonexecutable maps do not appear in the **Composition** view of the Navigator.

Each predefined server type is described in the following table.

Server Definition

Used When

Distributed

A system is composed of subsystems that are assigned to run on different servers in a coordinated manner to distribute the processing. Components in a distributed system must be subsystems, not maps.

Local Server

A system is going to run on the local workstation, usually where the Integration Flow Designer is installed. You might want to assign **Local Server** to systems to use your workstation as a test environment to verify execution. Or, you may plan to use **Local Server** as a part of your production environment for system execution.

None

No server is assigned. The system is not executable, but can be used as a subsystem within a system. A subsystem with an assigned server of **None** derives its server from its parent system that has a server assigned. A new system automatically has **None** assigned by default.

Working with system definition files

About this task

This section describes the basic procedures for working with .msd files, including opening, creating, closing, and so forth.

- [To create a new system definition file](#)
 - [To open a system definition file](#)
 - [To display the systems contained in a system definition file](#)
- To display systems in a system definition file, choose one of the following options:
- [To close a system definition file](#)
 - [To save a system definition file](#)
 - [To create a system](#)
 - [To copy a system to another system](#)
 - [To rename a system](#)
 - [To delete a system](#)
 - [Assigning a server to a system](#)
 - [Modifying a server definition](#)

Assigning a server to a system

About this task

In the Integration Flow Designer, there are two ways to assign a server to a system. You can either copy an existing server and drag it to a new server, or select the system in the Navigator and choose **New** from the Server menu. Both methods are described below.

To copy a server to system (drag-and-drop method):

Use this method if you want to add an existing server to a different system definition file (.msd). You must have both files open in the Navigator. **System File A** is the .msd file with the existing server; **System File B** is the .msd file receiving a copy of the server.

Procedure

1. From the Navigator, click and hold on the server in **System File A**, and press **Ctrl**.
2. Drag the server to the root server node in **System File B**.
The mouse pointer will display a plus sign (+) to indicate correct positioning.
3. Release the mouse to drop the server.
4. When a message is displayed to confirm this action, click **Yes**.

Results

You can also use this procedure to duplicate a server within the same system file. A new server with the same properties is generated, however, it has a unique name.

Modifying a server definition

About this task

The following procedures detail how to add or modify a server definition.

To change or assign a server in a system

Procedure

To change or assign a server for a system, select a new server from the Server drop-down list in the system window.

To edit a server definition

Procedure

1. Select a system containing the server definition that you want to edit.
2. Choose one of the following options:
 - a. From the Server menu, choose Edit.
The Servers dialog is displayed.
 - b. Select the server from the list and click OK.
Or
 - c. From the Navigator, right-click on the server and select Edit from the context menu.
The Edit Server dialog is displayed displaying the current server information.
3. Make changes and click OK.

Importing and exporting server definitions

About this task

The Integration Flow Designer permits the import and export of server definitions as XML files. For information about the Integration Flow Designer setting for the Import and Export operations, see Import and Export Options.

- [To export a server definition](#)
 - [To import a server definition](#)
-

To export a server definition

Procedure

1. From the List view of the Navigator, select the Servers node to include all servers, or select a single server to export.
2. From the menu bar, select Servers > Export.
The Save As dialog is displayed.
3. Enter a file name and click Save.
An XML server definition file is created.

To import a server definition

Procedure

1. From the List view of the Navigator, right-click the Servers node and choose Import. The Open dialog is displayed, prompting you to select an .xml file.
2. Choose a file and click Open. The Import - Servers dialog is displayed. Each server defined in the selected .xml file is displayed under Servers.
3. Under Servers, select the server to import and click the appropriate arrow button. The server names are displayed in the Import list.
4. To replace any existing servers with the same names as those you are importing, enable the Overwrite Servers check box. Otherwise, disable the check box to ensure that existing servers are not overwritten.
5. Click OK to complete the import process. The imported servers are displayed under the Servers node in the List view of the Navigator and on the Server drop-down list of the system window.

System execution modes

About this task

Each system you create has an execution mode. You specify the system execution mode by selecting Launcher or Command from the Execution Mode drop-down list.

The execution mode you select determines the type of execution server that will run the system.

- Selecting Launcher execution mode indicates that your system will run on a Launcher.
 - Selecting Command execution mode indicates the system will run on a Command Server.
-

Generating system files

To run an Integration Flow Designer system, you must generate a command (.txt) or Launcher system file (.msl) from your system definition file (.msd). An .msl file is the process control file for the Launcher. Before generating an .msl file, you should first build all maps and then analyze the system.

- [Building maps](#)
 - [Analyzing a System](#)
 - [Generating executable system files](#)
 - [Unresolved settings](#)
 - [System definition file differences](#)
-

Building maps

About this task

The Integration Flow Designer enables you to easily build maps for a specified system. You can build a single map, a selected group of maps, or all maps referenced within a system. The build process is the same as that performed in the Map Designer.

- [To build all maps referenced by all source map components in a system](#)
 - [To build one map referenced by a source map component in a system](#)
 - [To build a collection of maps referenced by a collection of source map components within a system](#)
 - [To view the last build results at any point in time](#)
-

To build all maps referenced by all source map components in a system

Procedure

1. Open a system.
2. From the System menu, choose Build Maps.
All of the executable maps are built. The Build Maps dialog is displayed displaying the progress of the build process. After the Integration Flow Designer builds all the maps in the system, a dialog is displayed displaying the build results.

To build one map referenced by a source map component in a system

Procedure

1. Place the focus on a system in the system window.
2. Right-click a source map component and choose Build.
The selected map is built.

To build a collection of maps referenced by a collection of source map components within a system

Procedure

1. Place the focus a system in the system window.
2. From the toolbox, click the Select tool and select each source map component to include.
3. Select System > Build Maps.
The selected maps are built.

To view the last build results at any point in time

Procedure

- Select View > Build Results.
 - The results from the last build appear.
-

Analyzing a System

About this task

To ensure that a system will execute properly before actually executing it, use the **Analyze** function. This function analyzes the system for logical consistency and displays the results.

To analyze a system:

Procedure

1. Place the focus on a system in the system window.
 2. Select System->Analyze.
The Analyze System dialog appears.
 3. If errors are detected, click Results to view the details.
The Analysis Results dialog appears.
-

Generating executable system files

About this task

In the Integration Flow Designer, there are two execution modes you can use to create a Launcher system file: **Launcher** and **Command**. **Launcher** execution mode generates system files (.msl) for the Launcher. **Command** execution mode produces command files (.txt) for the Command Server. A Command Server runs one command file at a time.

- [To generate a command file](#)
 - [To generate a Launcher system file](#)
-

To generate a command file

Procedure

1. Place the focus on a system in the system window.
2. From the Execution Mode drop-down list, select Command.
3. Select System->Generate.
The Save As dialog is displayed.
4. Specify the path of the command file to be saved.
The generated command file includes one command for each map specified in the selected system. Options for each map are based on the Command Server settings.

Results

```
; Date: Thu May 13 14:44:46 1999 (Version 2.0)
; File: C:\Examples\Business.msd
; System: Business
;
; Component: SourceMap
; Map File: C:\Examples\Business.mms
; Executable Map: SourceMap
;
E:\business\SourceMap.mmc
-WD
-Z14:18:21:26:27:28:29
;
; Component: business1
; Map File: C:\Examples\Business.mms
; Executable Map: business1
;
E:\business\business1.mmc
-ID1 '-T'
-WD
-Z14:18:21:26:27:28:29
;
; Component: business2
; Map File: C:\Examples\Business.mms
; Executable Map: business2
;
E:\business\business2.mmc
-ID1 '-Trig I trigtest1'
-WD
-Z14:18:21:26:27:28:29
;
; End System: Business
;
```

The commands within the generated command file are ordered based on the input and output dependencies of the maps in the system. Where no input or output dependencies exist, the commands are ordered based on the physical positioning of the maps in the system definition file.

To facilitate the moving of components to different positions, use the **Align to Grid** option.

To generate a Launcher system file

About this task

Procedure

1. Place the focus on a system in the system window.
2. From the Execution Mode drop-down list, select Launcher.
3. Select System > Generate.
The Save As dialog is displayed.
4. Specify the path and name of the Launcher system file (.msl) to be saved.
Other saving options:

- If you have a Windows share setup between your client and server machines, then you can save the .msl or command file directly to the server.
- You can use the Deploy option to copy the .msl or command file to the server running the corresponding Command or Launcher. (See *Deploying Systems*.)
- You can manually copy the .msl or command file to the server running the corresponding Command or Launcher.
- For Launchers, place a copy of the Launcher system file in your deployment directory.

When the Launcher runs, it manages the execution of all Launcher system files in the deployment directory. See the Launcher documentation for additional information.

Unresolved settings

About this task

If you use the Map Designer to delete or rename an input or output card that has override settings defined in a system, the Integration Flow Designer detects the change after the map source file is saved or after the compiled map file is produced. Within the Integration Flow Designer, settings that are no longer associated with any card are known as unresolved settings.

You know that unresolved settings exist in your system when one of the following scenarios occurs:

- You open a system and the Unresolved Settings dialog is displayed.
- You have a system open in the Integration Flow Designer and you use the Map Designer to delete or rename a card that is referenced by a map component in the system. Unresolved settings are detected when you save the map source file. When you return to the Integration Flow Designer, the Unresolved Settings dialog is displayed.

From the Unresolved Settings dialog, you can immediately delete or assign the unresolved settings to a different map card; or you can close the dialog and access it again at a later time by selecting Unresolved Settings from the System menu.

To view unresolved settings:

Procedure

1. Open the system that contains unresolved settings.
 2. From the System menu, choose Unresolved Settings.
The Unresolved Settings dialog is displayed.
The dialog displays the settings that are unresolved and are no longer associated with any card. These settings were defined in the Integration Flow Designer as the source or target settings for a card that no longer exists because it was deleted or renamed in the Map Designer.
- [Processing unresolved settings](#)

Processing unresolved settings

About this task

You can process unresolved settings by deleting the card or by modifying the card settings by copying the settings to an existing card.

After you display the affected system in the Unresolved Settings dialog, you can immediately modify the card settings or delete the card.

If you modify card settings, you can drag the updated card to a map card in the system window, which updates the existing card settings in the map component.

To resolve map card settings

Procedure

1. From the Unresolved Settings dialog, if you are not familiar with the map component that is displayed in the Unresolved Settings navigation pane, right-click the card and select Go to component.
The map component is displayed as a selected object in the system window.
2. Return to the Unresolved Settings dialog, and modify the card settings as required.
3. When the card settings are correct, click-and-hold on the card (in the navigation pane) and drag it to the corresponding card in the system window.
4. When the card in the system window is highlighted, release the mouse to drop the card.
5. Click Yes when you are prompted to confirm the copy operation.

To delete unresolved settings

Procedure

From the Unresolved Settings dialog navigation pane, right-click the affected card and select Delete.

System definition file differences

The Source Compare feature enables you to view differences between two system definition files.

- [Comparing system definition files](#)
 - [Comparing systems](#)
 - [Comparing servers](#)
 - [Comparing deploy scripts](#)
 - [Comparing components](#)
 - [Comparing compiled map components](#)
 - [Comparing source map components](#)
 - [Comparing a psuedo map component](#)
 - [Comparing subsystem components](#)
 - [Comparing document links](#)
 - [Comparing system definition files](#)
 - [Viewing and printing system definition file differences](#)
-

Comparing system definition files

When comparing system definition files, the files are considered "different" when:

- Any of the systems that exist in one file and do not exist in the second file.
 - Any of the systems are different.
 - Any of the servers that exist in one file and do not exist in the second file.
 - Any of the servers are different.
-

Comparing systems

When comparing systems, systems are considered "different" when:

- Any of the system properties are different.
 - Any of the components of the system are different.
 - A component exists in the system in the first file but does not exist in the same system in the second file.
 - If any of the deploy scripts are different.
 - If a deploy script exists in the file but not in the second file.
-

Comparing servers

When comparing servers, servers are considered "different" when any of the server settings are different.

Comparing deploy scripts

A deploy script is different if any of the script instructions are different.

Comparing components

A component is different if:

- If any of the doc links are different
 - If any of the doc links are present in the component in the first file but do not exist in the same component in the second file
 - If any of the unresolved settings are different
 - If any of the log files are different
 - If any of the input/output cards are different
 - If any of the input/output cards are present in the component in the first file but do not exist in the same component in the second file
-

Comparing compiled map components

A compiled map component is different if:

- File path of compiled map is different.
 - Map settings are different.
 - Launcher or Command Server settings are different.
-

Comparing source map components

When comparing source map components, components are considered "different" when:

- File path of map source is different.
 - Map settings are different.
 - Event or Command sever settings are different.
 - Map name for source map component is different.
-

Comparing a pseudo map component

When comparing pseudo maps, a component is considered "different" when:

- Pseudo map name is different
 - Event or Command sever settings are different
 - Map settings are different
-

Comparing subsystem components

When comparing subsystem components, components are considered "different" when:

- Referenced system name is different.
 - Referenced system definition file path is different.
 - Event or Command sever settings are different.
 - Any of the log files are different.
-

Comparing document links

A document link is different if the paths are different.

Comparing system definition files

About this task

System definition source file differences are compared for the source files that you select. You can view the results after the differences are identified.

- [To compare system definition source files](#)
 - [To view system definition file differences](#)
-

To compare system definition source files

Procedure

1. From the File menu, choose System Definition File Differences.
 2. The Select First File dialog box is displayed.
 3. Choose the first system definition source file to compare and click Open.
The Select Second File dialog box is displayed.
 4. Choose the second map source file to compare and click Open.
The progress of the comparison is shown briefly in the System Definition Source File Differences Analysis dialog and then the System File Differences window is displayed.
-

To view system definition file differences

Procedure

1. From the System Definition File Differences window, select the item you want to compare.

2. Press F8 to view the next difference or F7 to view the previous difference.

Viewing and printing system definition file differences

You can view and print all settings for an element that you select in the compared .msd file, regardless of the differences. For example, if you select a map source file, the Properties, Map Settings, and Launcher Settings display.

- To expand a list of Properties, Map Settings, or Card Settings, right-click on the list and choose Expand All.
 - To print Map Settings or Card Settings, right-click on the list and click Print Properties.
- You cannot edit information from this window.

Links

A link is a connection that the Integration Flow Designer creates between two components in a system definition file.

Links represent the direction of data flow among the components of a system at execution time. One of the linked components produces data and the other component uses it as a source. Links graphically present the source and target dependencies between system components.

- [Internal links](#)
- [External links](#)
- [Pseudo links](#)
- [Doc links](#)
- [Overriding a card source or target](#)

Internal links

The Integration Flow Designer treats a map component as a self-contained indivisible unit. This basic building block is displayed graphically using various visual cues to convey information about it. Double-click on a map icon in the system window to expand or contract the view.

The expanded view of a map component uses links to distinguish its sources from its targets. These links are called internal links because sources and targets are an integral part of a map component. By default, the Integration Flow Designer displays the sources to the left of a map component and the targets to the right. However, you can drag any of the source or target icons to any new location within a system window. A map's internal links always help you to distinguish source icons from target icons regardless of where these icons are placed.

The Integration Flow Designer also displays the audit log of a map as a target if you enable the map's audit log option (`MapAuditLogSwitch = ON`). The audit log file is represented as a target by a dotted internal link.

In general, the Integration Flow Designer automatically derives and displays internal links for an expanded map component. It uses information in the referenced source or compiled map file together with any source and target overrides that may be specified.

If you change the default sources or targets defined within a source or compiled map file using the Map Designer, the Integration Flow Designer will display these modified definitions when you expand the corresponding map component. Also, if you change the source or target (input or output) overrides for a map component by using the Integration Flow Designer, the changed overrides are displayed when you expand the corresponding map component.

The Integration Flow Designer prevents you from deleting internal links as well as the corresponding source and target icons. You must use the Map Designer to change the number and type of inputs and outputs.

External links

About this task

Map components and subsystem components are the fundamental building blocks of a system. The Integration Flow Designer uses links to display data flow relationships among these components. A link that is displayed between two system components is called an external link because the link is not an integral part of either component. An external link represents the fact that one of the components produces a data target and the other component uses this data as a source.

An external link can connect a map component to another map component, or a map component to a subsystem component, or a subsystem component to a subsystem component. External links are represented by a solid, directed line. An example follows. The map components are expanded, indicating that the internal links are visible. A visual clue that indicates a component is expanded is that the name is displayed in bold text. When the component is not expanded, the name text is regular, not bold.

Results

An external link conveys more information when the map components that it connects are expanded. In this example, you can explicitly view which targets of one component feed which sources of the other component. When a link is displayed between contracted map components, the link implicitly displays the data flow relationship between the two components.

The Integration Flow Designer automatically derives and displays an external link between two components that share common resources as sources and targets. As with internal links, the Integration Flow Designer uses information in the referenced source map or compiled map files together with any source and target overrides that you may have specified to determine if any two components in a system definition file have a data flow relationship.

The Integration Flow Designer automatically derives and displays database links, messaging links, and file links.

A database link is drawn between two map components when the Select statement defined in the database source of one of the maps references the table associated with the database target of another map. You can view the links explicitly if both map components are expanded.

A messaging link is drawn between two map components when the messaging source of one of the maps references the messaging target of another map.

A file link is created between two map components when the file named in the source of one of the maps references the same file in the target of another map.

Changing the default sources or targets defined within a source map or compiled map file using the Map Designer, or changing the source or target overrides by using the Integration Flow Designer has a direct effect on the external links that are drawn.

The Integration Flow Designer prevents you from deleting external links that it has automatically derived. You must either delete one of the components or change the source or target settings to remove an external link.

Pseudo links

The Integration Flow Designer automatically derives external links and displays them using solid directed lines. The Integration Flow Designer also enables you to manually define external links between two components. These are called pseudo links. A pseudo link is displayed using a dotted directed line.

Pseudo links represent data flow relationships that cannot be derived statically at design time. They help you to visualize data flow relationships that are established at execution time. The Integration Flow Designer does not embed pseudo links that you define into the process control information, which drives Command or Launchers. Pseudo links serve a visual purpose only.

- [Using a pseudo link](#)
-

Using a pseudo link

About this task

This section contains the basic procedures for working with pseudo links.

- [To add a pseudo link](#)
 - [To delete a pseudo link](#)
 - [To display or hide links within a system window](#)
-

To add a pseudo link

Procedure

1. In the system window, place the focus on the system that contains the components to link.
 2. From the toolbox, click on the Add Pseudo Link tool and click the source component or card in the system window.
The mouse pointer changes to a cross (+).
 3. Click on the target component or card.
A dashed line is displayed between the two components or cards.
-

To delete a pseudo link

Procedure

1. From the system window, select the link to delete.
Both ends of the pseudo link now have small bold rectangles indicating it is selected.
 2. Press Delete.
The pseudo link is deleted.
-

To display or hide links within a system window

Procedure

1. Place the focus on a system in the system window.
 2. From the toolbar, click the Display/Hide Links tool.
All links will be displayed or hidden.
-

Doc links

The Integration Flow Designer includes a feature called Doc Links to assist you in documenting your design. You can link any type of text file or Windows-registered document to a map component. For example, you can link a Microsoft Word document or a project plan to a map component that specifies design information for the map. You can view and edit the document by referencing it through the map component to which it is linked.

- [Using doc links](#)
-

Using doc links

About this task

You can define Doc Links for each map component and subsystem component as well as the sources and targets. A document link associates a document file of your choice, such as a design specification, with the component. For example, it is best to link a design specification to a subsystem component because it specifies information at the system level as opposed to the map level.

- [To add a document link for a component](#)
 - [To view a document link](#)
 - [To delete a document link](#)
-

To add a document link for a component

About this task

Procedure

1. Right-click the component icon where the document is to be linked.
 2. From the context menu select Doc Links > Add and select the document to be linked by clicking Open.
The document is now associated with the component.
-

To view a document link

Procedure

1. Launch the associated application from the Integration Flow Designer to view Doc Links.
 2. Right-click the map icon with the document link and select Doc Links > document.
Clicking the document will launch the associated viewing application provided that it is registered on the machine running the Integration Flow Designer.
-

To delete a document link

About this task

Note: Deleting a document link does not delete the document, only the link between the component and the document.

Procedure

1. Right-click the map icon with the document link and select Doc Links > Remove.
The Remove Doc Links dialog is displayed.
 2. Select the document link and click Remove.
The document link is deleted.
-

Overriding a card source or target

About this task

You can override a card source or target with the source or target of another card; however, the card names cannot be overridden or changed. Links are created automatically between cards when a relationship is established.

To override an input or output card:

Procedure

1. Place the focus on a system in the system window.
2. Double-click a map component to expand the view.

3. From the toolbox, click the Override tool and then click on the card source or target to copy.
 4. Move the cursor to and click on the card you want to override.
The source or target of the first and second cards are now identical.
-

Deploying systems

Using the Integration Flow Designer, you can create and run deploy scripts that automate the process of preparing systems to run on specific servers. For example, you can create a deploy script to build maps, generate system or command files, or transfer files to a designated server.

- [Using deploy scripts](#)
 - [Running a deploy script](#)
-

Using deploy scripts

About this task

Deploy scripts are only available for executable systems (systems with assigned servers).

- [To access deploy scripts](#)
 - [To configure a script command](#)
 - [To add a deploy script](#)
 - [To remove a deploy script](#)
 - [To rename a deploy script](#)
 - [Build and transfer maps](#)
 - [Generate and transfer files](#)
 - [Transfer additional files](#)
-

To access deploy scripts

Procedure

1. Place the focus on an executable system in the system window.
 2. Select System > Deploy > Definitions
If deploy definitions already exist for the system, they will appear in the submenu list.
 3. Enter a name for the deploy script and click OK.
The Define Deploy Scripts dialog is displayed. The Define Deploy Scripts dialog lists the deploy scripts that have been defined. Clicking a deploy script name in the list causes the Script Commands group to be refreshed based on the definition of the selected script.
The Script Commands group has three commands with a check box next to each: Build and Transfer Maps, Generate and Transfer Files, Transfer Additional Files. Enabling a check box implies that the command will be executed when the script is run. Files are transferred by means of file transfer protocol (FTP) as binary data or ASCII text.
-

To configure a script command

Procedure

Under Script Commands, do one of the following:

- Select the script command and click Details.
 - Double-click the script command name.
-

To add a deploy script

Procedure

1. Open the system that requires the deploy script.
 2. From the System menu, select Deploy > Definitions . The scripts listed are associated with the system whose window was active prior to running the Deploy command.
 3. When the Define Deploy Scripts dialog is displayed, click Add.
 4. When the Add Deploy Script dialog is displayed, give the new deploy script a name and click OK. The script name must be unique within the active system.
-

To remove a deploy script

Procedure

1. Open the system containing the deploy script to be removed.
2. From the System menu, select Deploy > Definitions . The scripts listed are associated with the system whose window was active prior to running the Deploy command.
3. Do one of the following:
 - Select the script to be removed, and click Remove.
 - Right-click the name of the script to be removed and select Remove from the context menu.

To rename a deploy script

Procedure

1. Open the system containing the deploy script to be renamed.
2. From the System menu, select Deploy > Definitions . The scripts listed are associated with the system whose window was active prior to running the Deploy command.
3. Select the script to be renamed by clicking on its name and do one of the following:
 - When the Define Deploy Scripts dialog is displayed, click Rename.
 - Right-click the script to be renamed and select Rename from the context menu.
4. Click OK to rename the script.

Build and transfer maps

About this task

By using the **Build and Transfer Maps** script command, you can identify which maps to be built and to where they are transferred.

- [To build and transfer a compiled map](#)
- [To use the Maps Referenced Below option](#)
- [To modify the component list](#)
- [To specify the directory where the compiled map \(.mmc\) is transferred](#)
- [Editing a path](#)

To build and transfer a compiled map

Procedure

1. Select a system and select System > Deploy > Definitions . The Define Deploy Scripts dialog is displayed.
2. Under Deploy Scripts, select a deploy script from the list.
3. Under Script Commands, double-click Build and Transfer Maps . The Build and Transfer Maps dialog is displayed.
4. From the Maps to Process tab, select one of the two options:
 - The All Maps option (the default selection) to indicate that all maps in the selected system are to be built and transferred when this deploy script is run.
 - The Maps Referenced Below option to indicate that only the referenced maps in the list are to be built and transferred when this deploy script is run.

To use the Maps Referenced Below option

About this task

There are two drag-and-drop operations that you can use to move map references into the list: one from the List view and one from the system window. Instructions for both operations are provided in the following procedure.

- [To drag from the List view](#)
- [To drag from the system window](#)

To drag from the List view

Procedure

1. From the List view in the Navigator, click on a map and drag it to the component list on the Maps to Process tab.
2. When the mouse pointer displays a plus sign (+), release the mouse to drop the map component.

To drag from the system window

About this task

To drag a map component from the system window, you must first ensure that the system window is not maximized. One way to make the active system window smaller is to go to the Window menu and select Cascade. Then you must arrange the open dialog boxes in such a way that they do not block your view of the map component in the active system window. Now you are ready to drag-and-drop from the system window.

Procedure

1. From the system window, click on the map component and drag it to the component list on the Maps to Process tab.
2. When the mouse pointer displays a plus sign (+), release the mouse to drop the map component.

To modify the component list

About this task

To modify the list of components in the Maps to Process tab, do one of the following:

Procedure

- Select an item in the component list and right-click to view a context menu.
- Click a column title button to sort by ascending or descending order.

To specify the directory where the compiled map (.mmc) is transferred

Procedure

- From the Build and Transfer Maps dialog, go to the Server Map Directories tab.
- On the Server Map Directories tab, the values in the Server View column are the directories of map server locations for the particular map components referenced on the Maps to Process tab.

Editing a path

About this task

The directories listed in the **Server View** column are for display only and cannot be edited. The **Integration Flow Designer View** column represents the target destination directory for the given source directory (under **Server View**) on the server.

The paths in the **Integration Flow Designer View** column can be edited either one at a time or by replacing text in a selected group of directories. Right-click to view the context menu.

One directory can be edited by selecting **Edit Integration Flow Designer View** (or by double-clicking a directory in the **Integration Flow Designer View** column), which allows inline editing directly in the list control. After the inline-editing mode is started, a box is displayed around the item in the list.

To edit multiple directories:

Procedure

1. Ctrl-click each directory to include and right-click. A context menu is displayed.
2. Select Replace Integration Flow Designer View.
The Replace dialog is displayed.
3. Enter the part of the selected string(s) to be replaced into the Replace field.
4. Enter the replacement string into the With field.
5. Click OK to modify the selected paths in the list.
If a directory cannot be changed based on the With field input, the directory stays highlighted and an error message is displayed.

Generate and transfer files

About this task

You can generate and transfer .msl or command files. Procedures for both are described in this section.

When deploying systems that contain maps with XML types to non-windows platforms, the XML Schema and XML input files must be transferred in binary format.

- [To generate and transfer a Launcher system file \(.msl\)](#)
- [To generate and transfer a command file](#)
- [Moving a map component to the deploy dialog](#)
- [Moving subsystems to the deploy dialog](#)

To generate and transfer a Launcher system file (.msl)

About this task

This procedure assumes that the execution mode of the active system is set to Launcher.

Procedure

1. In the Define Deploy Scripts dialog, select the deploy script.
2. Under Script Commands, select Generate and Transfer Launcher Control File and click Details.
If the server is local, distributed, or undefined, the Event Server control file path on server dialog is displayed. (The Command file path on server dialog is displayed when using Command execution mode.) For this option, select or name the .msl file and click Save. This specifies the name and location of the .msl file that will be created by the deploy script.
If the server is completely defined, follow the prompts to select the target directory.

To generate and transfer a command file

About this task

This procedure assumes that the execution mode of the active system is set to Command.

Procedure

1. From the active system, select System > Deploy > Definitions.
The Define Deploy Scripts dialog is displayed.
2. Under Script Commands, select Generate and Transfer Command File and click Details.
If the server is local, distributed, or undefined, the Command file path on server dialog is displayed. For this option, select or name the file and click Save.
If the server is completely defined, the Command control file path on server dialog is displayed. For this option, select the target directory and click OK. Click OK again from the Command file path on server dialog. This action specifies the name and location of the command file that will be created by the deploy script.

Moving a map component to the deploy dialog

About this task

Moving map components from a system window to the deploy dialogs is an easy drag-and-drop operation. A map component can also be moved from the Navigator to the dialog.

After the map component is moved to the dialog, the list is automatically refreshed.

Source map components as well as compiled map components can be moved to the dialog. When a script runs, the compiled maps are transferred. Pseudo maps cannot be moved; duplicates will be rejected.

Only map components that reside in the system being deployed can be moved into the dialog. For example, if you are defining a deploy script for system **ABC**, then only map components that are contained in the **ABC** system can be moved to the dialog.

Moving subsystems to the deploy dialog

About this task

Moving subsystems from a system window to the deploy dialog is an easy drag-and-drop operation. A subsystem can also be moved from the Navigator to the Build and Transfer Maps dialog. After the subsystem is moved to the dialog, the list is automatically refreshed.

Similar to map components, only subsystems that reside in the system being deployed can be moved to the dialog. If you are defining a deploy script for system **ABC**, then only subsystems that are contained in **ABC** can be moved to the dialog. If you open a system window unrelated to **ABC**, then subsystem components cannot be moved to the dialog.

If a subsystem has already been moved into the dialog, then any map component that the subsystem contains cannot be moved to the dialog. Also, if a map component has been moved to the dialog, then the subsystem that contains the map component cannot be moved to the dialog.

You can click one item in the list to select it or use the Shift and Ctrl keys to select multiple items. Right-click on one or more selected items in the list to display the following context menu.

If multiple items are selected, the Go To Component is disabled. If you select Go To Component, the system window where the component resides is displayed and the component is selected.

Transfer additional files

About this task

You can transfer files other than the compiled map, Launcher system file (.msl), or command file. For example, you can transfer a text file that contains notes about implementing the system. For each file, you can choose either an ASCII or binary transfer mode.

To transfer additional files

Procedure

1. From the Define Deploy Scripts dialog, select Transfer Additional Files and click Details.
The Transfer Additional Files dialog is displayed.
2. Right-click on an item under Source Location or Target Location to view the context.
 - Selecting Edit Source or Edit Target allows inline editing of the source or target location. After the inline-editing mode is started, a box is displayed around the item in the list indicating that it can be edited.
 - Selecting Replace Target Location allows you to do a find-and-replace operation.
 - Selecting Browse Target Location allows you to browse for a target location.
3. To change the transfer mode from the default setting (ASCII) to binary, click on the Mode field, display the drop-down menu, and select Binary from the list.
4. Click OK to save changes and close the dialog.

Running a deploy script

About this task

After you create a deploy script, you can easily select and run it by selecting System > Deploy. Each deploy script listed on the submenu is associated with the active system.

- [To run a deploy script](#)
- [To view deploy script results](#)

To run a deploy script

Procedure

1. Place the focus on a system in the system window.
2. From the System menu, select Deploy *script name*, where script name is the name of the predefined deploy script. The script runs.
3. When the script ends, click the Results button to view all activity. In addition, all activities reported on the Deploy dialog are written to a log created in the installation directory that you can view at a later time.
If a deploy script fails, an error message is displayed.

To view deploy script results

Procedure

- Select View > Deploy Results.
- The Deploy Results file is displayed in a text editor.

Return codes and error messages

- [Return codes and error messages](#)
- [Troubleshooting tips for error messages](#)

Return codes and error messages

This section contains return codes and error messages that result from an analysis of a system.

The System Analyzer checks the internal consistency of your systems to assure that they can be executed.

- [System analysis structure error messages](#)
- [Systems using wildcards analysis structure error messages](#)

System analysis structure error messages

The following table lists the structure error messages that result from a structural analysis of a system:

Return Code	Message
SE100	Map, <i>name1</i> , cannot be a component because system, <i>name2</i> , is distributed. A distributed system cannot have maps as components.
SE101	System, <i>name1</i> , contained in system, <i>name2</i> , must have a server. Each component of a distributed system must be distributed or have a server assigned (i.e., the server for a component of a distributed server cannot be "None").
SE102	System <i>name</i> has unresolved settings that must be resolved - (error). Unresolved settings must be resolved.
SE103	Wildcards are used in map source/target name, but map will be run by Command Server - Map Component <i>name1</i> , Card Name <i>name2</i> (error). Wildcards cannot be used in map source or target names if a Command Server will execute the map.
SE104	Wildcards are used in directory name that is specified in source/target name - Map Component <i>name1</i> , Card Name <i>name2</i> (error). Directories cannot have wildcards in their names.
SE105	Wildcards are used in database source name - Map Component <i>name1</i> , Card Name <i>name2</i> (error). A database source name cannot contain wildcards.
SE106	Wildcard must be used in at least one source trigger because wildcards are defined for non-trigger sources and/or targets - Map Component <i>name</i> (error). If wildcards are used as sources and/or targets, at least one source trigger must have a wildcard.
SE107	Question mark wildcard is used in a target name - Map Component <i>name1</i> , Card Name <i>name2</i> (error). Question marks cannot be used in target names. A target name can contain at most one asterisk.
SE108	Question mark wildcard is used in a source name that is not a trigger - Map Component <i>name1</i> , Card Name <i>name2</i> (error). Question marks cannot be used in sources that are not event triggers. A source that is not an event trigger can contain at most one asterisk.
SE109	Multiple wildcard characters are used in a source trigger name, but other source triggers are defined - Map Component <i>name1</i> (error). Multiple wildcards can be used in a source trigger name of a map as long as there are no other source triggers.
SE110	More than one source trigger name contains a wildcard character and the wildcard character used is different - Map Component <i>name1</i> (error). If more than one source trigger name contains a wildcard, that wildcard must be the same wildcard.
SE111	Wildcards are used in queue manager name and/or queue name specified for WebSphere MQ name - Map Component <i>name1</i> , Card Component <i>name1</i> (error).
SE112	Source map component <i>name</i> references a non-existent MMS file - (error). Unresolved MMS reference by source map components must be resolved.
SE113	Compiled map component <i>name</i> references a non-existent MMC file - (error) . Unresolved MMC reference by compiled map components must be resolved.
SE114	Subsystem component <i>name</i> references a non-existent system - (error) . Unresolved system reference by subsystem components must be resolved.
SE115	A system contains itself as a component - System <i>name1</i> , Subsystem <i>name2</i> (error) . A system cannot contain itself as a component.
SE116	Invalid <i>source/target name</i> - Map Component <i>name1</i> , Card Name <i>name2</i> (error) . Database, messaging and application adapters must have valid source/target names as checked by the individual adapter.
SE117	Asterisk wildcard is used multiple times in a target <i>name</i> - Map Component <i>name1</i> , Card Name <i>name2</i> (error) . Only one wildcard per target name is permitted.

Systems using wildcards analysis structure error messages

Wildcards can be used to specify a variable name for a source or target. The asterisk and question mark characters are used according to the conditions described in the SE103 thru SE110 error messages detailed in [System Analysis Structure Error Messages](#). The System Analyzer verifies that these conditions are satisfied. When a condition is not satisfied, it is flagged as an error.

Note: For more information about wildcards, refer to the Launcher documentation.

Return Code

Message

SE500

Map component *name* has no triggers. [Warning].
Maps that will be executed by a Launcher should have a trigger, otherwise the map will never run.

SE501

System name has no components. [Warning].
A system should have at least one component.

SE502

Pseudo map component *name1* exists in system *name2*. [Warning].
Pseudo maps should be converted to source map components.

SE503

Triggers are defined for a map that will be executed by a Command Server - Map Component *name1*. [Warning].
Triggers should not be defined for a map that will be executed by a Command Server.

SE504

The execution mode of system *name* does not match the execution mode of subsystems that it contains. [Warning].
The execution mode specified for a system should be the same as the execution modes specified for its subsystems.

SE505

Drive letters should not be used in UNIX paths - Map Component *name1*, Card Name *name2*. [Warning].
Drive letters should not be used in UNIX paths.

Troubleshooting tips for error messages

- [Not enough memory to execute error message](#)
- ["Version Mismatch" error message](#)

Not enough memory to execute error message

Problem: A map is running on a UNIX platform but fails on a different platform displaying the following error message:



Not enough Memory to execute (2)

Solution: The most common reason is that the map being run is not built for that platform. Ensure that the executable map corresponds to the correct UNIX system where the engine is running.

"Version Mismatch" error message

If you deploy a system to an incompatible platform, the logged error message might incorrectly indicate a version mismatch, when it should instead indicate a platform mismatch.

The problem causing an incorrect error message is that the version is reverse byte ordering on UNIX (as opposed to Windows). To the Launcher, it is displayed as a version mismatch because it there is no indication that it was meant for Windows. The Launcher can only conclude that it is not appropriate for UNIX systems.

Database Interface Designer

Database Interface Designer overview

The Database Interface Designer is used to import metadata about queries, tables, and stored procedures for data stored in relational databases. It is also used to identify characteristics of those objects to meet mapping and execution requirements such as update keys and database triggers.

The Database Interface Designer is used to:

- specify the databases to use for a source or target
- define query statements
- automatically generate type trees for queries or tables

After defining database queries or tables in the Database Interface Designer, define your map in the Map Designer where, in map cards, you can specify an input source as either a query or a stored procedure and an output target as either a table or a stored procedure.

Database connectivity is supported under the control of Relational Database Management Systems (RDBMS).

The adapters provide the option of using a driver to connect to the platform of your choice so that you can automatically create type trees for database queries and tables. The adapters also provide a test environment on the PC for maps using data stored in a database.

You can also install adapters on additional systems to provide remote database connectivity, such as with ODBC, Oracle, Sybase, or DB2®, on operating systems such as HP-UX, Sun Solaris, AIX®, and so on.

Non-Windows adapters do not require the Database Interface Designer if you plan to only use **mtsmaker** without a database/query file (MDQ) to generate each type tree. For information about using **mtsmaker**, see the Resource Adapters documentation.

- [Basic steps for using database data](#)
- [Database Interface Designer basics](#)
- [Database/query files](#)
- [Database type trees](#)
- [Database sources and targets](#)
- [Database functions](#)
- [Using stored procedures](#)
- [Database triggers](#)
- [Debugging and viewing results](#)

Basic steps for using database data

About this task

The following is a high-level description of the steps required to use database data.

To use the Database Interface Designer to import database definitions

Procedure

1. Create a database/query file (MDQ).
2. Define a database.
3. If the database is to be used as an input, define a query for that database.
4. Generate the type tree for the query, table, view, stored procedure, or message from which or to which you want to map.
5. If the database is to be used as an output, set keys and designate columns for performing SQL updates.
6. Save the MDQ file.

To use the Map Designer to configure database sources, targets, or operands in a rule

Procedure

1. From the input or output card(s) in the executable map, select Database as the value for either the GET Source or PUT Target setting.
2. Select the MDQ file that contains the database-specific source or target information.
3. Perform the following for an input and/or an output card:
 - In an input card, select a query from the MDQ file.
 - In an output card, specify a table name or stored procedure.

Database Interface Designer basics

This topic introduces the **Database Interface Designer** window and provides information about working with the graphical user interface.

- [Starting the Database Interface Designer](#)
- [Database Interface Designer user interface](#)
- [Menu commands and tools](#)
- [Configuring the environment](#)

Starting the Database Interface Designer

During installation, an entry for the Database Interface Designer is added to the IBM Transformation Extender program folder (listed under the Design Studio).

When the Database Interface Designer runs, the Startup dialog appears. In this dialog, you can select how the Database Interface Designer program should open. The following options are available:

- [Open an existing database/query file](#)

Browse for the MDQ file you want.

- **Create a new database/query file**

Create a new MDQ file.

- **Open a recently used database/query file**

Select one or more files from the displayed file list. You can also double-click on a file from this list to open it.

The Startup dialog can be disabled by enabling the **Do not show this at startup** check box; however, you can always access this dialog from the **Database Interface Designer** window Help menu by selecting the Startup Window menu option.

Database Interface Designer user interface

The main **Database Interface Designer** window provides a graphical user interface in which to create and maintain MDQ files. These files contain database definitions that include information such as database name, connection information, queries, stored procedures, and so forth.

When you create a database/query file in the Database Interface Designer, a file named **Database_Query File** (which is the default file name) followed by an assigned, sequential number appears in the Navigator. This indicates that you can begin defining databases, queries, and so on. Save this database/query file (a file name with an MDQ extension), providing an appropriate name and location.

- [The Navigator](#)
-

The Navigator

The Navigator graphically presents all of your opened MDQ files and the databases that they contain. It also provides a graphical representation of the queries, stored procedures, message queues, and tables or views that have type trees generated. Also displayed are tables and views with update keys defined, as well as variables that are defined in each MDQ file.

- [Changing the appearance of the navigator](#)
 - [Database Interface Designer icons](#)
-

Changing the appearance of the navigator

About this task

The **Navigator** can be shown or hidden. It also can be presented as either a docked window or a floating window. These choices are available from the context menu of the **Navigator**. To access the context menu, right-click on the border of the **Navigator**.

- [To show or hide the Navigator](#)
 - [To float the Navigator in the main window](#)
 - [To dock the Navigator](#)
-

To show or hide the Navigator

Procedure

1. To show the Navigator, from the View menu, select Navigator.
A check mark appears next to Navigator in the View menu, indicating that the Navigator is displayed.
 2. To hide the Navigator, from the context menu of the Navigator, select Hide.
or
Repeat Step 1 above to disable the Navigator selection from the View menu.
-

To float the Navigator in the main window

Procedure

1. Starting with the Navigator being docked, right-click on the top border of the Navigator.
The context menu appears.
 2. Click **Float In Main Window**.
The Navigator is now a separate window, floating in the main window.
-

To dock the Navigator

About this task

This procedure assumes that the Navigator is floating in the main window.

Procedure

1. In the Navigator, right-click anywhere around the border of the window, except in the title bar.
The context menu appears.
 2. If the Allow Docking option is not available, disable the check mark from the Float In Main Window option.
The Navigator is docked.

or

If the Allow Docking option is available, select it and go to the next step.
 3. After selecting Allow Docking, you can toggle between a docked window and a floating window by double-clicking the top border (below the title bar) of the Navigator. Double-click the title bar so that the Navigator is again docked.
-

Database Interface Designer icons

In addition to the hierarchical structure indicating the contents of each MDQ file, information about entries in the Navigator is visually conveyed using different icon representations defining the object and its state. For example, when a type tree has been generated for a query, the query icon changes to represent the existence of a generated type tree.

Menu commands and tools

Actions can be performed in the **Database Interface Designer** window using menu commands, tools, and shortcut keys. Not all menu commands have corresponding tools, and not all tools are represented in the command menus. When working with database/query files, you can activate commands in several different ways:

- Select functionality using the menu bar.
- Right-click on any entry in the Navigator to display its context menu.
- Click the tools on the toolbar.
- Double-click objects in the Navigator.

Note: Database Interface Designer commands are available using the methods listed above. Most procedural information in this guide uses the menu access as a default method. Use the access method most convenient for you.

- [Menu](#)
 - [Toolbar](#)
-

Menu

The **Database Interface Designer** window menu provides the common menu structure for commands generally used with Windows programs, as well as for commands to accomplish specific Database Interface Designer tasks.

Toolbar

The toolbar is a part of the **Database Interface Designer** window, providing quick access to various tools that invoke Database Interface Designer actions while working with database/query files. The applicability and behavior of each tool depends upon the Database Interface Designer object selected and its current state.

- [File menu](#)
 - [Edit menu](#)
 - [View menu](#)
 - [Database menu](#)
 - [Query menu](#)
 - [Tools menu](#)
 - [Window menu \(Alt+W\)](#)
 - [Help menu](#)
-

File menu

The File menu provides the commands that are generally available in Windows applications.

Using the keyboard, press **Alt F** to view the File menu.

Command	Shortcut Key	Description
New	Ctrl+N	Creates a new MDQ file

Command	Shortcut Key	Description
Open	Ctrl+O	Opens an existing MDQ file
Close		Closes the selected MDQ file
Save	Ctrl+S	Saves the selected MDQ file
Save As		Saves the selected MDQ file with a different name
Source Control > Create Project	Alt+F, L, C	Allows you to begin the process of creating a project using the third-party source control application you have selected.
Source Control > Open Project	Alt+F, L, O	Displays the Project window in which you can begin the process of selecting an existing project in which to work using a third-party source control dialog.
Source Control > Get Latest Version	Alt+F, L, G	Displays the Get Latest Version dialog in which you can select related files that you can get.
Source Control > Check Out	Alt+F, L, E	Displays the Check out file(s) dialog in which you can specify the files to be checked out.
Source Control > Check In	Alt+F, L, I	Displays the Check in file(s) dialog in which you can either only check in or both check in and check out again the selected files.
Source Control > Undo Check Out	Alt+F, L, U	Displays the Undo Check Out dialog in which you can reverse a check out of one or more files without either the check out or the undo of the check out showing up in the archive history.
Source Control > Add to Source Control	Alt+F, L, A	Displays the Add to source control dialog in which you select one or more files to add to your source control project archives.
Source Control > Remove from Source Control	Alt+F, L, R	Displays the Remove from source code control dialog in which you can specify the file(s) to be removed from the specified source control project archives.
Source Control > Show Workfiles	Alt+F, L, F	Displays the Project dialog in which you can view all of the workfiles in this project archive.
Source Control > Add Workfiles	Alt+F, L, W	Displays the Add Workfiles dialog in which you can specify the workfiles to be added to the project archives.
Source Control > Show History	Alt+F, L, H	Displays a third-party, source control application-specific dialog that allows you to define what you want to see in a history report generated by that third-party application.
Source Control > Show Properties	Alt+F, L, P	Displays a third-party, source control application-specific dialog that allows you to view labels and other source control-determined information on a revision-by-revision basis as recorded in that third-party source control application.
Source Control > Refresh Status	Alt+F, L, R	Synchronizes the project information for files displayed in your designer with the archival information in your third-party source control.
Database Query File Differences	Alt+F, D	Displays the Select First File dialog in which you can begin the process of specifying the two database/query (MDQ) files to compare.
Trace	Alt+F, T	Enables trace functionality to create a Database Interface Designer trace file for the selected MDQ file
Print	Ctrl+P	Displays the Print dialog in which to specify and print definitions for the selected database(s)
Print Setup	Alt+F, R	Displays the Print Setup dialog in which to configure printer options
Recent File		Opens the selected MDQ file from this list
Exit	Alt+F, X	Quits the application and prompts to save changes

Edit menu

The Edit menu provides the editing commands generally available in Windows applications. It also contains the **Find** command that can be very useful when viewing trace files in a trace window.

Using the keyboard, press **Alt E** to view the Edit menu.

Command	Shortcut Key	Description
Undo	Ctrl+Z	Reverses the last-performed action
Cut	Ctrl+X	Deletes the selection and places a copy on the clipboard
Copy	Ctrl+C	Copies the selection to the clipboard
Paste	Ctrl+V	Inserts the contents of the clipboard to the cursor location
Find	Ctrl+F	Displays the Find dialog in which to locate specified information in the active trace window
Replace	Ctrl+H	Displays the Replace dialog in which to specify the text upon which to search and the text that is replacing it

View menu

The View menu provides selections to control the appearance of your Database Interface Designer environment.

Using the keyboard, press **Alt V** to view the View menu.

Command	Shortcut Key	Description
Toolbar(s)	Alt+V, T	Displays the Customize dialog for toolbars in which to select the toolbars to be displayed and to adjust the tools displayed on specific toolbars
Status Bar	Alt+V, S	Shows or hides the status bar in the Database Interface Designer window
Navigator	Alt+V, N	Shows or hides the Navigator
Trace File	Alt+V, F	Displays the Database Interface Designer trace file for the selected MDQ file in a separate trace window

Database menu

The Database menu provides commands to initiate actions for a selected database.

Using the keyboard, press **Alt D** to view the Database menu.

Command	Shortcut key	Description
New	Ctrl+D or Insert	Displays the Database Definition window so that you can add a database to the associated MDQ file. Note: You must first select the Databases object in the Navigator to be able to do this.
Edit	none	Displays the Database Definition window that allows you to modify the selected database. Note: There is no shortcut key. But you can double-click the database object name to perform the same function as the tool.
Delete	Delete	Deletes the selected database.
Generate Tree From > Table	Alt+D, G, T	Displays the Generate Type Tree from Tables dialog that allows you to generate a type tree from a table or view in the selected database. Note: You can also double-click the Tables object in the Navigator to perform the same function as the tool.
Generate Tree From > Procedure	Alt+D, G, P	Displays the Generate Type Tree from Stored Procedures dialog for the selected database or stored procedure in which you can specify the generation of a type tree from a stored procedure. Note: You can also double-click the Stored Procedures object in the Navigator to perform the same function as the tool.
Generate Tree From > Queue	Alt+D, G, Q	Displays the Generate Type Tree from Queues dialog in which you can specify the generation of a type tree from a message queue
Set Update Keys	Alt+D, S	Displays the Set Table Update Key Columns dialog in which you can specify update keys and the key columns to update in the selected database

Query menu

The Query menu provides commands to initiate actions for a selected query.

Using the keyboard, press **Alt Q** to view the Query menu.

Command	Shortcut Key	Description
New	Ctrl+Q or Insert	Displays the New Query dialog in which to add a new query. Note: You must first select the Queries object in the Navigator to be able to do this.
Edit	Alt+Q, E	Displays the Edit/View Query dialog in which to modify the selected query. Note: You can also double-click the database query name to perform the same function as the tool.
Delete	Delete	Deletes the selected query
Generate Tree	Alt+Q, G	Displays the Generate Type Tree from Query dialog that allows you to generate a type tree from the selected query
Define Variables	Alt+Q, V	Displays the Define Variables dialog in which to specify pseudo variable values. Note: You can also double-click the Variables object in the Navigator to perform the same function as the tool.
Define Trigger	Alt+Q, T	Displays the Trigger Specification dialog in which to specify the input events that must occur to launch a map using the selected query as a data source

Tools menu

The Tools menu provides options to customize your Database Interface Designer environment.

Using the keyboard, press **Alt L** to view the Tools menu.

Command	Shortcut Key	Description
Shortcuts	Alt+L, S	Displays the Shortcut Keys dialog in which to assign shortcut keys or key combinations to specific Database Interface Designer operations
Options	Alt+L, O	Displays the Options dialog in which to configure Database Interface Designer options for backups, the Navigator, trace windows, tables/views, and confirmations

Window menu (Alt+W)

The Window menu contains commands to provide control of open trace windows.

Using the keyboard, press **Alt W** to view the Window menu.

Command	Shortcut Key	Description
Close All	none	Closes all open trace windows
Cascade	none	Arranges all open trace windows so that they overlap in a descending pattern
Tile Horizontally	none	Arranges all open trace windows as horizontal, non-overlapping tiles
Tile Vertically	none	Arranges all open trace windows as vertical, non-overlapping tiles
Arrange Icons	none	Arranges all minimized trace windows in an orderly fashion
List of recently used files	none	Selecting a specific trace file from this list makes it the active window. A check mark appears next to the trace file that is in the active trace window.

Help menu

The Help menu offers choices to display information about the Database Interface Designer.

Using the keyboard, press **Alt H** to view the Help menu.

Command	Shortcut Key	Description
Contents	none	Displays the contents of the Help system which also lists the Help topics
Startup Window	none	Displays the Startup dialog in which to select whether to create a new or open an existing MDQ file
About Database Interface Designer	none	Displays application-specific information

Configuring the environment

Much of the Database Interface Designer environment can be configured to accommodate your preferred work environment. For example, you can:

- Specify various user interface options (font, line appearance, dialog display)
- Select the tools to display
- Change the look of the tools on the toolbar
- Assign shortcut keys
- [Tools > Options](#)
- [Shortcut keys](#)

Tools > Options

From the Tools menu, select Options. The Options dialog appears. Select choices representing various aspects of the Database Interface Designer environment and configure them as desired.

- [General options](#)
- [Navigator options](#)
- [Trace window options](#)
- [Tables/views option](#)
- [Confirmations options](#)

General options

In the list of options, select General to specify values concerning the backing up and saving of your MDQ files. The fields in this dialog are as follows:

Field	Description
Auto-save files every <i>n</i> minutes (where <i>n</i> represents a number)	This spin button indicates the time interval at which to automatically save opened MDQ files. The default value is 0.
Show Banner	This check box specifies whether to display the banner, which appears between the title bar and the menu bar of the Database Interface Designer window. The default setting is enabled.
Backup on save	This check box specifies whether to create a backup copy of each MDQ file when the file is saved. The default setting is enabled.
Configuration File	Enter the name of the resource configuration (.mrc) file to use to resolve resource aliases.

Navigator options

In the list of options, select Navigator to specify how to display objects in the Navigator.

Trace window options

In the list of options, select Trace Window to specify the font used to display text in the Trace window.

Tables/views option

In the options list, select Tables/Views to determine the objects to be displayed that are associated with the database. The fields in this dialog are as follows:

Field	Description
List tables	This check box determines whether tables are displayed in the database list. The default value is enabled.
List views	This check box determines whether views are displayed in the database list. The default value is enabled.
List synonyms	This check box determines whether database synonyms are displayed in the database list. The default value is enabled.

Confirmations options

In the list of options, select Confirmations to specify the actions for which you want a confirmation dialog displayed before completion of those actions.

Field/Button	Description
Database operations	
Deleting database(s)	This group box displays the options that control whether confirmation dialogs are displayed with regard to database operations. Select any or all of these options.
Copying database(s)	This check box determines whether a confirmation dialog appears when deleting a database. The default value is enabled.
Moving database(s)	This check box determines whether a confirmation dialog appears when copying a database. The default value is enabled.
Table operations	
Deleting table(s)	This check box determines whether a confirmation dialog appears when deleting a table. The default value is enabled.
Copying table(s)	This check box determines whether a confirmation dialog appears when copying a table. The default value is enabled.
Moving table(s)	This check box determines whether a confirmation dialog appears when moving a table. The default value is enabled.
Query operations	
Deleting query(s)	This check box allows you to display the name of each object when the cursor is held over it and the Navigator is sized too small for the name to be completely displayed. The default setting is enabled.
Copying query(s)	This check box determines whether a confirmation dialog appears when deleting a query. The default value is enabled.
Moving query(s)	This check box determines whether a confirmation dialog appears when copying a query. The default value is enabled.

Shortcut keys

You can assign your own shortcut keys for any existing or new menu items. Using the Shortcut Keys dialog, you can:

- Assign shortcut keys.
- Remove shortcut key assignments.
- Restore the shortcut key assignments present at installation.

For information about these procedures, see the Design Studio Introduction documentation.

Database/query files

This topic discusses how to use the Database Interface Designer when working with MDQ files to perform the following types of tasks:

- Create MDQ files and learn about their XML format. (For more information about the XML format, see [Understanding the MDQ XML format](#).)
- Define the various objects included in an MDQ file. (For more information about such objects as queries and variables, see [Defining a query](#) and [Defining Variables in SQL Statements](#).)
- Generate type trees. (See [Generating type trees](#).)
- Print reports and enable trace functionality. (See [Printing reports](#) and [Database Interface Designer trace files](#).)
- [Creating database/query files](#)
- [Defining a database](#)
- [Defining a query](#)
- [Editing or deleting a database or query](#)
- [Understanding the MDQ XML format](#)
- [Comparing database/query files](#)
- [Defining variables in SQL statements](#)
- [Generating type trees](#)
- [Printing reports](#)
- [Database Interface Designer trace files](#)

Creating database/query files

About this task

An MDQ file contains the definitions for one or more databases, as well as queries, stored procedures, and other specifications, that may contribute to the execution of a map. Use the commands on the File menu in the Database Interface Designer to create and save a database/query file with an **.mdq** file extension. This file name and path appears in the title bar of the **Database Interface Designer** window when it is the selected file in Navigator, indicating that it is the active MDQ file.

After starting the Database Interface Designer, the Navigator lists one or more MDQ files depending upon whether your selection was to create a new MDQ file or to open one or more existing files.

When an MDQ file is created, it appears in the Navigator next to the appropriate icon. The **Database_QueryFile** file name is automatically assigned, along with a sequential number.

To save an MDQ file or rename it:

Procedure

1. From the File menu, select Save As.
The Save As dialog appears.
2. Enter the new file name and select the path.
3. Click OK.

Defining a database

About this task

When an MDQ file appears in the Navigator, you can add new database definitions to it or you can modify the name of an existing database. This is done using the Database Definition window. Each **Database Definition** window contains some settings that are common across all platforms and others that are platform-specific. To view more information about each platform-specific setting, refer to either the context-sensitive help available from the dialog itself or to each platform-specific reference guide.

For the purposes of outlining a basic procedure to define a database, the ODBC adapter for the Windows platform will be used. An example follows.

see the following table for a listing of the settings in this window and their descriptions.

Setting	Description
Database Name	This is the name of the database being defined. This name appears in the Navigator.
Adapter	Adapter properties: Type This is a list of adapters that can host the database you are defining. Select one from the list. The default value is ODBC. This selection will affect the list of supported platforms displayed in the Platform list.
Platform	This is a list of platforms upon which the adapter (that you selected in the Type list) is supported.
Access user tables/procedures only	This setting determines level of user access and, to some extent, the presentation of the information being presented. The default value is Yes. If No is selected, all of the names (of tables, views, and so on) in the database are accessible and are presented, including the respective prefix of each associated user ID. If Yes is selected, only the names (of tables, views, and so on) in the databases associated with the current user ID are accessed and presented. Because these are all associated with the same user, no prefix is added.
Surround table/column names with	This setting indicates the character you want to use to enclose table and column names. For example, many databases require that names be enclosed by single or double quotation marks if they have spaces in their names, or when the names are SQL reserved words. When you specify table names in output cards or column names in update keys, these names will be enclosed by the character defined in this setting to provide database compatibility. There is no default value.
Data Source	Data source properties: Database Interface Designer Both this and the Runtime settings are platform-specific settings. On other platforms, you may have different settings. Again, see the platform-specific adapter reference guide or the context-sensitive help for more setting-specific information. This is the name of the data source used to access database information for design-time (pre-production or testing) purposes.
Runtime	This is the name of the data source used to access database information for run-time (map execution) purposes.
Security	Security properties:
User ID	

This is the user ID used to access the database. If you do not know this information, contact your database administrator. There is no default value.
Password

This is the password used to access the database. If you do not know this information, contact your database administrator. There is no default value.

To define a database:

Procedure

1. From the Navigator, select Databases under the MDQ file.
2. From the Database menu, select New.
or
Right-click the Databases object in the Navigator and select New.
The Database Definition window appears.
3. Enter information for the remaining settings as desired.
4. Click OK.
The new database name appears in the Navigator next to the database icon.

Defining a query

About this task

After you have opened an MDQ file in the **Database Interface Designer** window and a database has been defined, you can specify queries. To use a query as a source, define the query by assigning it a name and entering either the SQL SELECT statement or the stored procedure invocation statement.

To define a query:

Procedure

1. In the Navigator, select Queries under the database icon for which you want to add a query.
2. From the Query menu, select New.
The New Query dialog appears.
3. In the Name field, enter a name for your query. Use this name to reference this query using the Map Designer or you can use it in a data source override execution command.
4. In the Query text box, enter an SQL statement that defines how data should be retrieved from the database.
If your SELECT statement includes table names with spaces, you must enclose them with either single or double quotation marks.
5. Click OK.
The new query appears in the Navigator next to the query icon.

Editing or deleting a database or query

About this task

The following procedures describe how to edit and delete a database or a query, respectively.

To edit a database or query

Procedure

1. In the Navigator, select the name of the database or query to be edited.
2. From the Database menu (or Query menu as appropriate), select Edit.
or
In the Navigator, right-click the database or query icon and select Edit.
3. Make any necessary changes and click OK.

To delete a database or query

Procedure

1. In the Navigator, highlight the database(s) or one or more queries you want to delete.
2. From the Database menu (or Query menu as appropriate), select Delete.
or
In the Navigator, right-click the database or query icon(s) and select Delete.
3. To confirm the deletion, click Yes.

Understanding the MDQ XML format

The database/query (MDQ) file is a configuration file that is used by IBM Transformation Extender when accessing databases using database adapters. Some of the information contained within an MDQ file could include the following:

- database logical names (to distinguish among settings for the databases defined in this file)
 - database adapter types
 - database connection parameters
 - user names and passwords
 - information about tables
 - queries and stored procedures accessed by the adapters
 - primary keys for the tables (used for the database updates)
 - event trigger information for the Launcher
- [XML Schema](#)**
[Saving database/query files](#)
[Processing MDQ files](#)

XML Schema

The MDQ files are well-formed, valid XML documents. MDQ files can be created using the Database Interface Designer (DID) or any other utility capable of creating XML documents. The XML Schema file that is used by IBM Transformation Extender to validate MDQ files is **mdq.xsd**.

Note: Do not modify the content of the **mdq.xsd** file. Unexpected results may occur if you do.

If *install_dir* is the directory into which the IBM Transformation Extender product is installed, the **mdq.xsd** file is copied to the following platform-specific directories as part of that installation process:

Platform	Location
Windows	<i>install_dir</i>
UNIX	<i>install_dir /src</i>

Saving database/query files

Note: If you do not use the Database Interface Designer to create MDQ files, verify that the generated XML files are well-formed and valid for the provided **mdq.xsd** XML Schema.

The following topics contain information related to saving MDQ files from the Database Interface Designer (regardless of whether they were originally created in the Database Interface Designer or using another utility):

- [XML prolog text](#)
- [XML data from XML-generated MDQ files](#)
- [Attribute formatting](#)
- [Password encryption](#)
- [Backup copies](#)

XML prolog text

The XML prolog will always be generated as shown below:

```
<?xml version="1.0" encoding="UTF-8"?>
```

This means that the MDQ files saved from the Database Interface Designer use the UTF-8 encoding scheme.

XML data from XML-generated MDQ files

If you have created this MDQ file outside of the DID, the following information is ignored by the DID when it reads the MDQ file and the information is removed from the MDQ file when it is saved from the DID:

- XML comments
- namespace declarations
- CDATA sections and processing instructions

The content from the CDATA sections is preserved. However, it is saved using the XML entity references.

Attribute formatting

Attribute values are always enclosed in double quotes.

Password encryption

The XML schema provides two different password elements:

- `PASSWORD.literal`
- `PASSWORD.encrypted`

In an MDQ file that has not been saved in the Database Interface Designer, only the `PASSWORD.literal` value can be specified, which is not encrypted. When the file is saved in the Database Interface Designer, the password will be encrypted (using the `PASSWORD.encrypted` element).

Backup copies

If the MDQ file was created (and saved) in an earlier version of the DID and when that file is edited and saved from the DID, this newly saved MDQ file is in the current version. In addition, a backup copy of the MDQ file is created for the older version using the ODQ filename extension (instead of MDQ).

Processing MDQ files

The MDQ files can be processed either at runtime or design-time, depending upon how (and where) the files are specified.

- [Runtime execution](#)
- [Design-time execution](#)

Runtime execution

If the `-MDQ` adapter command is used along with the `DBLOOKUP`, `DBQUERY`, `GET` or `PUT` map functions or if it is specified as an adapter command in the database cards, the MDQ file is parsed at runtime.

Design-time execution

If the MDQ file name is specified in the `DatabaseQueryFile` setting of a map card, the MDQ file is parsed when the map is compiled, which is during design time for the map, not at runtime.

Comparing database/query files

Database/query (MDQ) files contain definitions of one or more databases, as well as queries, stored procedures, tables, queues, and variables. The file differentiation process identifies MDQ file differences first at the database-level, and then at the query-, stored procedure-, table-, and variable-levels. See [Creating a database/query file comparison](#).

After you have performed a comparison between two MDQ files, color-coding is used to highlight what is unique between the two:

- Differences are highlighted in red.
- New information is highlighted in blue.

For more information, refer to [Viewing database/query file differences](#).

- [Criteria for analyzing differences](#)
- [Creating a database/query file comparison](#)
- [Viewing database/query file differences](#)

Criteria for analyzing differences

The criteria for analyzing differences for the different entities is as follows:

- MDQ file
 - Any of the database definitions are different.
 - Any of the variables are different.
 - Any of the objects exist in one, but not the other, MDQ file.
- database definition
 - Any of the database definition settings are different.
 - Any of the syntax of the queries, column overrides of the stored procedures, column overrides or update keys of the tables, or any of the values of the variables are different.

- query
 - The syntax of the queries is different.
 - Column overrides of the queries are different.
 - stored procedure
 - Column overrides for the stored procedures are different.
 - table
 - Column overrides for the tables are different.
 - Update keys for the tables are different.
 - variable
 - Values for the variables are different.
-

Creating a database/query file comparison

About this task

Use this procedure to compare database/query file differences between two selected MDQ files. After you have created the comparison, you can view the results.

To create an MDQ file comparison:

Procedure

1. From the File menu, select Database Query File Differences.
The Select First File dialog appears.
 2. Navigate your file system and select the first MDQ file to be compared.
 3. Click Open.
The Select Second File dialog appears.
 4. Navigate your file system and select the second MDQ source file to be compared.
 5. Click Open.
The progress of the comparison is shown briefly in the Database Source File Differences Analysis dialog. When the analysis has completed, the Database Differences window appears.
-

Viewing database/query file differences

About this task

Note: The Database Differences window should already appear. If you do not see it, you must create the file comparison as described in [Creating a database/query file comparison](#).

Differences are indicated using color-coded text:

- Differences are **red**.
- Additions or new information items are **blue**.

To view MDQ file differences:

Procedure

1. Place current focus in the Database Differences window.
 2. Resize any of the four panes as needed.
 3. Select any database, query, stored procedure, table, or variable to view the differences.
The settings display in the two lower windowpanes, depending upon what you select.
 4. Press F8 or F7 to view the next or previous difference, respectively.
In the Database Differences window, you can view all of the settings for the selected element in the compared MDQ file, even though differences may not exist.
- [Database/query file difference results](#)
 - [Database definition setting differences](#)
 - [Query differences](#)
 - [Variable differences](#)
 - [Unique database information](#)
 - [Stored procedure differences](#)
 - [Table differences](#)
-

Database/query file difference results

When database/query source files are compared, they are considered different if any of the database definition settings are different, any of the variables are different, or any of the objects exist in one MDQ file and not the other. Text color is used to provide a visual distinction. see the table below as a guide:

Condition	Upper Panes	Lower Panes
Same name and content	black text	black text on tab, only one pane at a time
Same name, different content	red text	red text on tab, both panes appear, unique text in red

Condition	Upper Panes	Lower Panes
Different name and content	blue text	red text on tab, only one pane at a time, unique text in green (for database settings only, rest have unique text in blue)

Database definition setting differences

If any database definition settings of either of the two compared MDQ files are different, both database names appear in red letters in the Database Differences window.

The **Database Properties** tab in each lower windowpane also appears in red letters. (If there were no differences, this tab would appear in black letters.)

Query differences

If there are differences between the two compared MDQ files with regard to query syntax or column overrides, the name of the query with differences appears in red letters in the Database Differences window. If the query in each MDQ file is uniquely named, each query name appears in blue letters.

If each MDQ file has a query with the same name, but containing unique information, the **Query** and/or **Column Overrides** tabs in each lower windowpane appear in red letters. (If there were no differences, these tabs would appear in black letters and you would only see one display in the bottom pane at a time, depending upon where your cursor is in the upper pane.) When the tabs are red, you can go to each display and see the differences indicated in red letters.

Click the **Query** and **Column Overrides** tabs in each pane to view the differences.

Variable differences

If the variables of the two compared MDQ files have the same name, but different settings, the variable name will appear in red letters in the upper pane and both variables will appear in the bottom panes with the tab letters also in red letters. In the bottom panes, the unique value(s) will appear in red letters. If the variable names of either of the two compared MDQ files are different, the name of each variable appears in blue letters in the Database Differences window. If the variables are exactly the same, the name of the variable in both the upper and lower pane and the **Variables** tab would all appear in black letters.

Click the **Variables** tab in each pane to view the differences.

Unique database information

If the databases of the two compared MDQ files have the same name, but different settings, this tab will appear in red letters. Also, both databases will appear in the bottom panes of the Database Differences window. If the database names of either of the two compared MDQ files are different, the name of each database appears in blue letters in the Database Differences window. If the databases are exactly the same, the name of the database in both the upper pane and the **Database Properties** tab would all appear in black letters. In both of these latter scenarios (with the different-named or duplicate databases), you can view only one tab at a time by clicking on the Database icon in the upper (Navigator-like) portion of the window for the specific MDQ file. The resulting display in this tab is a reproduction of the settings of the Database Definition dialog. To see the respective database differences, you must click alternately on each database icon in each upper half of the Database Differences window.

Stored procedure differences

If the stored procedures of the two compared MDQ files have the same name, but different values, the stored procedure name will appear in red letters in the upper pane and both stored procedures will appear in the bottom panes with the **Column Overrides** tab letters also in red letters. In the bottom panes, the unique value(s) will appear in red letters. If the stored procedure names of either of the two compared MDQ files are different, the name of each stored procedure appears in blue letters in the Database Differences window. If the stored procedures are exactly the same, the name of the stored procedure in the upper pane and the **Column Overrides** tab and columns would all appear in black letters.

The columns on this tab (**Column Name**, **Column Length**, **Presentation**, and **Interpret As**) correspond to fields and list names in the Column Datatype Specification dialog.

Table differences

If the tables of the two compared MDQ files have the same name, but different values for column overrides or update key definitions, the table name will appear in red letters in the upper pane and both tables will appear in the bottom panes with the **Column Overrides** and/or **Update Keys** tab letters also in red letters. In the bottom panes, the unique value(s) will appear in red letters. If the table names of either of the two compared MDQ files are different, the name of each table appears in blue letters in the Database Differences window. If the tables are exactly the same, the name of the table in the upper pane and the **Column Overrides** and Update Keys tabs and related information would all appear in black letters.

Click the **Column Overrides** and **Update Keys** tabs in each pane to view the differences.

The columns on the **Column Overrides** tab (**Column Name**, **Column Length**, **Presentation**, and **Interpret As**) correspond to fields and list names in the Column Datatype Specification dialog.

Defining variables in SQL statements

Elements of SQL statements can be executed as map sources that are determined at runtime. Use the Database Interface Designer to define a statement variable with a pseudo value in an SQL statement and then pass the actual value on the command line at runtime. This technique is beneficial when using the `RUN` function because it allows one map to modify the SQL statement of another map or to build, potentially, an entire SQL statement.

When you generate type trees using the Database Interface Designer, a substitution value must be entered for each variable to ensure that the syntax of the SQL statement is valid. The Database Interface Designer provides a facility for specifying a value for these variables; however, the value you enter for a variable in the Database Interface Designer does not have to be the same value passed at runtime. (Any value can be passed.)

- [Defining a query with variables](#)
 - [Specifying the values at runtime](#)
-

Defining a query with variables

About this task

In the Database Interface Designer Query dialog, variables can be specified in SQL statements as literals enclosed in pound sign (#) characters. For example, you might enter a statement that defines a variable named `ID`:

```
select * from BigTable where Identifier = #ID#
```

Because the value of a variable may be a text string, you can also create larger elements of the statement variable. For example:

```
select * from BigTable where #WhereClause#
```

When you define variables in a query, the Database Interface Designer automatically detects the presence of the variables in the statement and lists each variable in the Navigator, along with a variable icon.

Use the Define Variables dialog to enter the variable values. Note that you cannot generate a type tree for the query until you have specified a value for *each* variable it contains. Also, if the variable you are defining in the Define Variables dialog is a text string, you must enclose the value in single quotation marks.

The pseudo values specified using the Define Variables dialog are used in the Database Interface Designer only when accessing the database to generate the type tree. They are not used when executing a map.

To specify values in the Define Variables dialog

Procedure

1. In the Navigator, select the variable to which you want to add one or more values.
2. From the Query menu, select Define Variables.
The Define Variables dialog appears, listing all the variables in the MDQ file.
3. In the Value field, enter a value for each variable and click OK.
The pseudo values are stored in the MDQ file and can only be changed in the Define Variables dialog. Also, if the variable you are defining is a text string, you must enclose the value in single quotation marks.

To delete a variable

About this task

When a variable is removed from a query statement that was previously defined in the Database Interface Designer, the variable is automatically deleted from the Navigator and no longer displays.

Specifying the values at runtime

Even though you have specified pseudo values for the defined variables using the Database Interface Designer, they are only substitution values that are not referenced at runtime. Values *must* be provided for the variables at runtime; otherwise, the SQL statement will be syntactically invalid.

The correct value for each variable is specified at run time using the Variable adapter command (`-VAR`) in the Input Source - Override execution command (`-ID`). For more information about how to specify values for these variables at runtime, see the Resource Adapters documentation.

Generating type trees

Use the Database Interface Designer to generate a type tree from a table, query, stored procedure, or message queue associated with a particular database. A type tree generated using the Database Interface Designer contains a type for the table, query, stored procedure, or message, as well as a row type and types for each column in that table, query, stored procedure, or message.

As an alternative, use **mtsmaker** on non Windows-based platforms to generate type trees when you cannot use the Database Interface Designer. For more information about using **mtsmaker**, see the Resource Adapters documentation.

- [From a table or view](#)
- [From a stored procedure](#)
- [From a message queue \(Oracle AQ\)](#)
- [From a query](#)

From a table or view

About this task

When you generate a type tree for a table or view, information from your database and from the Database Interface Designer is used to create a type tree file (MTT). If you change your database table definition or edit the database definition in the Database Interface Designer, you must generate a new type tree to reflect the new information.

Generated type trees may differ when generated using different database adapters. For example, a date field may be represented differently by different databases. For this reason, use the same adapter type that will be used at run time to generate the type tree with the Database Interface Designer. Or, if you use ODBC, use the Column Attributes Override dialog that is described in the Resource Adapters documentation.

To define the type tree to generate from a table or view, use the Generate Type Tree from Tables dialog or the Generate Type Tree from Views dialog. The following table contains fields and descriptions.

Database Name	This is the name of the database from which you want to generate a type tree. (This name is automatically inserted based upon your selection in the Navigator.)						
Tables/Views	This list displays all of the tables, views, or synonyms that have been defined in this database. Select one or more from this list. Note: If the table, view, or synonym you want is not displayed in this list, add it by right-clicking in the list, selecting Insert, and typing the desired name.						
File name	This field displays a system-generated file name based upon the database selection in the Navigator. The default value is the database name with a type tree extension (MTT) and a default path. To change any of this information, click the browse button.						
Overwrite file	This check box determines whether you want to replace the existing type tree file (MTT) upon saving it. The default value is disabled.						
Override type	This check box determines whether to replace existing types with matching, newly generated types and to add any non-matching new types to the type tree file (MTT). This is useful when you want to add types for additional tables into an existing type tree. The default value is enabled.						
Row group format	<p>This list displays options to determine the row format in the generated type tree. The default value is Delimited.</p> <table border="1"><tr><th>Option</th><th>Description</th></tr><tr><td>Delimited</td><td>Select this option to create rows delimited with a pipe character () or some other user-defined selection. This is the preferred option if you are using large column widths where the values may be smaller than the width of the columns. The pipe character is the default selection.</td></tr><tr><td>Fixed</td><td>Select this option to create rows of fixed size in the generated type tree. There are certain situations in which you would want to select Fixed:<ul style="list-style-type: none">• The query result contains a large number of rows where each row consists of small, fixed-length fields. In this situation, delimiters are unnecessary and they also consume memory.• You need to append fields in the output. This may be true if multiple fields have some meaning when concatenated together.</td></tr></table>	Option	Description	Delimited	Select this option to create rows delimited with a pipe character () or some other user-defined selection. This is the preferred option if you are using large column widths where the values may be smaller than the width of the columns. The pipe character is the default selection.	Fixed	Select this option to create rows of fixed size in the generated type tree. There are certain situations in which you would want to select Fixed: <ul style="list-style-type: none">• The query result contains a large number of rows where each row consists of small, fixed-length fields. In this situation, delimiters are unnecessary and they also consume memory.• You need to append fields in the output. This may be true if multiple fields have some meaning when concatenated together.
Option	Description						
Delimited	Select this option to create rows delimited with a pipe character () or some other user-defined selection. This is the preferred option if you are using large column widths where the values may be smaller than the width of the columns. The pipe character is the default selection.						
Fixed	Select this option to create rows of fixed size in the generated type tree. There are certain situations in which you would want to select Fixed: <ul style="list-style-type: none">• The query result contains a large number of rows where each row consists of small, fixed-length fields. In this situation, delimiters are unnecessary and they also consume memory.• You need to append fields in the output. This may be true if multiple fields have some meaning when concatenated together.						
Delimiter	This is the value for the delimiter to be used when generating this type tree. For a list of available symbols, click the browse button. The Symbols dialog appears, allowing you to select the desired value. The default value is a pipe character ().						
Terminator	This is the value for the terminator to be used when generating this type tree. For a list of available symbols, click the browse button. The Symbols dialog appears, allowing you to select the desired value. The default value is <CR><LF>.						
Release	This is the value for the release character to be used when generating this type tree. For a list of available symbols, click the browse button. The Symbols dialog appears, allowing you to select the desired value. The default value is an exclamation point (!).						
Override Column Definitions	(For ODBC only) This check box indicates whether the database driver (with check box disabled) or a command line override (with check box enabled) is used to interpret columns. The default value is disabled. For more information about this field, see the adapter-specific documentation in each adapter reference guide.						
Generate type trees in 1.4.0 format	(For Oracle adapter only) This check box determines whether you can generate type trees that are compatible with maps for Version 1.4.0. You might enable this functionality if you have legacy 1.4.0 maps that you want to maintain for some reason. The default value is disabled. For more information about this field, refer to adapter-specific documentation in each adapter reference guide.						
Represent date/time columns as text items	If you are generating new type trees, make sure this check box is disabled. The default is disabled. Because of adapter-specific differences in date and time formats, see the adapter-specific instructions in each adapter reference guide.						
Treat text item as	This list displays the languages available in which to handle text items. The default value is Western.						

To generate a type tree from a table or view:

Procedure

1. In the Navigator, highlight the icon of the database containing the table from which you want to generate a type tree.
2. From the Database menu, select Generate Tree From > Table.
or

In the Navigator, right-click the database icon and select Generate Tree From > Table.

The Generate Type Tree from Tables dialog appears, including a list of tables and views.

3. If you need to add a table because it is not displayed in the Tables/Views list, you must insert it using the instructions provided in the Note below.
or

If you see one or more tables in the Tables/Views list that you want to select, if it is one; select it. If it is multiple tables, press either Shift or Ctrl and click the desired table.

If the table or view you want is not displayed in this list, add it by right-clicking in the list, selecting Insert, and typing the desired name.

4. To create a new type tree, in the File name field, specify the path and name (with an .mtt extension).
or

Click the browse button to select a type tree from the Save As dialog. Highlight it and click OK.

5. Specify the remaining options as desired. Refer to context-sensitive help for field-specific information.

6. Click Generate.

The Database Interface Designer and the Type Tree Maker produce a type tree corresponding to the selected database table(s) or view(s).

Results

The generated type tree is represented in the Navigator with the table/view icon next to the name.

From a stored procedure

About this task

Use the Database Interface Designer to generate type trees from the parameters of stored procedures.

An example of the Generate Type Trees from Stored Procedures dialog follows, along with a table describing its fields and their descriptions.

Database Name	(Display only) This is the name of the database from which you want to generate a type tree. (This name is automatically inserted based upon your selection in the Navigator.)						
Stored Procedures	This list displays all of the stored procedures that have been defined in this database. Select one or more from this list. The list of stored procedures includes stand-alone procedures by default. (Procedures that are parts of a package are not listed.) If you need a procedure that is part of a package (whether this functionality is available is dependent upon your database), you may need to insert it into this list before selecting it. To insert a packaged stored procedure, right-click in the list, select Insert, and type the desired name. Use the following format: <i>schema . package . procedure</i> .						
File name	This field displays a system-generated file name based upon the database selection in the Navigator. The default value is the database name with a type tree file extension (.mtt) and a default path. To change any of this information, click the browse button.						
Overwrite file	This check box determines whether you want to replace the existing type tree file (MTT) upon saving it. The default value is disabled.						
Override type	This check box determines whether to replace existing types with matching, newly generated types and to add any non-matching new types to the type tree file (MTT). This is useful when you want to add types for additional stored procedures into an existing type tree. The default value is enabled.						
Row group format	<p>This list displays options to determine the row format in the generated type tree. The default value is Delimited.</p> <table><thead><tr><th>Option</th><th>Description</th></tr></thead><tbody><tr><td>Delimited</td><td>Select this option to create rows delimited with a pipe character () or some other user-defined selection. This is the preferred option if you are using large column widths where the values may be smaller than the width of the columns. The pipe character is the default selection.</td></tr><tr><td>Fixed</td><td>Select this option to create rows of fixed size in the generated type tree. There are certain situations in which you would want to select Fixed.</td></tr></tbody></table> <ul style="list-style-type: none">The query result contains a large number of rows where each row consists of small, fixed-length fields. In this situation, delimiters are unnecessary and they also consume memory.You need to append fields in the output. This may be true if multiple fields have some meaning when concatenated together.	Option	Description	Delimited	Select this option to create rows delimited with a pipe character () or some other user-defined selection. This is the preferred option if you are using large column widths where the values may be smaller than the width of the columns. The pipe character is the default selection.	Fixed	Select this option to create rows of fixed size in the generated type tree. There are certain situations in which you would want to select Fixed.
Option	Description						
Delimited	Select this option to create rows delimited with a pipe character () or some other user-defined selection. This is the preferred option if you are using large column widths where the values may be smaller than the width of the columns. The pipe character is the default selection.						
Fixed	Select this option to create rows of fixed size in the generated type tree. There are certain situations in which you would want to select Fixed.						
Delimiter	This is the value for the delimiter to be used when generating this type tree. For a list of available symbols, click the browse button. The Symbols dialog appears, allowing you to select the desired value. The default value is a pipe character ().						
Terminator	This is the value for the terminator to be used when generating this type tree. For a list of available symbols, click the browse button. The Symbols dialog appears, allowing you to select the desired value. The default value is <CR><LF>.						
Release	This is the value for the release character to be used when generating this type tree. For a list of available symbols, click the browse button. The Symbols dialog appears, allowing you to select the desired value. The default value is an exclamation point (!).						
Override Column Definitions	(For ODBC only) This check box indicates whether the database driver (with the check box disabled) or a command line override (with the check box enabled) is used to interpret columns. The default value is disabled. For more information about this field, see the adapter-specific documentation in each adapter reference guide.						
Generate type trees in 1.4.0 format	(For Oracle adapter only) This check box determines whether you can generate type trees that are compatible with Version 1.4.0 maps. You might enable this functionality if you have legacy 1.4.0 maps that you want to maintain for some reason. The default value is disabled. For more information about this field, refer to adapter-specific documentation in each adapter reference guide.						
Represent date/time columns as text items	This check box determines whether to automatically format this information in its database-specific date and time format (enabled) or as a text string (disabled). If you enable this check box (generating date and time as a text string), you may have to use either the TEXTTODATE or TEXTTOTIME function in a map rule to convert the text string to the database-specific date and time format. If you are generating new type trees, make sure this check box is disabled. The default is disabled. Note: Because of adapter-specific differences in date and time formats, see the adapter-specific instructions in each adapter reference guide.						
Treat text item as	This list displays the languages available in which to handle text items. The default value is Western .						

To generate a type tree from a stored procedure:

Procedure

- In the Navigator, highlight the icon of the database containing one or more stored procedures from which you want to generate a type tree.
- From the Database menu, click Generate Tree From > Procedure.

or

In the Navigator, right-click the database and select Generate Tree From > Procedure.

The Generate Type Tree from Stored Procedures dialog appears, including a list of stored procedures.

3. If you need to add a stored procedure because it is not displayed in the Stored Procedures list, you must insert it following the instructions provided in the Note below.
or

If you see one or more stored procedures in the Stored Procedures list that you want to select, if it is one, select it; if it is multiple procedures, press either Shift or Ctrl and click the desired stored procedures.

Note: The list of stored procedures includes stand-alone procedures by default. (Procedures that are parts of a package are not listed.) If you need a procedure that is part of a package (whether this functionality is available is dependent upon your database), you may need to insert it into this list before selecting it. To insert a packaged stored procedure, right-click in the list, select Insert, and type the desired name. Use the following format: schema.package.procedure.

4. To create a new type tree, in the File name field, specify the path and name (with an .mtt extension).
or

Click the browse button to select a type tree from the Save As dialog. Highlight it and click OK.

5. Specify the remaining options as desired. Refer to context-sensitive help for field-specific information.

6. Click Generate.

The Database Interface Designer and Type Tree Maker produce a type tree corresponding to the selected stored procedure(s).

Results

The generated type tree is represented in the Navigator with the stored procedure icon next to the name.

From a message queue (Oracle AQ)

About this task

Use the Database Interface Designer to generate type trees for a message on an Oracle AQ message queue. An example of the Generate Type Tree from Queues dialog follows, along with a table describing its fields and their descriptions.

Database Name	This is the name of the database from which you want to generate a type tree. (This name is automatically inserted based upon your selection in the Navigator.)						
Tables/Views	This list displays all of the queues that have been defined in this database. Select one or more from this list. If the queue you want is not displayed in this list, add it by right-clicking in the list, selecting Insert, and typing the desired name.						
File name	This field displays a system-generated file name based upon the database selection in the Navigator. The default value is the database name with a type tree extension (.mtt) and a default path. To change any of this information, click the browse button.						
Overwrite file	This check box determines whether you want to replace the existing type tree file (MTT) upon saving it. The default value is disabled.						
Override type	This check box determines whether to replace existing types with matching, newly generated types and to add any non-matching new types to the type tree file (MTT). This is useful when you want to add types for additional stored procedures into an existing type tree. The default value is enabled.						
Row group format	This list displays options to determine the row format in the generated type tree. The default value is Delimited. <table border="1"><thead><tr><th>Option</th><th>Description</th></tr></thead><tbody><tr><td>Delimited</td><td>Select this option to create rows delimited with a pipe character () or some other user-defined selection. This is the preferred option if you are using large column widths where the values may be smaller than the width of the columns. The pipe character is the default selection.</td></tr><tr><td>Fixed</td><td>Select this option to create rows of fixed size in the generated type tree. There are certain situations in which you would want to select Fixed.</td></tr></tbody></table>	Option	Description	Delimited	Select this option to create rows delimited with a pipe character () or some other user-defined selection. This is the preferred option if you are using large column widths where the values may be smaller than the width of the columns. The pipe character is the default selection.	Fixed	Select this option to create rows of fixed size in the generated type tree. There are certain situations in which you would want to select Fixed.
Option	Description						
Delimited	Select this option to create rows delimited with a pipe character () or some other user-defined selection. This is the preferred option if you are using large column widths where the values may be smaller than the width of the columns. The pipe character is the default selection.						
Fixed	Select this option to create rows of fixed size in the generated type tree. There are certain situations in which you would want to select Fixed.						
Delimiter	This is the value for the delimiter to be used when generating this type tree. For a list of available symbols, click the browse button. The Symbols dialog appears, allowing you to select the desired value. The default value is a pipe character ().						
Terminator	This is the value for the terminator to be used when generating this type tree. For a list of available symbols, click the browse button. The Symbols dialog appears, allowing you to select the desired value. The default value is <CR><LF>.						
Release	This is the value for the release character to be used when generating this type tree. For a list of available symbols, click the browse button. The Symbols dialog appears, allowing you to select the desired value. The default value is an exclamation point (!).						
Override Column Definitions	(For ODBC only) This field is not available for message queues.						
Generate type trees in 1.4.0 format	(For Oracle adapter only) This check box determines whether you can generate type trees that are compatible with Version 1.4.0 maps. You might enable this functionality if you have legacy 1.4.0 maps that you want to maintain for some reason. The default value is disabled. For more information about this field, refer to adapter-specific documentation in each adapter reference guide.						
Represent date/time columns as text items	This check box determines whether to automatically format this information as a text string. By default, the checkbox is deselected, and the information is formatted in its database-specific date and time format. If you enable this check box (generating date and time as a text string), you might have to use either the TEXTTODATE or TEXTTOTIME function in a map rule to convert the text string to the database-specific date and time format. If you are generating new type trees, make sure this check box is disabled. Note: Because of adapter-specific differences in date and time formats, see the adapter-specific instructions in each adapter reference guide.						
Treat text item as	This list displays the languages available in which to handle text items. The default value is Western.						

To generate a type tree from a queue:

Procedure

1. In the Navigator, select the icon of the database from which you want to generate a type tree.
2. From the Database menu, click Generate Tree From > Queue.
The Generate Type Tree from Queues dialog appears, including a list of queues.
3. If you need to add a queue because it is not displayed in the Tables/Views list, you must insert it following the information in the Note below.
or

If you see one or more queues in the Tables/Views list that you want to select, if it is one, select it. If it is multiple queues, press either Shift or Ctrl and click the desired queues.

4. To create a new type tree, in the File name field, specify the path and name (with an .mtt extension).
or

Click the browse button to select a type tree from the Save As dialog. Highlight it and click OK.

5. Specify the remaining options as needed.
6. Click Generate.

The Database Interface Designer and the Type Tree Maker produce a type tree corresponding to the selected queue(s).

Note: Type trees generated from queues are not graphically represented in the Navigator.

From a query

About this task

When generating a type tree from a query, information is used from your database and from the Database Interface Designer to create the type tree. If you change the definition of your query in the Database Interface Designer or edit that database definition, you must generate a new type tree to reflect the new information.

When attempting to generate a type tree, if your query is incorrectly specified, an error message from your database is received and a message displays, indicating that the column definitions for that query could not be accessed. In this scenario, a type tree is not generated.

Generating a type tree provides a way to test the syntax of your select statement. If you can successfully generate a type tree from a query, you know that the query has valid syntax.

An example of the Generate Type Trees From Query dialog follows, along with a table describing its fields and their descriptions.

Field	Description
File name	This field displays a system-generated file name based upon the database selection in the Navigator. The default value is the database name with a type tree extension (.mtt) and a default path. To change any of this information, click the browse button.
Overwrite file	This check box determines whether you want to replace the existing type tree file (MTT) upon saving it. The default value is disabled.
Override type	This check box determines whether to replace existing types with matching, newly generated types and to add any non-matching new types to the type tree file (MTT). This is useful when you want to add types for additional stored procedures into an existing type tree. The default value is enabled.
Row group format	This list displays options to determine the row format in the generated type tree. The default value is Delimited .
Delimited	Select this option to create rows delimited with a pipe character () or some other user-defined selection. This is the preferred option if you are using large column widths where the values may be smaller than the width of the columns. The pipe character is the default selection.
Fixed	Select this option to create rows of fixed size in the generated type tree. There are certain situations in which you would want to select Fixed. <ul style="list-style-type: none">• The query result contains a large number of rows where each row consists of small, fixed-length fields. In this situation, delimiters are unnecessary and they also consume memory.• You need to append fields in the output. This may be true if multiple fields have some meaning when concatenated together.
Delimiter	This is the value for the delimiter to be used when generating this type tree. For a list of available symbols, click the browse button. The Symbols dialog appears, allowing you to select the desired value. The default value is a pipe character ().
Terminator	This is the value for the terminator to be used when generating this type tree. For a list of available symbols, click the browse button. The Symbols dialog appears, allowing you to select the desired value. The default value is <CR><LF>.
Release	This is the value for the release character to be used when generating this type tree. For a list of available symbols, click the browse button. The Symbols dialog appears, allowing you to select the desired value. The default value is an exclamation point (!).
Override Column Definitions	(For ODBC only) This check box indicates whether the database driver (with the check box disabled) or a command line override (with the check box enabled) is used to interpret columns. The default value is disabled. For more information about this field, see the adapter-specific documentation in each adapter reference guide.
Generate type trees in 1.4.0 format	(For Oracle adapter only) This check box determines whether you can generate type trees that are compatible with Version 1.4.0 maps. You might enable this functionality if you have legacy 1.4.0 maps that you want to maintain for some reason. The default value is disabled. For more information about this field, refer to adapter-specific documentation in each adapter reference guide.
Represent date/time columns as text items	If you enable this check box (generating date and time as a text string), you may have to use either the TEXTTODATE or TEXTTOTIME function in a map rule to convert the text string to the database-specific date and time format. If you are generating new type trees, make sure this check box is disabled. The default is disabled. Note: Because of adapter-specific differences in date and time formats, see the adapter-specific instructions in each adapter reference guide.
Treat text item as	This list displays the languages available in which to handle text items. The default value is Western .

To generate a type tree from a query:

Procedure

1. In the Navigator, highlight the icon of the query from which you want to generate a type tree.

2. From the Query menu, select Generate Tree.
or
In the Navigator, right-click the query and select Generate Tree.
- The Generate Type Tree from Query dialog appears, displaying the name of the selected type tree.
3. To create a new type tree, in the File name field, specify the path and name (with an .mtt extension).
or
Click the browse button to select a type tree from the Save As dialog. Highlight it and click OK.
4. Specify the remaining options as desired. Refer to context-sensitive help for field-specific information.
5. Click OK.
The Database Interface Designer and the Type Tree Maker produce a type tree corresponding to the selected database query.

Results

The generated type tree is represented in the Navigator with the query icon displayed next to the name.

Printing reports

About this task

You can print reports including detailed information about each specified database and query contained in an MDQ file. Reports contain information about the databases and queries that you designate. They can also include the time when the report was printed, the full path name of the MDQ file, the database definition, and definitions of each query and stored procedure.

You can only specify one database at a time.

To print a report:

Procedure

1. Highlight the database within an MDQ file for which you want to print the information.
2. From the File menu, select Print.
or
In the Navigator, right-click the icon of the database and select Print.
- The Print dialog appears.
3. To print all of the entities displayed, skip to the next step.
or
To select certain entities to not be printed, disable the appropriate check box next to the entity name.
4. To use the default font for the report, skip to Step 6.
or
To change the font for the report, click Font.
- The Font dialog appears.
5. Change the settings as desired and click OK.
The Font dialog is closed and you are returned to the Print dialog.
6. Click OK.
The Print dialog appears.
7. Make any changes as desired and click OK.
The report is printed.

Database Interface Designer trace files

The Database Interface Designer trace is a facility that produces information about accessing your database from the Database Interface Designer and then logs this information to a file. This can be useful in determining problems encountered when generating type trees or setting update keys using the Database Interface Designer.

If trace is enabled, a Database Interface Designer trace file (.dbl) is automatically created when you generate a type tree. This trace file is placed in the same directory as the MDQ file. The newly created Database Interface Designer trace file is named using the full name of the MDQ file with a .dbl extension. For example, if your MDQ file is named **Orders.mdq**, the trace file is named **Orders.dbl** and is located in the same directory. If the MDQ file has been newly created, the trace file is named its assigned name (**Database_QueryFile n .dbl**, where **n** represents the assigned sequential number) and resides in the directory in which the Database Interface Designer is installed.

- [Database Interface Designer trace](#)
- [Viewing Database Interface Designer trace files](#)
- [Finding text in trace files](#)

Database Interface Designer trace

About this task

You can enable or disable trace for any MDQ file listed in the Navigator.

To enable or disable trace:

Note: Because this is a toggle command, use the same procedure for both enabling and disabling trace.

Procedure

1. In the Navigator, highlight the MDQ file for which you want to enable a trace.
2. From the File menu, select Trace.

Viewing Database Interface Designer trace files

About this task

After trace has been enabled and a Database Interface Designer trace file has been generated, the contents of this file can be viewed in a trace window in the Database Interface Designer.

To view trace files:

Procedure

1. In the Navigator, highlight the MDQ file for which you want to view the results of a trace.
2. From the View menu, select Trace File.
A trace window appears, showing the results of the last trace.
Note: If a trace file has not yet been generated, a blank window appears.
3. To define how you want to view multiple trace windows simultaneously, from the Window menu, enable either Cascade, Tile Horizontally, Tile Vertically or Arrange Icons.

Results

In addition to the Database Interface Designer trace that is applicable only when working in the Database Interface Designer environment, you can also generate an additional database trace file that records activities occurring during map execution.

Finding text in trace files

About this task

Use the **Find** command to locate specific text in a Database Interface Designer trace window. This is helpful when analyzing errors or searching for specific information. The **Find** command is implemented as a result of values entered in the Find dialog. An example of this dialog follows, along with a table describing its fields and their descriptions.

Find what	Enter the text upon which to search. The default value is blank.						
Match whole word only	This check box determines whether to use the value entered in the Find what field to match only whole words against in the trace file. (For example, if this was enabled and text was your find value, texts would not be a match.) The default value is disabled.						
Match case	This check box determines whether to use the value entered in the Find what field to exactly match its case in the trace file. (For example, if this was enabled and Text was your find value, text would not be a match.) The default value is disabled.						
Direction	Select the option to determine the direction of the find, that is whether to start at the beginning point and search backward (Up) or forward (Down). The default selection is Down. <table><thead><tr><th>Option</th><th>Description</th></tr></thead><tbody><tr><td>Up</td><td>Select this option to begin the find at this point and go backwards in the trace file. The default value for this is cleared.</td></tr><tr><td>Down</td><td>Select this option to begin the find at this point and go forward in the trace file. The default value for this is selected.</td></tr></tbody></table>	Option	Description	Up	Select this option to begin the find at this point and go backwards in the trace file. The default value for this is cleared.	Down	Select this option to begin the find at this point and go forward in the trace file. The default value for this is selected.
Option	Description						
Up	Select this option to begin the find at this point and go backwards in the trace file. The default value for this is cleared.						
Down	Select this option to begin the find at this point and go forward in the trace file. The default value for this is selected.						

To use the Find command:

Procedure

1. Select the trace window containing the trace file upon which you want to perform a text search.

Note: Ensure the trace window containing the desired trace file is the active window.

or

From the Window menu, select the name of the trace file.

2. From the Edit menu, select Find.

The Find dialog appears.

3. In the Find what field, enter the text you want to find.
 4. Make other selections as desired.
 5. Click Find Next.
If a match is found, the text is highlighted in the file in the trace window. Otherwise, there may be a message asking if you want to continue searching.
 6. Repeat Step 5 until you have finished your search.
 7. Click Cancel.
-

Database type trees

Use the Database Interface Designer or **mtsmaker** to generate a type tree to be used in a map. You can also define your own type tree using the Type Designer. However, for the database adapters to correctly process your database data, your type tree must conform to the format described in this topic. For information about using **mtsmaker** to generate type trees, see the Resource Adapters documentation.

Although you can manually create a type tree for a table or a query using the Type Designer, it is not recommended.

The following examples represent type trees generated by the Database Interface Designer from two queries (one SELECT and one calls a stored procedure), a table, a stored procedure, and a message queue.

As shown in these examples, the type trees generated from tables, views, queries, and stored procedures are very similar, while the type tree generated for message queues differs slightly.

In these example type trees:

- The type tree named **ActvProj.mtt** was created for a query that referenced a Microsoft Access table.
 - The type tree named **Stores.mtt** was created for a table in a Microsoft SQL Server database.
 - Two type trees were created for stored procedures in an Oracle database.
 - The **PymtInfo.mtt** type tree was generated for a query called **GetPaymentInfo** that calls an Oracle stored procedure to be used as a map data source.
 - The **ApplyPmt.mtt** type tree was generated for the **SCOTT.APPLY_PAYMENT** stored procedure to be used as a map output.
 - The type tree named **Paulmsg.mtt** was generated from a message queue on the **PAUL_MSG_QUEUE**.
 - [Table and query type tree structure](#)
 - [Stored procedure type tree structure](#)
 - [Oracle AQ Message type tree structure](#)
-

Table and query type tree structure

Type trees generated for a query and a table closely resemble one another.

Each type tree generated by the Database Interface Designer or **mtsmaker** for a database table or query (or a query that calls a stored procedure) will have the following standard characteristics:

- The root of the generated type tree is named **Data**.
 - The generated type tree contains a category whose name corresponds to any one of the following:
 - the name of the table in the database
 - the name of the query in the MDQ file
 - the name specified using the Category Type (-N) parameter for **mtsmaker**

The types defining the columns and rows returned by the database adapter stem from this type. In the examples, the categories are **ActiveProjects** and **stores**.
 - Stemming from the category named for the table or query are types that define the contents of the table or the results of query execution.
 - Each column in the database table or in the results of the query must have a corresponding item type with a name identical to that of the column in the database. These types stem from an item type (shown in each example) named **Column**.
 - A group type named **Row** represents a single row of data—either the result of a SELECT statement or a row to be inserted/updated in a table or view.
 - An implied group represents a collection of rows reflecting the results of a query, table, or view. This group is named either **DBSelect** for a query or **DBTable** for a table or view (or **DBProcedure** for a stored procedure).
 - [Special characters in type names](#)
 - [Characteristics of the DBSelect or DBTable type](#)
 - [Components and format of the row type](#)
 - [Defining the column type\(s\)](#)
-

Special characters in type names

Some characters used in column names are not valid within type names. The Database Interface Designer replaces these characters when it creates the type name for the column. If you create a type tree without using the Database Interface Designer or **mtsmaker**, you need to replace these characters in the type name.

Again, if you use the Database Interface Designer, these conversions are automatically performed. If you use **mtsmaker**, you must manually convert these characters.

Invalid Character(s)	Conversion to Type Name
space or hyphen (-) example: My_Column_Name	replaced with underscore (_) becomes: My_Column_Name
punctuation (other than ~ # % \ ^ ?) example: ^^Total^Amount	replaced with pound sign (#) becomes: ##Total##Amount
digit (as first character of column name) example: 1st-Game-of-2	prefixed with pound sign (#) becomes: #1st_Game_of_2

Characteristics of the DBSelect or DBTable type

In each type tree generated by the Database Interface Designer or **mtsmaker**, an implied group type represents the results of executing the query or the contents of the table.

- **DBSelect**

The type tree defines a query used as a data source for a map.

- **DBTable**

The type tree defines a table used as a data target for a map.

The **DBSelect** or **DBTable** type is an implied group with a series of row objects as its only component.

Components and format of the row type

As its name implies, the **Row** group represents a row (or tuple). The **Row** type is a group with an explicit format. The syntax of that format is based upon the options specified when the tree was created from the Database Interface Designer or the value passed to **mtsmaker** using the Format **mtsmaker** command (-x).

see the following table for information about specifying the value for the **Row group format** list when generating a type tree for a query or table.

Using

Information

Database Interface Designer

Database/query files

mtsmaker

Resource Adapters documentation

There are two different row group formats:

- delimited row group format
- fixed row group format

You can specify the syntax for each of these when generating the type tree.

- [**Delimited row group format**](#)
- [**Fixed row group format**](#)

Delimited row group format

If you select Delimited as the value in the **Row group format** list in either the Generate Type Tree from Tables dialog, the Generate Type Tree from Query dialog, or the Generate Type Tree from Stored Procedures dialog, the Properties window in the generated type tree indicates that the Group Subclass > Format is an explicit group defined as having a delimited syntax.

In the **Group options** pane, which is located in the dialog for generating the type tree, you can specify the **Delimiter**, **Terminator**, and **Release** setting values to be used in the type tree. The default values are |, None, <CR><LF>, and !, respectively. The **Initiator** value cannot be specified; it is always set to None.

When using the **DBLOOKUP**, **DBQUERY**, **GET** and **PUT** map functions with the database adapter, the adapter will use the default values for the **Delimiter**, **Terminator**, and **Release** settings. This is because the data passed to and from the adapter is, in this case, not represented by the type trees generated in the Database Interface Designer. The adapter will use the default values for the **Delimiter**, **Terminator**, and **Release** settings since it cannot access this information from a type tree.

Each component of **Row** is a type representing a column. These components appear in the component list in the same sequence as they appear in the query or table, as opposed to the type tree, which lists them in alphabetical order. In a delimited **Row**, each of the columns is defined as optional within the **Row** by specifying a range of (0:1) for each column component.

Fixed row group format

If you select Fixed as the value in the **Row group format** list in either the Generate Type Tree from Tables dialog, the Generate Type Tree from Query dialog, or the Generate Type Tree from Stored Procedures dialog, the Properties window in the generated type tree indicates that the **Row** type is an explicit group defined as having a fixed syntax.

In the **Group options** pane, which is located in the dialog for generating the type tree, you can specify the **Terminator** setting values to be used in the type tree. The default value is <CR><LF>.

When using the **DBLOOKUP**, **DBQUERY**, **GET** and **PUT** map functions with the database adapter, the adapter will use the default values for the **Delimiter**, **Terminator**, and **Release** settings. This is because the data passed to and from the adapter is, in this case, not represented by the type trees generated in the Database Interface Designer. The adapter will use the default values for the **Delimiter**, **Terminator**, and **Release** settings since it cannot access this information from a type tree.

Defining the column type(s)

Each named column in the database has a corresponding item type under the **Column** type in the type tree with the same name. For example, if the **DEMO.DEPARTMENT** table in the database has three columns (**DEPARTMENT_ID**, **LOCATION_ID**, and **NAME**), the type tree will have three identically named items (subject to restrictions described in [Special Characters in Type Names](#)) under the **Column** type.

Each item is defined according to the data type and length information returned to the database adapter by the database driver. For more information about the correspondence between database data types and item formats, see the specific adapter reference guides.

The minimum size of item types for columns in a delimited **Row** is 0 unless otherwise specified by the database driver. The minimum size of item types for columns in a fixed **Row** equals the maximum size.

- [Binary column types](#)
 - [Column types for expressions](#)
 - [Specifying column aliases](#)
-

Binary column types

If a database column corresponds to a binary type, your type tree contains additional types so that the binary data can be correctly interpreted. The method used to define binary columns allows a distinction to be made between a **NULL** column and a binary value of 0.

A category called **SizedGroup** is added to the tree. This category has an item subtype called **Sizeof** and group subtypes with the same names as the item types for each binary column.

The following changes are made to the type tree to account for columns representing binary data:

- Two variable length binary items are created as subtypes of the **Column** type to represent the value of the two binary type columns.
- The **Sizeof** item is a subtype of the **SizedGroup** category and is defined as a character integer item.
- Also, as a subtype of the **SizedGroup** category, two groups given the same name as the binary column type are created, each group including two components. The first component of each group is the **Sizeof** item; the second, the variable length binary item (either).
- The formats of those group types defined as subtypes of the **SizedGroup** category are defined in the same way as the format for the **Row** group type.
If the type tree has a delimited **Row** group format, the group types for the binary columns are also defined as delimited groups with the pipe character (|) as an infix delimiter. In this example, the **Sizeof** component is defined as being required in the component list. However, the binary item type is optional (with a range of 0:1).

You cannot generate a fixed **Row** group format type tree for a table or query containing variable length binary data.

If the type tree has a fixed **Row** group format, the group types for the binary columns are also defined as fixed groups. In this situation, both the **Sizeof** component and the binary item type are required.

- In the group type component list, the **Sizeof** component has the **Sized** attribute, indicating that the **Sizeof** item contains the size of the binary item that follows it.
When mapping to a table or view containing binary columns, use the **SIZE** function to assign the size of the binary data to the **Sizeof** data item.
 - The components for the **Row** type are the group types under the **SizedGroup** category for the binary columns.
-

Column types for expressions

Each constant or function specified in a query is given a name based upon the name returned from the database driver to the database adapter. If the database driver does *not* return a name for a column in the results, the item type created for the column is given a name beginning with Expr and is concatenated with a number beginning with 1000 for the first constant or function, 1001 for the second, and so on.

For example, the type tree generated for the following query

```
SELECT MIN(salary), MAX(salary) FROM Employee
```

would have a **Row** with two columns representing the result of the SQL MIN and MAX functions. In the type tree, the items created to represent the results of these functions would be Expr1000 and Expr1001 for the MIN and MAX functions, respectively.

Specifying column aliases

When you want to control the name of the generated item to represent a particular column or expression in a query, use the AS keyword to specify an alias for that column.

For example, you could specify the query as:

```
select min(salary) as min_salary, max(salary) as max_salary from employee
```

which would result in a type tree containing a **Row** with two columns representing the result of the SQL MIN and MAX functions. In the type tree, the items created to represent the results of these functions would be **min_salary** and **max_salary**. An example of this follows.

You can also use the AS keyword to specify an alias for a column having a name containing any of the characters that are invalid in a type name. Or, you can specify an alias for a column having a name longer than 32 bytes in UTF-8 encoding, which is the limit for the size of a type name.

The following example results in a type tree having a **Row** with three columns.

```
Select SomeReallyReallyReallyLongColumnName AS  
SomeLongColumnName,  
    1st-Payment AS Payment#1,  
    2nd-Payment AS Payment#2  
from some_table
```

The items created to represent the results of this query would be **SomeLongColumnName**, **Payment#1**, and **Payment#2**.

Stored procedure type tree structure

The type trees generated by the Database Interface Designer for stored procedures used for outputs from a map are slightly different from the ones generated for tables, queries, and queries calling stored procedures (that are used as the source of data for a map).

Major differences between a type tree for a stored procedure that will be used as output and a type tree for a table that will also be used as output are:

- Whereby table and query type trees have a **Column** item type from which stems all of the individual column types, the type tree for a stored procedure has an **Argument** item type from which stems item types representing each of the arguments passed to the stored procedure.
- Instead of a **Row** group, the type tree for a stored procedure has a **ProcedureCall** type. The **ProcedureCall** group represents the set of arguments passed to the stored procedure for each execution of the procedure.
The **ProcedureCall** group is defined in the same way as a **Row** type for a table or a query in the type tree. Its group format is determined by the value selected in the **Row group format** list in the Generate Type Tree from Tables dialog or the Generate Type Tree from Query dialog, respectively. The selected group format determines the terminator and release characters.
- Similar to the **DBTable** or **DBSelect** types in type trees for tables or queries, the type tree for a stored procedure has a **DBProcedure** type, which is an implied group consisting of a series of **ProcedureCalls**. The stored procedure is called once for each **ProcedureCall** in the **DBProcedure** output.
For information about using stored procedures as a data target, see [Using stored procedures](#). For information about generating type trees for a stored procedure, see [From a stored procedure](#).

Oracle AQ Message type tree structure

The type trees generated by the Database Interface Designer for Oracle AQ messages used as output from a map are different from those generated for tables, queries, or stored procedures. For information about these type trees, see the Oracle AQ Adapter documentation.

Database sources and targets

This topic explains how to use the Map Designer to configure a map having a database source or target. For detailed information about using the Map Designer and defining sources and targets in input and output cards, see the Map Designer documentation.

- [Using a database as a source](#)
- [Using a database as a target](#)
- [Database connections and transactions](#)
- [Updating database tables](#)

Using a database as a source

About this task

After you have used the Database Interface Designer to define a query for a database, you can use that query as an input source.

Note: You can use either a standard SQL SELECT query or a query using a stored procedure. For specific information about using stored procedures, refer to [Using stored procedures](#).

To use a database query as a source:

Procedure

- Define the database using the Database Interface Designer.
 - Define a query to extract data from the database.
 - Generate a type tree for the query.
 - Use the Map Designer to create a map with an input card referencing the query.
- [Defining a database source in the Map Designer](#)
 - [Database GET > Source settings](#)

Defining a database source in the Map Designer

About this task

When you generate the type tree from a query, one of the types in the tree is a group representing the results from executing the query (for example, DBSelect). Select this type as the input card type when defining an input card for the executable map.

After you have defined the query and have generated the type tree, using a database as an input is very similar to using a file as input.

Using the Map Designer, specify Database as the GET > Source setting in the input card and supply the name of the MDQ file containing the definition of the database and query you want to use. see the following procedure and example.

To define a database as a data source:

Note: For more information about the map settings in this procedure, see the **Map Designer** documentation. This procedure specifically addresses database-specific parameters and settings.

Procedure

1. In the Map Designer, when defining the settings for SourceRule, select Database as the GET > Source setting.
The settings displayed in the Input Card dialog change to display the database adapter settings for a source.
 2. From the Card menu, select New.
The Add Input Card dialog appears.
 3. For the CardName setting, enter a name for the card that describes the data object represented by this card.
 4. For the TypeTree setting, select the type tree containing the group type that defines the desired query.
 5. For the TypeName setting, select the group type from the type tree that defines the desired query (for example, DBSelect).
 6. Specify the FetchAs, WorkArea, FetchUnit, and all Backup settings as desired. For detailed information about these settings, see the Map Designer documentation.
For database adapter-specific information about source settings, see the adapter-specific reference guide.
 7. For the GET > Source setting, select Database from the list.
The SourceRule settings change to display the database adapter settings.
 8. Specify the values as desired in the adapter settings and click OK.
-

Database GET> Source settings

After you select Database as the GET > Source setting and the database adapter settings become available for an input, configure the settings for a source that uses the database adapters.

Many of the adapter settings provide transactional control and connection management that allow you to control the connections made to your database. For information about specifying these particular GET > Source and **SourceRule** settings, see [Database Connections and Transactions](#).

- [GET> Source> Command setting](#)
-

GET> Source> Command setting

The **GET > Source > Command** setting is used to enter the applicable database adapter commands for this source. For general information about these settings, see the Map Designer documentation. For specific information about using database-specific adapter commands, see the Resource Adapters documentation.

Using a database as a target

About this task

After using the Database Interface Designer to define tables, views, message queues, or stored procedures for a database, use the database to specify the database adapter as the data target.

To use a database table, view, message queue, or stored procedure as a target:

Procedure

1. Define the database using the Database Interface Designer.
 2. Generate a type tree from a table, view, stored procedure, or message queue.
 3. Using the Map Designer, create a map with an output card that specifies the database table, view, message queue, or stored procedure as the data target.
- [Defining a database target in the map designer](#)
 - [Database PUT > Target settings](#)
-

Defining a database target in the map designer

About this task

When you generate the type tree for a table, view, message queue, or stored procedure, one of the types in the tree is a group representing the contents of the table, view, message queue, or stored procedure. For example, a type tree generated for a table or view is named **DBTable**. Use the Map Designer to select the appropriate type representing the contents as the output card type of the executable map.

After you have defined the database and have generated the type tree, using a database as an output is very similar to using a file as output.

Use the Map Designer to specify **Database** as the PUT > Target setting in the output card at the executable map level. Provide the name of the MDQ file containing the definition of the database, along with the table, view, message queue, or stored procedure to be used.

To define a database as a target:

Note: For more information about the map settings in this procedure, see the Map Designer documentation. This procedure specifically addresses database-specific parameters and settings.

Procedure

1. In the Map Designer, select the To window.
2. From the Card menu, select New.
The Add Output Card dialog is displayed, an example of which follows.
3. For the CardName setting, enter a name for the card.
4. For the TypeTree setting, select the type tree file containing the group type that defines the content of the desired output.
5. For the TypeName setting, select the group type from the type tree that defines the desired output (for example, DBSelect).
6. Specify all Backup settings as desired. For detailed information about these settings, see the Map Designer documentation.
7. For the PUT > Target setting, select Database from the list.
The settings displayed in the Output Card dialog change to display the database adapter settings for a target. For database adapter-specific information about target settings, see the adapter-specific reference guide.
8. Specify the values as desired in the adapter settings and click OK.

Database PUT > Target settings

After you select Database as the PUT > Target setting and the database adapter settings for a target become available in the output card, you can configure the settings for a target that uses the database adapters.

Many of the adapter settings provide transactional control and connection management that allow you to control the connections made to your database. For information about specifying these particular PUT > Target and **TargetRule** settings, see [Database connections and transactions](#).

- [PUT > Target > Command setting](#)
-

PUT > Target > Command setting

The PUT > Target > Command setting is used to enter the database adapter commands applicable for this target. For general information about these settings, see the Map Designer documentation. For specific information about using database-specific adapter commands, see the adapter-specific documentation.

Database connections and transactions

IBM Transformation Extender maps allow superior transactional control and connection management by providing settings that allow the scope of transactions to be controlled by the definitions provided in a map. It is important to understand how these connections to databases are controlled and how this relates to transactional scope.

- [Transactional control](#)
 - [Database connection management](#)
 - [Connection example](#)
-

Transactional control

Scope is one of the settings that may be specified for a source (GET > Source > Transaction > Scope) or target (PUT > Target > Transaction > Scope) in a map. For information about **Scope** settings, see the Map Designer documentation.

There is one restriction for **Scope**. If the value of the SourceRule > FetchAs setting in an input card is set to **Burst**, the **Scope** setting is always **Map**. The adapter scope is determined to be **Map** because the context of the SELECT statement cannot be maintained after the transaction has been terminated through a commit or rollback. Therefore, if **Burst** is the value for the SourceRule > FetchAs setting in an input card, any value in the **Scope** setting as well as any adapter command (-CCARD or -CSTMT) specified in the GET> Source > Command setting is ignored.

Additionally, the global transaction management functionality is available for certain adapters by using the Global Transaction Management (-GTX) adapter command in the **Command** setting. For more information about this functionality, see the Global Transaction Management documentation.

Database connection management

Connection management results in fewer connections being made to the database, which may improve performance. The IBM Transformation Extender engine reuses existing connections whenever possible. With global transaction management for those adapters able to take advantage of this functionality, connections have an even bigger role in maintaining data integrity. For more information about global transaction management, see the Global Transaction Management documentation.

There are two situations in which connection sharing occurs:

- between cards and maps within a map
- between multiple maps running serially within the Launcher

When connection sharing occurs between cards and maps within a map, the connection is described as active as long as there are one or more cards or rules that access a database and that have yet to be committed or rolled back. (For example, an active transaction exists.) When there are no such cards or rules, the connection is inactive, yet still alive.

When using a DBLOOKUP, DBQUERY, GET or PUT function, the Transaction > Scope setting defaults to **Map** when the map SourceRule > FetchAs setting is set to **Integral**. The Transaction > Scope setting will change to **Burst** when the map SourceRule > FetchAs setting is set to **Burst**; the Transaction > OnFailure setting is always **Rollback**.

- [Connection factors](#)
 - [Connection rules](#)
-

Connection factors

The factors determining whether a connection is reused are the following:

- the database type
- the connection string (if applicable), datasource, or database name
- the user ID
- the **OnFailure** setting (**Commit** or **Rollback**)
- the Transaction > Scope setting
- the SourceRule > FetchAs setting
- whether **-CCARD** or **-CSTMT** are specified within the adapter's command

Also, whether a connection will be part of a global transaction is determined by the usage of the Global Transaction (**-GTX**) adapter command in the **Command** setting.

Connection rules

Existing connections are reused whenever possible while adhering to certain connection rules. The rules are as follows:

- An inactive connection is reused if the database type, connection string, and user ID of the new card or rule match those of the previous card or rule that had established the connection.
- In addition to the database type, connection string and user ID, an active connection is reused only if the **OnFailure** and Transaction > Scope settings match in the previous card or rule and the new card or rule.
- If a card or rule has **-CSTMT** set within the adapter's command, this connection may be shared with any other card or rule that has **-CSTMT** set or that has an Transaction > Scope setting of **Card** (or **-CCARD** within the adapter's command).
- If **Burst** is the SourceRule > FetchAs setting for an active card, it (the card) is always executed in its own transaction. It establishes a connection that cannot be shared by any other card for the duration of the map.

Connection example

A map has four database input cards. Assume the connections specified (datasource, userID, and so on) are the same for each.

In this example,

- A connection will be made and a transaction started for **Card 1**.
- Because **Card 2** has Transaction > Scope set to **Card**, this initial connection cannot be shared. Therefore, another connection will be made for this card.
- For **Card 3**, because its **OnFailure** setting differs from the setting for **Card 1**, the connection established in **Card 1** cannot be reused. However, the connection established for **Card 2** is inactive (because Transaction > Scope was set to **Card** and the card has completed) and, therefore, will be reused for **Card 3**. A new transaction is started on this existing connection.
- **Card 4** is able to share the connection established by **Card 1** and can be part of the same transaction because all of its settings match.

If the map fails, the transaction containing both **Card 1** and **Card 4** will be rolled back and the transaction for **Card 3** will be committed.

Updating database tables

This topic discusses how you can designate specific columns in your database to be updated with data produced by a mapping operation.

- [Using key and update columns](#)
 - [Defining key and update columns](#)
 - [Specifying update mode](#)
 - [Update key columns](#)
 - [Example using update columns](#)
-

Using key and update columns

The data produced by a mapping operation can be inserted as new rows in a database table or can update only specific columns in a table as designated.

Designate particular columns in a table to be used as key columns determining whether output data updates an existing row or whether it is inserted as a new row in the table or view. In addition to specifying key columns, you can designate the columns in a table that will be affected by any update operation. You define the columns to update.

The term key in this usage does not necessarily mean that the column is part of a database primary or foreign key.

For any update to be performed, update mode must be specified for the target. This is accomplished by specifying the Update adapter command (-UPDATE) in the PUT > Target > Command settings using the Map Designer or Integration Flow Designer, or by passing it with the override execution command on the command line at execution time. For more information about how to enable update mode, refer to [Specifying update mode](#).

Defining key and update columns

About this task

To update a table from a map, use the Database Interface Designer to specify the columns in a table to be used as key columns and to specifically specify the columns that will be updated.

To designate columns as key columns or columns to update:

Procedure

1. In the Navigator, select the database containing the table for which you want to set update keys.
2. From the Database menu, choose Set Update Keys.
The Set Table Update Key Columns dialog appears.
3. From the Table name list, select the table containing the columns to be updated.
The columns in the table appear in the Columns list.
4. To set a column as a key column, select it from the Columns list and click the right arrow button associated with the Key columns list.
The selected column moves to the Key columns list.
5. To designate a column as a column to update, select it from the Columns list and click the right arrow button associated with the Columns to update list.
The selected column moves to the Columns to update list.
6. To specify all non-key columns as Columns to update, click the all button associated with the Columns to update list.
All of the columns in the Columns list move to the Columns to update list.
7. To remove columns from either the Key columns list or the Columns to update list, select the column and click the associated "delete" button.
The selected columns are moved back to the Columns list.

Results

A generated type tree with specified update keys is represented in the Navigator under Tables.

Specifying update mode

To update a table, you must specify update mode using the Map Designer, Integration Flow Designer, or the command line. By specifying update mode in one of these ways, an update operation can be performed in which each row produced by your map is analyzed and the update is performed based upon the update keys and columns defined in the Database Interface Designer.

- [Using the Map Designer or Integration Flow Designer](#)
- [Using an adapter command at execution time](#)

Using the Map Designer or Integration Flow Designer

After key columns and update columns are defined for a table, specify the table as a target in the **TargetRule** settings in the Map Designer or in the **Output(s)** settings in the Integration Flow Designer.

You must specify -UPDATE in the PUT > Target > Command setting so that your specified columns are automatically updated.

Using an adapter command at execution time

The update mode for a table can be specified at execution time by using the command line to pass the -UPDATE database adapter command. Specify OFF or ONLY to control the updating operation. For more information about using database-specific adapter commands, see the Resource Adapters documentation.

Update key columns

When you specify a column as a key column, the value in that column is used to determine whether a row produced by a mapping operation should be inserted as a new row into the table or should be used to update existing row(s) in a table.

- If the values in the rows generated by the mapping operation match the values in the key columns as defined in the Database Interface Designer, the specified update columns are updated.
- If the values in the rows generated by the mapping operation do *not* match the values in the key columns, the rows are inserted as new rows into the table or view as defined in the Database Interface Designer.

Note: You can override the behavior defined in the Database Interface Designer by using the command line to specify the appropriate database adapter commands for the database adapter. For information about the database-specific adapter commands, see the Resource Adapters documentation.

- If more than one column is designated as a key column, the values generated by a map must match the values in each designated key column for the row to be updated.
- If your map generates a row with key column values that match more than one existing record, all of the matching records in the columns you have specified as update columns are updated with the new row values.
- If your map produces multiple rows with the same values for the key columns, you lose any updates that are made as a result of all but the last row. The updates produced by each row are overwritten by the updates from subsequent rows with the same key values.

Example using update columns

When only some columns in a table are specified as columns to be updated, the values corresponding to all other columns are ignored when the update statement is built and executed. For any row produced by the map, if the values for the update key columns do not match any of the existing rows in the table or view, the values of all columns will be inserted in the new row.

In the following example, a table (**PersonalInfo**) has the following key columns and update columns defined in the Database Interface Designer.

When this map runs, because **-UPDATE** is enabled for the output, the database adapter will first go through the results of the map and update all rows in the table matching the key columns in the output produced. Essentially, the following SQL statements are executed:

```
UPDATE PersonalInfo
SET FirstName='Karl', LastName='March', PhoneNumber='(847)
555-1234'
WHERE ID = 10
UPDATE PersonalInfo
SET FirstName='Janice', LastName='Armstrong',
PhoneNumber='(203) 555-9898'
WHERE ID=14
```

In the first UPDATE statement, because this statement does not find any rows to update, the following SQL statement is executed.

```
INSERT INTO PersonalInfo VALUES (10,'Karl', 'March',
'(847) 555-1234', '999-88-7766')
```

This execution creates a new row in the table for **Karl March**-including values for the **ID**, **FirstName**, **LastName**, **PhoneNumber**, and **SSN** columns.

In the second UPDATE statement, because an existing row has an **ID** value of **14**, only the values of the **FirstName**, **LastName**, and **PhoneNumber** columns are updated because of the settings specified in the **Columns to update** list in the Set Table Update Key Columns dialog of the Database Interface Designer. In this example, **Janice Taylor**'s last name changed to **Armstrong** and her telephone number is changed to **(203) 555-9898**. Her social security number remains unchanged because it is not a column that has been designated for update.

Database functions

This topic discusses two functions (**DBLOOKUP** and **DBQUERY**) that are designed exclusively for use with databases. These functions can be used in component rules in the Type Designer and map rules in the Map Designer when creating a map to be used with a database.

- [Accessing database information in a map rule](#)
- [Using DBLOOKUP and DBQUERY](#)
- [Using bind values in database functions](#)

Accessing database information in a map rule

In many cases, database information for a map will be one of the following:

- the results of a database query or a stored procedure defined as the data source for an input card
- the rows to update or insert into a table defined as the target for an output card.

However, there are certain situations in which you might not want to define the entire query as a data source or the table as a target. For example, you may have a very large table that is used as a source for cross-reference in your map. However, because of its size, reading in and validating all of the information as an input card is impractical. In other situations, you may want to call one map from another using the **RUN** function when one of the data sources is a dynamically built query.

Two functions provide the ability to access database information from within a rule:**DBLOOKUP** and **DBQUERY**. The difference between the two functions is subtle and is based upon how the data resulting from either function is returned.

Using DBLOOKUP and DBQUERY

A single-text-item is returned by either the **DBLOOKUP** and **DBQUERY** function. If your SQL statement is a **SELECT** statement, the **DBLOOKUP** or **DBQUERY** function returns the results of the query in the same format as a query specified in a map input card. If your SQL statement is anything other than a **SELECT** statement, these functions return **NONE**.

Both functions return the results of a query (SQL **SELECT** statement) in the same format as a query specified for a map input card, using the delimited row format. However, the **DBLOOKUP** function strips off the last carriage return/line feed. Because this information is stripped, it is easier to make use of a single value extracted from a database.

`DBLOOKUP` and `DBQUERY` execute an SQL statement within a rule against the database. The SQL statement can be any statement permitted by your database management system or database-specific driver.

There are two syntax methods that can be used to specify the arguments for these two functions: Syntax1 and Syntax2.

- [Syntax1 - using a static MDQ file](#)
- [Syntax2 - using dynamic adapter commands](#)
- [Examples](#)
- [Uses](#)

Syntax1 - using a static MDQ file

Use Syntax1 to execute a SELECT statement that retrieves a specific column value from a large table in a database using the value of another input, as opposed to having to define the entire table as an input card and using other functions such as `LOOKUP`, `SEARCHDOWN`, or `SEARCHUP`. For more information about these functions, see the *Functions and Expressions documentation*.

If column data from a table or database varies because it may be based upon a parameter file, use Syntax2. For more information, refer to [Syntax2 - using dynamic adapter commands](#).

Syntax1 is a three-argument syntax, examples of which follow:

```
DBLOOKUP
(
    SQL_statement,mdq_file,database_name[:adapter_commands])
DBQUERY
(
    SQL_statement,mdq_file,database_name[:adapter_commands])
```

Using this syntax, `DBLOOKUP` and `DBQUERY` use the following arguments.

Argument	Explanation	Must Be
<code>SQL_statement</code>	single-text-expression	a valid SQL statement
<code>mdq_file</code>	single-filename	a string literal
<code>database_name</code>	single-database-name	a string literal
<code>adapter_commands</code>	adapter-commands	a valid adapter command(s)

The following table describes these arguments in more detail.

Argument	Description
<code>SQL_statement</code>	SQL statement as a text string. It can be any valid SQL statement permitted by your database management system and supported by your database-specific driver. In addition to a fixed SQL statement, this argument can be a concatenation of text literals and data objects, enabling the concatenation of data values into your SQL statement.
<code>mdq_file</code>	Name of a database/query file (MDQ) produced by the Database Interface Designer. It contains the definition of the database against which the SQL SELECT statement is to be executed. If the MDQ file is in a directory other than the directory of the map, the complete path must be specified. Note: The MDQ file is accessed at map build time and is not needed at runtime.
<code>database_name</code>	Name of a database in the MDQ file as defined in the Database Interface Designer. Note: This name is case-sensitive and must exactly match the name as defined in the Database Interface Designer.
<code>adapter_commands</code>	Name of an adapter command or commands. This name is an optional argument that appears after <code>database_name</code> , which allows you to specify database adapter commands in component rules in the Type Designer and map rules in the Map Designer. Note: The rules are defined and compiled into your map at design time. Note: All adapter commands (for example, <code>-CS</code>) can be either uppercase or lowercase, but not mixed case.

Syntax2 - using dynamic adapter commands

Use Syntax2 to execute a SELECT statement that retrieves a specific column value from a table or database when the database, table, or other database parameters may vary. This can be used when parameters are being supplied by a parameter file.

```
DBLOOKUP (
    SQL_statement,adapter_commands
)
DBQUERY (
    SQL_statement,adapter_commands
)
```

Using this syntax and the Syntax 2 formatting issues ([Syntax2 formatting issues](#)), `DBLOOKUP` and `DBQUERY` use the following arguments.

Argument	Explanation
<code>SQL_statement</code>	single_text_expression
<code>adapter_commands</code>	Either <code>-MDQ mdq_file -DBNAME db_name</code> or <code>-DBTYPE database_type [database_adapter_commands]</code>

The following table describes these arguments in more detail.

Argument	Description
SQL_statement	SQL statement as a text string. It can be any valid SQL statement permitted by your database management system and supported by your database-specific driver. In addition to a fixed SQL statement, this argument can be a concatenation of text literals and data objects, enabling the concatenation of data values into your SQL statement.
adapter_commands	Either the <code>-MDQ</code> or the <code>-DBTYPE</code> adapter command can be used for this argument.
<code>-MDQ</code>	This adapter command is followed by the name of the MDQ file produced by the Database Interface Designer. This MDQ file contains the definition of the database against which the SQL statement is to be executed. If the MDQ file is in a directory other than the directory of the map, the path must be specified. The MDQ file name is followed by the <code>-DBNAME</code> adapter command and the name of the database in the MDQ file, as defined in the Database Interface Designer.
	Using this syntax, the specified MDQ file must be available at run time.
<code>-DBTYPE</code>	This adapter command is followed by the type of database (for example, <code>ORACLE</code> or <code>ODBC</code>) followed, optionally, by database-specific adapter commands. This syntax does not use an MDQ file because the database-specific adapter commands provide the information required to connect to the database. For more information about your particular database adapter and database-specific adapter commands, see the adapter-specific documentation in the adapter reference guides.
• Syntax2 formatting issues	

Syntax2 formatting issues

When using Syntax2 for `DBLOOKUP` or `DBQUERY`, your function must conform to these rules:

- All adapter commands (for example, `-DBTYPE`) can be either upper or lower case, but not mixed case.
- A space is required between the adapter command and its value (for example, `-DT database_adapter`).
- The order of the adapter commands is not important.

Examples

The section presents different `DBLOOKUP` and `DBQUERY` examples.

- [Example 1 - obtaining a single column value](#)
- [Example 2 - using DBQUERY to obtain multiple columns or rows](#)
- [Example 3 - using DBQUERY to provide map input to RUN function](#)
- [Example 4 - using WORD to parse multi-column output from DBQUERY](#)
- [Example 5 - using DBQUERY with adapter commands to obtain a single column value](#)

Example 1 - obtaining a single column value

Assume that you have a table named **PARTS** that consists of the following data:

PART_NUMBER	PART_NAME
1	1/4" x 3" Bolt
2	1/4" x 4" Bolt

Also assume that this database has been defined in a file named **mytest.mdq** using the Database Interface Designer. The name of the database, as specified in the MDQ file, is **PartsDB**. Notice the difference between the returned values from the execution of the following two Syntax1-formatted functions.

Function	Returns PART_NAME
<code>DBLOOKUP ("SELECT PART_NAME from PARTS where PART_NUMBER =1", "mytest.mdq", "PartsDB")</code>	1/4" x 3" Bolt
<code>DBQUERY ("SELECT PART_NAME from PARTS where PART_NUMBER =1", "mytest.mdq", "PartsDB")</code>	1/4" x 3" Bolt<cr/lf> where <cr/lf> is a carriage return followed by a line feed

Using Syntax2, you can also specify the `DBLOOKUP` or `DBQUERY` functions as in the following examples.

```
DBLOOKUP ("SELECT PART_NAME from PARTS where PART_NUMBER =1",
          "-MDQ mytest.mdq -DBNAME PartsDB")
DBQUERY ("SELECT PART_NAME from PARTS where PART_NUMBER =1",
          "-MDQ mytest.mdq -DBNAME PartsDB")
```

Note that both the MDQ file name and database name are specified.

The examples below use the Syntax2 format to specify the database type and the appropriate database-specific adapter commands (in this example, using the `-DBTYPE`, `-CONNECT`, `-USER`, and `-PASSWORD` commands for an Oracle database):

```
DBLOOKUP ("SELECT PART_NAME from PARTS where PART_NUMBER =1",
          "-DBTYPE ORACLE -CONNECT MyDatabase
          -USER janes -PASSWORD secretpw")
DBQUERY (" SELECT PART_NAME from PARTS where PART_NUMBER =1",
          "-DBTYPE ORACLE -CONNECT MyDatabase
          -USER janes -PASSWORD secretpw")
```

Example 2 - using DBQUERY to obtain multiple columns or rows

Assume that you have a table named **PARTS** that consists of the following data:

PART_NUMBER	PART_NAME
1	1/4" x 3" Bolt
2	1/4" x 4" Bolt

Also assume that this database has been defined in a file named **mytest.mdq** using the Database Interface Designer. The name of the database, as specified in the MDQ file, is **PartsDB**.

Using the Syntax1 format, the following DBQUERY function:

```
DBQUERY ("SELECT * from PARTS", "mytest.mdq", "PartsDB")
```

returns:

```
1|1/4" x 3" Bolt<cr/lf>2|1/4" x 4" Bolt<cr/lf>
```

where <cr/lf> is a carriage return followed by a line feed.

Notice that if the same function was executed using DBLOOKUP, the results would be:

```
1|1/4" x 3" Bolt<cr/lf>2|1/4" x 4" Bolt
```

The difference between the two results is that the final carriage return/line feed is stripped off the end of the results of the DBLOOKUP function.

Using Syntax2, you can obtain the same results as in the previous DBQUERY function as shown in the following examples:

```
DBQUERY ("SELECT * from PARTS where PART_NUMBER =1",
        "-MDQ mytest.mdq -DBNAME PartsDB")
```

or:

```
DBQUERY      ("SELECT * from PARTS where PART_NUMBER =1",
              "-DBTYPE ORACLE -CONNECT MyDatabase -USER janes
              -PASSWORD secretpw")
```

Example 3 - using DBQUERY to provide map input to RUN function

In another example, assume that you have an input file containing one order record. To map that order to another proprietary format, you also have a **PARTS** table with pricing information for every part for every customer-a very large table. Rather than using the entire **PARTS** table as the input to your map, you might include the **RUN** function with your Syntax1-formatted DBQUERY to dynamically select only those rows from the **PARTS** table corresponding to the customer in the order file, as follows:

```
RUN ("MapOrder.MMC",
     "-IE2'" + DBQUERY ("SELECT * FROM PARTS WHERE CustID =" +
CustomerNo:OrderRecord:OrderFile +
" ORDER BY PARTNO",
"PartsDB.MDQ",
"PartsDatabase") +"'")
```

Example 4 - using WORD to parse multi-column output from DBQUERY

In certain situations, you may want to use one of the database functions, rather than a database source, due to the size of a cross-reference table. However, you need the function to return the data from several different columns.

For example, assume you need to create a map that processes inventory requests, one order at a time, using a messaging system. Within the transformation of your data, you need to reference the item master table that contains hundreds of thousands of rows. However, for each item within the inventory request, you need to get the internal item number, vendor ID, and description column value. You can select from several available options, such as:

- Use a database source.
Define a query for only those columns needed. Use this for a database source that you can then use within a **LOOKUP**, **EXTRACT**, or **SEARCHUP/SEARCHDOWN** function. However, due to the size of the item master table, this might mean validating hundreds of thousands of rows to find the item information for only a few items.
- Use multiple DBLOOKUP functions.
Assuming that the internal item number, vendor ID, and description column values are going to be used within different outputs, you could use three separate DBLOOKUP functions to get the appropriate column value for each item. However, this means executing three SQL statements to access the same row within the item master table.
- Use the DBLOOKUP and WORD functions.
Assuming that a functional map will be used to build an object containing the three desired columns, a DBLOOKUP could be used as an argument to the functional map that retrieves the desired column values. see the following example:

```
=F_MakeOne ( Item Set:SomeInput ,
DBLOOKUP ( "SELECT INT_ITEM_NO, VENDOR, ITEM_DESC " +
```

```
"FROM ITEM_MASTER WHERE ITEM_NO = '" +  
CatalogID:.:SomeInput + "'", "PRODXL.MDQ"  
"WDDM" ) )
```

The functional map **F_MakeOne** has two inputs: an **Item Set** and a text item (that is the result of the **DBLOOKUP** function). The text item will contain the three column values separated by the pipe character (|). An example follows:

```
ARQJ06X6|DFQCO|6' Jump Rope
```

Then, each rule requiring one of these pieces of data will use the **WORD** function to access the appropriate column's data. For example, if the input card for the results of the **DBLOOKUP** function was called **ItemData**, the rule using the vendor ID column would be:

```
=WORD ( ItemData , "|" , 2 )
```

The values of the other columns can be retrieved in a similar manner.

Example 5 - using DBQUERY with adapter commands to obtain a single column value

This example demonstrates how you can use the **DBQUERY** function to enter adapter commands as an argument following **database_name** using the **Syntax1** format:

```
DBQUERY("SELECT PART_NAME from PARTS  
where PART_NUMBER =1",  
"mytest.mdq","PartsDB : -T -CS")
```

In the **DBQUERY** example above, the **PART_NAME** column value in the row that contains the **PART_NUMBER** column value of 1 will be retrieved from the **PARTS** table. The **PARTS** table is in the **PartsDB** Oracle database specified in the **mytest.mdq** file, which was produced by the Database Interface Designer.

This example uses optional database-specific adapter commands. The Trace (-T) adapter command will produce a database trace file. The Commit by Statement (-CS) adapter command will commit or rollback the transaction after the rule executes.

Uses

When you might want to use **DBLOOKUP** to execute an SQL statement:

- To retrieve a single column value from a database based upon another value in your data without the carriage return/line feed row terminator

When you might want to use **DBQUERY** to execute an SQL statement:

- To look up multiple column values for multiple rows in a database using a parameterized query based upon another value in your data
- When your SQL statement is a **SELECT** statement, the **DBQUERY** function can be used along with the **RUN** function to issue dynamic **SELECT** statements whose results can be used as input to a map.
- When using other SQL statements such as **INSERT**, **UPDATE**, **DELETE**, and so on.

For information about functions such as **EXTRACT**, **LOOKUP**, **SEARCHUP**, **SEARCHDOWN**, **GET**, and **PUT**, see the Functions and Expressions documentation.

Using bind values in database functions

For information about the availability and usage of the bind facility for your specific database and platform, see the database-specific adapter reference guides.

When the Database Management System (DBMS) receives an SQL request, the request is cached because many applications repeatedly issue the same SQL statement. If the SQL statement differs from one that the DBMS has recently processed, the statement is reevaluated. (The DBMS performs parsing, derives an execution plan, and so on.)

Similarly, if a **DBLOOKUP** or **DBQUERY** function repeatedly issues the same statement, the second and subsequent executions of the statement execute much faster than the first execution. However, if any element of the statement varies, the DBMS considers the statement to be new and does not take advantage of caching. For example, the two following statements are distinct to a DBMS:

```
SELECT * FROM MyTable WHERE CorrelationID=123  
SELECT * FROM MyTable WHERE CorrelationID=124
```

Use the bind facility for **DBLOOKUP** and **DBQUERY** functions to submit such statements to the DBMS so that the statements are syntactically identical. By binding a value to a placeholder in the SQL statement, the actual syntax of the statement can be made static.

The syntax for specifying a value in the SQL statement as a bind value is:

```
:bind(  
value  
)
```

For example, to use a bind variable in the statement above, the SQL statement would be:

```
SELECT * FROM MyTable WHERE CorrelationID=:bind(123)
```

The database adapter strips out the **:bind** keyword and binds the value 123 to a placeholder in the statement.

The value in the parentheses is always a text item. Single quotation marks should not be specified around string literals. For example, if you had the statement:

```
SELECT Artist FROM CDList WHERE Title = 'Goodbye'
```

and you want to bind the value for the title, the syntax would be:

```
SELECT Artist FROM CDList WHERE Title=:bind(Goodbye)
```

Within the context of a DBLOOKUP or DBQUERY function, the elements of the statement to be bound are dynamic elements. For example, if the following call to DBLOOKUP is in a map:

```
DBLOOKUP      ("SELECT Name FROM MyTable WHERE ID="+ Item1:Row +
    "and CorrelationID= '" + Item2:Row + "'",
    "DB.mdq",
    "MyDB")
```

The call could be modified to benefit from binding values as follows:

```
DBLOOKUP      ("SELECT Name FROM MyTable WHERE ID=:bind("+ Item1:Row +")
    and CorrelationID=:bind(" + Item2:Row + ")",
    "DB.mdq",
    "MyDB")
```

There is no performance benefit unless *all* values that change from one invocation of the statement to the next are bound. For example, if the ID value is bound and *not* the CorrelationID value, the statement will vary because the CorrelationID value varies.

Using stored procedures

This topic describes the various mechanisms for using stored procedures to access all types of parameters and return values from stored functions. You can call stored procedures when using DBQUERY and DBLOOKUP functions and when defining a query in the Database Interface Designer for a map data source. When defining a database as the target for an output card, you can also output to a stored procedure.

For information about the availability and usage of calling stored procedures, including the correct native syntax for these calls, refer to each database-specific adapter reference guide.

- [Calling stored procedures](#)
- [Examples using stored procedures](#)
- [Using a stored procedure as an input](#)
- [Using a stored procedure as an output](#)
- [Stored procedures with object type parameters](#)

Calling stored procedures

When used as a source of data in one of the database functions (DBQUERY or DBLOOKUP), in a GET function, or in a query defined in the Database Interface Designer, there are two ways to call stored procedures:

- Use the database-independent syntax for calls.
This provides a means of accessing the output parameters and return values from stored functions.
- Use the native syntax for the database.
Using this method, you cannot access output parameters or return values from stored functions.

In addition to the above methods, when using Oracle object types as parameters in stored procedures, there is a special syntax that must be used. For more information about this, see [Stored Procedures with Object Type Parameters](#).

- [Database-independent syntax for calls](#)

Database-independent syntax for calls

The database-independent syntax for calling stored procedures is:

```
CALL [?=?] procedure_name [(argument_list)]
```

The arguments are described below.

Option	Description
?=	Indicates that the return value from a stored function should be returned.
<i>procedure_name</i>	The name of the stored procedure or function. To invoke a stored procedure in an Oracle package, fully qualify the stored procedure name. For example:

```
schema_name.package_name.stored_procedure_name
```

argument_list

A comma-separated list of arguments in which each argument is one of the following:

Argument	Description
----------	-------------

value	This is the value to be passed to an input parameter. If the value contains spaces, it must be surrounded by single quotation marks (for example, 'city of'). The quotation mark characters are not passed as part of the parameter.
?	Return the value from an output parameter.
X	Do not return the value from an output parameter.
?/value	This is the value to be passed to an In / Out parameter and the value to be returned from the parameter.

Examples using stored procedures

In the following example, there is a stored procedure named **DoSalaryIncrease** with the following parameters:

Employee ID -	parm1 - In
Merit Increase (%) -	parm2 - Out
Salary -	parm3 - In / Out
Effective Date -	parm4 - Out

Using the DBLOOKUP function, an example of the syntax used to call **DoSalaryIncrease** would be:

```
DBLOOKUP
("call DoSalaryIncrease (
+ EmpID::PayrollFile
+ ?,?/?"
+ Salary::PayrollFile + ?,?)", "AdminDb.mdq", "HR_DB")
```

If **EmpID** has a value of **SM01930** and **Salary** has a value of **42,750**, the call syntax would expand to:

```
call DoSalaryIncrease (SM01930, ?, ?/42750, ?)
```

The output from this call is a single text item in which each field is delimited by the pipe character (|). In this example, there will be three fields in the output-one for each question mark character (?) placeholder. If you do not want to use all of the output parameters, you can use any other character in place of the question mark character. For example, if you did not use the **EffectiveDate** (parm4), the call could be changed to:

```
DBLOOKUP
("call DoSalaryIncrease (" + EmpID::PayrollFile
+ ?,?/?" + Salary::PayrollFile + ",X)",
"AdminDb.mdq", "HR_DB")
```

Once again, if **EmpID** has a value of **SM01930** and **Salary** has a value of **42,750**, the call syntax would expand to:

```
call DoSalaryIncrease (SM01930, ?, ?/42750, X)
```

In this example, the **EffectiveDate** (parm4) would not be returned.

- [Returning the Value from a stored function](#)

Returning the Value from a stored function

If you want the value from a stored function, prefix the procedure or function name with **?=**. For example:

```
DBLOOKUP("call ?= MyFunction('cat') , "My.mdq" , "MyDB")
```

Using a stored procedure as an input

Stored procedures can be used in input cards by specifying the call in the **Query** field in the New Query dialog of the Database Interface Designer.

After the stored procedure call is entered in the New Query dialog or the Edit/View Query dialog (if you are editing an existing query), you can generate a type tree for the stored procedure.

Generated type trees for stored procedures adhere to the same format as type trees for queries and will contain a **Row** group, the components of which correspond to the ? placeholders in the CALL statement. Return values from stored functions will typically have the field name **RETURN_VALUE** unless the database returns a specific name.

The call syntax for a stored procedure can be used effectively in input cards in combination with substitution variables to provide values to input parameters. For example, to call **GetPaymentInfo** in an input card, because **GetPaymentInfo** takes a single input parameter, first define the text in the **Query** field.

Next, provide the value on the command line for **parm1** at execution time as follows:

```
dtx MyMap.mmc -ID1 '-VAR parm1=2000-03-11'
```

- [Using a query to execute a stored procedure](#)

Using a query to execute a stored procedure

If your database permits the execution of a SELECT statement in a stored procedure or function and can return values through subsequent row fetches, you can specify a procedure call using the native database syntax in the **Query** field of the **New Query** or Edit/View Query dialog using the Database Interface Designer.

For example, to call a stored procedure by means of ODBC using the native syntax, the query text you enter in the **New Query** or Edit/View Query dialog might be:

```
{call MyProc(-1)}
```

The query text for a call to a stored procedure must be in the native database syntax. For more information about the correct syntax of the procedure call using your particular database adapter, see the database-specific adapter reference guide.

To define a query using a stored procedure:

When you use a stored procedure to define a query, follow the same steps as when you use a SELECT or other statement:

1. Define the query in the **New Query** or Edit/View Query dialog of the Database Interface Designer.
2. Generate a type tree from the query.
3. In the Map Designer, specify the query when defining the data source for the executable map input card for the stored procedure.

Using a stored procedure as an output

Using the database adapters, you can call stored procedures in output cards.

Type trees generated for stored procedures contain an item type for each input argument. Instead of a **Row** type, these type trees have a **ProcedureCall** type that represents the set of arguments passed to the stored procedure for each execution of that procedure. The **DBProcedure** type consists of a series of **ProcedureCalls**. The stored procedure is called once for each **ProcedureCall** in the **DBProcedure** output. This information is highlighted in the following example.

There are two ways to call stored procedures in output cards:

- In the output card dialog in the Map Designer, specify `-PROC procedure_name` in the PUT> Target > Command setting. For more information, refer to [Using a database as a target](#).
or
- When overriding an output card using database adapter commands on the command line, use the `-PROC` adapter command and specify the stored procedure name.
For more information, see the Resource Adapters documentation.

The values for each parameter are passed to the stored procedure with the stored procedure being called for each row of data.

The types of the parameters may be IN, OUT, or IN/OUT. However, there is no mechanism to return values from output parameters. Any values passed to an OUT parameter will be ignored.

Stored procedures with object type parameters

The Oracle adapter provides full support for Oracle object types to be used as parameters to stored procedures or functions. Mapping to a stored procedure call in an output card is no different than mapping to a table; object types as parameters are fully supported.

Similarly, invoking stored procedures with object type parameters using the 'call' syntax is possible. For output parameters, there is no special syntax required. Using a question mark character (?) will result in the entire object being returned from the stored procedure call. However, for input parameters, a special syntax is required to specify the objects. The syntax rules are as follows:

- The object must be contained within square brackets. An example of this is: "[.....]"
- Each element of the type is separated from other elements of the type by the pipe (|) delimiter character.
- Spaces are not allowed unless they are contained within the data itself.

see the following example. The type 'outer' is defined by the following 'create type' statements:

```
create type inner as object (
  a_char varchar(10),
  b_int number(10,0));
create type outer as object (
  x_inner inner,
  y_date date);
```

Assume that there is a table named 'object_holder' and a stored function 'insert_object' defined to insert an object of type 'outer' and to return the number of objects in the table. The SQL required to create this follows:

```
create table object_holder (
  myobj outer);
create or replace function insert_object (obj in outer) return
number is
row_count number;
begin
  insert into object_holder values (obj);
  select count(*) into row_count from object_holder;
  return row_count;
end;
```

To provide an object with the following attributes:

```
x_inner.a_char = 'hello'
x_inner.b_int = 23
y_date = 2000-10-12 04:59:23
```

use the following query:

```
call ?= insert_object([[hello|23]2000-10-12 04:59:23])
```

Note that the inner object is also delimited by square brackets.

Database triggers

Data sources using database adapters can serve as input event triggers that are defined in the Database Interface Designer, enabled in the Integration Flow Designer, and executed by an Launcher.

For information about the availability and usage of triggering for your specific database and platform, see the database-specific adapter reference guide.

- [Database triggers overview](#)
- [Using a database as a map trigger](#)
- [Specifying a combination of different event classes](#)
- [Specifying triggers on the command line](#)
- [Using the Integration Flow Designer to enable triggers](#)

Database triggers overview

This introductory section provides information about the following topics:

- [Database Support](#)
This topic discusses those databases supporting triggering and the types of triggering supported.
- [Installation Requirements](#)
This topic discusses the database-specific scripts required to be run to enable database triggering.
- [Tables Created for Triggering](#)
This topic discusses the four tables created as a result of running the installation scripts for triggering.
- [Maintaining Triggering Tables](#)
This topic discusses table maintenance required in unusual situations such as unexpected shutdown of Launchers and working with truncated tables.
- [Database support](#)
- [Installation requirements](#)
- [Tables created for triggering](#)
- [Maintaining triggering tables](#)
- [Table-based triggering](#)
- [Row-based triggering](#)
- [Issues for both row- and table-based triggering](#)
- [Issues for row-based triggering only](#)
- [Defining a trigger using a database source](#)

Database support

The event(s) that occur causing a trigger to run a map may be the result of a map or some other application.

The following databases support triggering:

- Oracle client and server (for Windows and UNIX)
- Microsoft SQL Server (for Windows only)
For information about the versions of these databases that are supported in this release, see the [system requirements](#).

The Database Interface Designer allows two different triggering actions during event-based map execution:

- [Table-based](#) triggering
- [Row-based](#) triggering
 - [Column-based](#) triggering (an enhancement to row-based)

Installation requirements

Before you can take advantage of the database triggering functionality, either one of three database-specific installation scripts must be run by a system administrator:

Note: This is a one-time only operation except for a possible unexpected Launcher shutdown. For more information about that scenario, refer to [Handling Unexpected Shutdowns](#).

- [m4ora.sql](#) (for Oracle)
- [m4ora_col.sql](#) (for Oracle)
- [m4sqlsvr.sql](#) (for Microsoft SQL Server)

Executing the respective script installs the database objects required for database triggering:

- tables that track "watch" events Four tables are added to your database. These are required for the triggering functionality. For more information about these tables, refer to [Tables created for triggering](#).
 - stored procedures that interface to the tables
 - (for Microsoft SQL Server only) extended stored procedures that contain signaling logic
 - (for Oracle only) public synonyms for tables and stored procedures
 - (for Oracle only) sequences that generate unique trigger IDs
Certain privileges must be granted to the user running the database card triggers. The specific details for this and additional information are presented at the beginning of each script.
 - [Column-based triggering](#)
-

Column-based triggering

To use column-based triggering (for Oracle only), run either of the following scripts:

Script

Use
m4ora_col.sql

if you are upgrading from a release prior to IBM Transformation Extender 8.0, and had been using Oracle database triggering

m4ora.sql

to re-install the database objects required for column-based triggering

For more information, refer to ["Column-Based Triggering"](#).

Tables created for triggering

See the following table for information about the four tables that are created by executing the SQL script.

Table

Description

Trigger_Server

Used at startup, this table tracks the Launchers that are accessing this database. The Launchers must have unique machine name and TCP/IP address combinations. (In other words, triggering is only supported for one Launcher per machine.)

Trigger_Catalog

Used at startup, this table tracks all triggers for all tables that have been defined on this database. This functionality allows table "watches" to be reused across multi-user/Launcher environments.

Trigger_Registry

Used during processing, this table registers every "watched" table for each Launcher. The "watches" on each card are placed here. In this registry is every map on every machine that is being "watched".

Trigger_Events

Used during processing, this table records all events that have occurred as a result of the table "watches" defined in the Trigger_Registry table.

Note: For Oracle only. If an Oracle table is truncated or dropped while having unprocessed row-based table entries, there may be map execution problems. To avoid these problems, you may have to perform some maintenance on this table. For information about how to do this, refer to [Handling truncated tables](#).

Maintaining triggering tables

The current implementation of the triggering functionality should be self-maintaining. However, there are unusual situations in which some user maintenance might need to be performed as described in the following sections:

- [Handling unexpected shutdowns](#)
 - [Handling truncated tables](#)
 - [Handling unexpected shutdowns](#)
 - [Handling truncated tables](#)
-

Handling unexpected shutdowns

Note: Make sure that there are no other Launcher systems running on the target DBMS before running the maintenance operations listed below.

If an Launcher system (containing database triggers) has encountered an unexpected shutdown and cannot be re-started and subsequently shut down in a normal sequence, you must re-run the triggering installation script (specific to your database) to clean up any triggering resources that have been left behind on the target DBMS. This is because when an Launcher is prematurely shut down, any defined database triggers will remain in operation on the backend DBMS (for example, Oracle) until the same system is re-started and then properly shut down.

The database triggers will continue to monitor for events while the Launcher is shut down, thus enabling the system to be fault-tolerant with respect to any database changes that occur during an unexpected downtime. This fault-tolerant behavior requires that a normal system shutdown eventually take place. Otherwise, the database triggers will continually remain in operation, taking up processing and space resources on the target DBMS.

If the triggering installation script cannot be run, a temporary solution would be to run the following SQL statements on the target DBMS:

1. DELETE FROM Trigger_Events;
2. DELETE FROM Trigger_Registry;

3. COMMIT;

Handling truncated tables

Note: For Oracle only.

If an Oracle table is truncated or dropped while having unprocessed row-based table entries, there may be map execution problems. To avoid these problems, you may have to perform some maintenance on the Trigger_Events table as shown in the following procedure.

Make sure you only delete those rows in the Trigger_Events table associated with the truncated table. Also, you can perform this procedure after dropping and recreating a table.

To delete table-specific truncated rows:

1. Enter the following SQL statement:

```
DELETE from Trigger_Events WHERE ID IN (
    SELECT ID FROM Trigger_Registry WHERE TriggerName IN (
        SELECT TriggerName FROM Trigger_Catalog
        WHERE UPPER(SourceSchema) = UPPER('schema_name') AND
        UPPER(SourceTable) = UPPER('table_name')))
```

where *schema_name* is the name of your schema and *table_name* is the name of the truncated table.

2. Commit the transaction (for example, `COMMIT;`).
-

Table-based triggering

A typical usage for table-based triggering would be to set up a trigger that will execute a map to run after another map has completed processing and has inserted rows into a table.

For information about the Source > Transaction > Scope setting, see [Database sources and targets](#). For information about using the Commit by Statement adapter command (`-CSTMTC`), see the Resource Adapters documentation.

To see how to construct the WHEN expression for table-based triggering, see the [Specifying when](#) topic.

Row-based triggering

The row-based triggering design enables a fault-tolerant, multi-server-aware triggering mechanism that can enable maps to selectively process those rows associated with a triggered watch event. In the preceding sentence, fault-tolerant means that data is not lost if a process fails, a connection is dropped, an Launcher stops, or some other unforeseen interruption occurs. Multi-server-aware means that Launchers on different machines can connect to and "watch" the same table.

The main difference between row-based and table-based triggering is that with row-based triggering, the query executes on only those rows that have been updated or inserted, not on the entire table. Additionally, batch-like processing is possible if the When clause is used to control the time and/or date at/upon which processing should occur.

When using column-based triggering, batch processing does not occur.

To see how to construct the WHEN expression for row-based triggering, refer to the example in [SELECT 1 FROM Format](#).

- [Column-based triggering](#)
-

Column-based triggering

An enhancement to the row-based triggering functionality is column-based triggering. It enables you to trigger an event to occur on a row when a particular column condition is met. This functionality is available in both the **m4ora.sql** and the **m4ora_col.sql** scripts. To use this functionality to trigger an Oracle database watch event for a map when a column condition is met, supply the column condition extension of the SQL WHEN expression in the Trigger Specification dialog of the Database Interface Designer. The portion of the WHEN expression that appears after **column condition** is the SQL clause that is passed to the Oracle database for processing.

Refer to Oracle documentation for the WHEN clause syntax when using it in a trigger definition.

For information on installing the **m4ora.sql** and the **m4ora_col.sql** scripts, refer to ["Installation Requirements"](#).

To see how to construct the WHEN expression for column-based triggering, see the example in the [Format of the When Expression](#) section.

Issues for both row- and table-based triggering

The following list represents what is and is not supported in row- or table-based triggering:

- The appropriate database must be used (supported versions of either Oracle client and server [for Windows and UNIX] or Microsoft SQL Server [for Windows only]).
- The respective SQL file (refer back to ["Installation requirements"](#)) must be installed and executed by your system administrator.
- SELECT statements containing Union or Intersect clauses are not supported.

- Case sensitivity for either column or table names is not supported.
- Only one Launcher per machine is supported.

Issues for row-based triggering only

The following list represents what is and is not supported in row-based triggering only:

- For Microsoft SQL Server, each table defined in the query must contain either an identity column or one numeric primary key.
- Queries that reference views or public synonyms are not supported.

Defining a trigger using a database source

About this task

Specify that you want a database source to be a trigger for a map when using a database adapter that supports triggering and an Launcher.

To define a trigger:

Procedure

1. Use the Database Interface Designer to create an MDQ file containing the desired query.
2. From the Trigger Specification dialog in the Database Interface Designer, define the conditions under which changes in database tables will trigger a map. The trigger you define is associated with a query and is saved in the MDQ file. For more information about this, refer to [Defining a trigger for a query](#).
3. Use the Map Designer to create an executable map with an input card that uses the MDQ file containing the trigger specification. For more information, see the Map Designer documentation.
Make sure that the input card corresponding to the database trigger is set to integral mode, not burst.
4. Use the Integration Flow Designer to create a system with the map component containing the trigger specification. Enable the Input(s) > GET > Source setting in the Launcher Settings dialog. For information about how to do this, refer to [Using the Integration Flow Designer to enable triggers](#).
The Integration Flow Designer does not perform any validation of a database source used as a trigger. Ensure that the trigger has been defined using the Database Interface Designer prior to specifying it as an input event trigger.
5. Generate the Launcher system file (.msl) on the Launcher as appropriate. For more information about how to do this, see the Integration Flow Designer documentation.

Using a database as a map trigger

Database sources can be used as input events for the Launcher. Insertions, deletions (for table-based triggering only), or updates to database tables can be the trigger mechanism that will run a map.

A trigger specification is defined in the Database Interface Designer for an individual query and is stored with the query in the MDQ file. When a database source is specified as an input event in the Integration Flow Designer, the trigger specification is evaluated by the Launcher and is run accordingly.

- [Defining a trigger for a query](#)
- [Defining events](#)

Defining a trigger for a query

About this task

After you have defined a query in the Database Interface Designer, you can define a trigger specification that can be used to launch a map using that same query as a data source.

To define a trigger specification:

Procedure

1. In the Database Interface Designer, select the query for which you want to define a trigger specification.
2. From the Query menu, select Define Trigger.
The Trigger Specification dialog appears. When initially displayed, the lists on the left show all of the tables referenced by the SELECT statement of the selected query.
3. Determine whether you can define row-based triggering for this query as determined by the availability of the Row-based triggering check box.
If this is a new query and the table supports it, this check box will be enabled as the default value. (For old queries, this check box will be cleared as the default value, which means that you could use the functionality by enabling it. In another scenario, the entire check box could be disabled because this functionality is not available for the query and table being used.)
4. To specify the events, select a table name from the Insert into or Delete from lists.
or
Select a table name from the Update of list.

The name of the table you select will be either the destination into which data will be inserted, from which data will be deleted, or in which data will be updated when the specified event occurs.

Note: Delete from cannot be used with row-based triggering.

5. To move the table name, in the text box area corresponding to the selected table name (either Insert into, Delete from, or Update of), click the arrow button. The table name moves into the corresponding list.
6. Select as many names as necessary to define the event(s) that must occur so that conditions can be met for the trigger specification.
7. If more than one table is added to a list, you can define conditions that must be met. Select either the AND or OR radio button located directly above each list. For more information, refer to "[Specifying a combination of different event classes](#)".
- The AND option dictates that the condition is not met until the event occurs for all tables in the list when data is either inserted, deleted, or updated. The OR option dictates that the condition is met when the specified event occurs in only one table in the list.
8. To define an additional condition that will be evaluated only after the other specified event(s) has/have occurred, enter an expression in the When field. For more information, refer to "[Specifying when](#)".
9. After you have finished, click OK to save your trigger specification.

The query with the trigger specification is represented in the Navigator with an icon displayed next to the name.

Note: To modify or delete an event, use the same procedure as indicated above. Select the table name in the list on the right and click the corresponding left arrow button. The When expression can also be edited or deleted as required.

Defining events

About this task

Use the Trigger Specification dialog to define a trigger specification for an individual query that will be stored with the query in the MDQ file. As you add and remove the table names, the lists on the right display the tables for which an **Insert into**, **Delete from**, or **Update of** event comprises the condition(s) that must be met for the trigger specification. Additionally, you can specify whether row-based triggering will be used. The different classes of events that you can specify are:

- **Insert into**
The insertion of rows into the specified table serves as an input event trigger to the map that uses this query as a data source.
- **Delete from**
The deletion of rows from the specified table serves as an input event trigger to the map that uses this query as a data source.
Note: This functionality cannot be used with row-based triggering. As soon as you move a table to the right, the Row-based triggering check box is disabled.
- **Update of**
The update of rows in the specified table will serve as an input event trigger to the map that uses this query as a data source.

Specifying a combination of different event classes

About this task

If you define your trigger specification using a combination of event classes (**Insert into**, **Delete from** or **Update of**), there is an implicit OR between the different event classes.

For example, if you defined events in all three event classes, the implied condition to be met for the trigger specification would be:

```
Insert into TableA  
OR  
Delete from TableB  
OR  
Update of TableC
```

When all events have occurred within only one of the specified event classes, the condition(s) has/have been met for the trigger specification and the map is run.

- [Specifying AND or OR](#)
- [Specifying when](#)

Specifying AND or OR

Within the **Insert into** and **Delete from** event classes, you can define multiple insert and delete events by specifying multiple tables. Similarly, you can define multiple **Update of** events by specifying multiple table names.

Using the **Delete from** event class disables row-based triggering. However, you can still define multiple **Insert Into** and **Update of** events using row-based triggering.

When you specify multiple table names within an event class, you must also specify how you want the condition within the event class to be met-either when all of the events occur for all of the specified tables or when one event occurs on any one entry in the list.

The options that dictate the condition(s) to be met when you set up multiple events are:

Option	Description
AND	This option dictates that all of the specified events must occur (for example, TableA AND TableB) for the condition to be met.
OR	This option dictates that conditions are met when at least one event occurs (for example, TableA OR TableB).

Specifying when

After specifying at least one **Insert into**, **Delete from**, or **Update of** event, you can create an expression that must evaluate to true to satisfy the conditions of the trigger specification. Enter an expression in the **When** field that will be evaluated after the other events have occurred. If the **When** expression evaluates to true, the conditions of the trigger specification are met and the map is run. If the **When** expression is not true, the state is restored to what it was prior to the occurrence of any events. When row-based triggering is used, all of the changed rows will be batched together for subsequent processing after the event is re-triggered and the **When** clause has been satisfied.

When using column-based triggering, batch processing does not occur.

- [Format of the when expression](#)

Format of the when expression

There are three basic formats that can be used for the **When** expression as specified in the **Trigger Specification** dialog:

- Clauses referencing table columns ([Table column format](#))
- Clauses using the SELECT 1 FROM format to establish conditions of execution ([SELECT 1 FROM format](#))
- Clauses using the column condition ([Column condition format](#))

Table column format

The **When** expression in the **Trigger Specification** dialog can contain any SQL expressions that are valid for the database. If database columns are referenced in the expression, the column name must be qualified with the `tablename` and the `tablename.column_name` enclosed in square brackets ([]).

For example, if a map should be triggered only when there is a row in the **MyTable** table having a column entitled **Status** with a value of **Ready**, specify the **Insert into**, **Delete from**, and **Update of** events and enter the following expression in the **When** field:

```
[MyTable.Status] = 'Ready'
```

SELECT 1 FROM format

The **When** expression can support any valid SQL statement that begins with a **SELECT 1 FROM** clause. After a database event is detected on the DBMS, the entire statement (as it appears in the Trigger Specification dialog) is executed.

For example, if a database input card should be triggered for execution only during a certain time of day, the following statement could be entered:

Note: This example only applies to versions of the Oracle DBMS supported in this release.

```
select 1 from dual where  
TO_CHAR(SYSDATE, 'HH24') > '00' AND  
TO_CHAR(SYSDATE, 'HH24') < '06'
```

This **When** clause would restrict processing of any database "watch" events to the timeframe of between midnight and 6AM.

Column condition format

The **When** expression can support column-based triggering.

An example of the syntax of the column condition that you would specify in the Trigger Specification dialog of the Database Interface Designer is presented below:

```
column condition new.part_number = 'S'
```

Because column-based triggering is based on row-based triggering, the **Row-based triggering** check box must be enabled. You would enter the column condition `new.part_number = 'S'` clause into the **When** pane. The portion, `new.part_number = 'S'`, specified after column condition, is the SQL clause that the Oracle database will process.

Specifying triggers on the command line

If a trigger specification for a query has not been defined in the Database Interface Designer or if you want to override conditions that are already defined in a trigger specification associated with a query, use the command line.

For more information about using the command line and specifying the Trigger adapter command (-TR), see the Resource Adapters documentation.

Using the Integration Flow Designer to enable triggers

About this task

After you have defined the trigger using the Database Interface Designer and have created an executable map using the Map Designer, use the Integration Flow Designer to define a system containing the associated map component and to enable the trigger for execution by the Launcher. For more information, see the Integration Flow Designer documentation.

To set the database source as a trigger:

Procedure

1. In the Integration Flow Designer, display the Launcher Settings dialog for the map you want to trigger.
2. Set the SourceEvent setting value to ON.
3. Click OK.
A trigger icon appears on the map component in the system diagram. When the map component appears in expanded state, a trigger icon also appears on the input card.

The trigger icon provides a visual cue showing the inputs that serve as triggers.

Debugging and viewing results

This topic explains various troubleshooting tools available when you encounter problems using database objects as data sources or targets for a map or when using the Database Interface Designer to define databases and queries. Also presented are various methods for viewing data extracted from a database or loaded into a database.

- [Troubleshooting tools](#)
 - [Database trace files](#)
 - [Viewing database source and target data](#)
 - [Database audit files](#)
 - [DBMS trace utilities and SQL command tools](#)
-

Troubleshooting tools

If you receive an error while attempting to generate a type tree in the Database Interface Designer or if you run a map that uses database sources and/or targets and receive a runtime error or do not get the expected output, use any or all of the following troubleshooting tools:

- Map audit log (either `Mer_mapname_processkey_mapcounter_hostname.log` for unique audit logs or `mapname.log` for audit logs that do not use the Unique setting)
For information about the map audit log and related settings, see the Map Designer documentation.
 - Database Interface Designer trace (`mdq_filename.dbl`)
 - Database execution trace file (`mapname.dbl`)
 - Database source and target data
 - Database audit file (`audit.dbl`)
 - DBMS trace utilities and SQL command tools
-

Database trace files

Use the information contained in database trace files (**.dbl**) as one of the primary tools to assist in troubleshooting. These files contain detailed information generated during Database Interface Designer activity and also during map execution. For example:

- The trace file recorded for a Database Interface Designer trace contains information about the activities taking place when generating type trees such as database connections, SQL statements executed, and so on.
 - The database trace file produced at map execution time records detailed information about the database adapter activity such as records retrieved, data source and target activity, and so on.
 - [Format of database trace files](#)
 - [Producing the database trace in the Database Interface Designer](#)
 - [Producing the database trace during map execution](#)
 - [Tracing database errors only](#)
-

Format of database trace files

Information displayed in the database trace file varies depending upon the database generating the trace file. Database adapter trace files contain what appears to be a range of numbers at the beginning of each line in the file. These numbers represent:

- the process ID
This is the number representing the process that is running on the machine. If you were watching the processes running on your machine (viewing the **Processes** tab in the Task Manager window), the value displayed in the **PID** (process ID) column would match the value displayed in the database trace file produced as a result of this process.
- the thread ID
This ID distinguishes each map that is running concurrently. Each running map has its own thread.

Producing the database trace in the Database Interface Designer

About this task

Click the **Trace** tool or select Trace from the File menu to enable a database trace file (.dbi). If trace is enabled, a database trace file is automatically created when you generate a type tree. This file is placed in the same directory as the currently open MDQ file. The newly created database trace file is named using the full name of the MDQ file plus a .dbi extension. For example, if your MDQ file is named **Orders.mdq**, the trace file is named **Orders.dbi** and is located in the same directory. If your MDQ file is not yet named, the trace file is named **Database_QueryFile 1.dbi** and resides in the same directory, typically the default installation directory.

Most problems encountered in the Database Interface Designer are relayed to the user in message dialogs. For example, if an incorrect user-ID or password is specified for a database, a dialog appears when attempting to generate type trees for tables in that database. The following example dialog reports an Oracle error ORA-01017 with the following message as returned by the database driver to the database adapter:

```
invalid username/password; logon denied
```

If trace is enabled when the message in this dialog appears, a corresponding message is also written to the database trace file (**.mdq_file_name.dbi**).

The following is a sample of a database trace file when generating a type tree that describes two tables in an Oracle database.

The contents of any trace file are database platform-specific.

```
<1776-940>: Datalink: bocadb2\\Northwind
<1776-940>: UserId : test
<1776-940>: Password:*****
<1776-940>: No existing connection was found.
<1776-940>: Local transaction usage: Transaction ID 0x019287F4
<1776-940>: Transaction started - ISOLATIONLEVEL_READCOMMITTED
<1776-940>: Connection to SQL Server bocadb2 has been established.
<1776-940>: Retrieving 1 rows per fetch.
<1776-940>: The columns are of the following types:
<1776-940>: Column 1 (CustomerID) type is nchar(5) [DBTYPE_WSTR].
<1776-940>: Column 2 (CompanyName) type is nvarchar(40) [DBTYPE_WSTR].
<1776-940>: Column 3 (ContactName) type is nvarchar(30) [DBTYPE_WSTR].
<1776-940>: Column 4 (ContactTitle) type is nvarchar(30) [DBTYPE_WSTR].
<1776-940>: Column 5 (Address) type is nvarchar(60) [DBTYPE_WSTR].
<1776-940>: Column 6 (City) type is nvarchar(15) [DBTYPE_WSTR].
<1776-940>: Column 7 (Region) type is nvarchar(15) [DBTYPE_WSTR].
<1776-940>: Column 8 (PostalCode) type is nvarchar(10) [DBTYPE_WSTR].
<1776-940>: Column 9 (Country) type is nvarchar(15) [DBTYPE_WSTR].
<1776-940>: Column 10 (Phone) type is nvarchar(24) [DBTYPE_WSTR].
<1776-940>: Column 11 (Fax) type is nvarchar(24) [DBTYPE_WSTR].
```

The following example was received when attempting to run a map to retrieve data from a table in an SQL Server database.

```
<1324-3020>: Validating the adapter command...
<1324-3020>: Database type is MS SQL Server 7
<1324-472>: Connecting...
<1324-472>: Datalink: MY_2000_SERVER\\test
<1324-472>: UserId : test
<1324-472>: Password:*****
<1324-472>: OLE DB Error code: 0x80004005
<1324-472>: [DBNMPNTW]Specified SQL server not found.
<1324-472>: Returned status: (-3) No error text found
```

In this example, the following lines:

```
<1324-472>: OLE DB Error code: 0x80004005
<1324-472>: [DBNMPNTW]Specified SQL server not found.
```

indicate the OLE DB provider-specific error code and the corresponding error description. The last line (beginning with `Returned status:`) indicates the error code returned by the adapter that caused the map to fail.

This information can be used, along with the SQL Server documentation, to resolve the problem. In this particular example, the error was caused by typing the incorrect server name in the SQL Server **Server** settings in the Database Definition dialog, rather than by selecting the name from the list of servers. In this example, the server name was incorrectly entered as `MY_2000_SERVER`, instead of its actual name (`MY_2000_SRVR`).

Producing the database trace during map execution

There are several different ways to enable the database trace for reporting database-related information during map execution as discussed in the following sections:

- Using the [Trace adapter](#) command
- Source usage:
 - for a [valid source](#)
 - for a [source with errors](#)
- Target usage:
 - for a [valid target](#)
 - for a target with a [missing required value](#)
 - for a target using the [Bad Data](#) adapter command
 - for a target with [-UPDATE off](#)
 - for a target using the [DBLOOKUP/DBQUERY](#) functions
- [Using the Trace adapter command](#)
- [Database trace for a valid source](#)
- [Database trace for a source with errors](#)
- [Database trace for a valid target](#)
- [Database trace for a target: missing required value](#)
- [Database trace for a target - using the bad data adapter command \(-BADDATA\)](#)

- [Database trace for a target with -UPDATE off](#)
- [Database trace for a target: DBLOOKUP/DBQUERY functions](#)

Using the Trace adapter command

About this task

To produce database trace information for specific database data sources or targets, use the Trace adapter command (-TRACE). For information about the syntax of this command, see the Resource Adapters documentation.

To produce the database trace file when using the DBQUERY or DBLOOKUP functions, use the Syntax2 format to call the function. For example, if output card 3 contains a DBLOOKUP function, call the function as

```
DBLOOKUP ( "SELECT ...", "-DBTYPE DB2 -SOURCE LAMBIN
-USER ARL97IN -TRACE") .
```

Database trace for a valid source

To produce the following database trace file example, we want to produce a database trace for input card 2 (which is a database). To do this, you must include the Trace adapter command (-TRACE) in the GET > Source > Command setting. Upon execution, database trace information is generated for input card 2; the example follows.

Note: Line numbers are for reference purposes only.

```
01 Database type is Oracle
02 Status returned to engine: (0) Success
03 No existing connection was found.
04 Connection to Oracle has been established.
05 Interface library version 6.0(140)
06 Data being retrieved for input card 2.
07 Database adapter version 6.0(140)
08 Starting a database unload...
09 Host string:
10 Userid      : demo
11 Password    : ****
12 Query       : SELECT I.* , P.LIST_PRICE, P.MIN_PRICE, P.START_DATE, P.END_DATE
13   FROM ITEM I, PRICE P, PRODUCT PR
14   WHERE I.PRODUCT_ID = P.PRODUCT_ID and PR.PRODUCT_ID = 15  I.PRODUCT_ID
15   ORDER BY PR.PRODUCT_ID
16 Query size : 189
17 Output is to a buffer.
18 Statement execution succeeded.
19 The columns are of the following types:
20   Column 1 (ORDER_ID) type is NUMBER(4) .
21   Column 2 (ITEM_ID) type is NUMBER(4) .
22   Column 3 (PRODUCT_ID) type is NUMBER(6) .
23   Column 4 (ACTUAL_PRICE) type is NUMBER(8,2) .
24   Column 5 (QUANTITY) type is NUMBER(8) .
25   Column 6 (TOTAL) type is NUMBER(8,2) .
26   Column 7 (LIST_PRICE) type is NUMBER(8,2) .
27   Column 8 (MIN_PRICE) type is NUMBER(8,2) .
28   Column 9 (START_DATE) type is DATE.
29   Column 10 (END_DATE) type is DATE.
30 Number of buffers in fetch array = 574
31 Writing results to a buffer.
32 Retrieved 543 records (34451 bytes).
33 Status returned to engine: (0) Success
34 Cleaning up and closing the transaction...
35 The transaction was successfully committed.
36 Status returned to engine: (0) Success
37 Commit was successful.
38 Database disconnect succeeded.
```

This sample database trace file (`map_name .dbl`) reveals important information, examples of which are described below.

- Lines 1-5 show information about the connection made, identifying the database type as Oracle as well as the version number of the interface library. In this example, there was no existing database connection and a new connection was successfully established.
- Line 6 indicates that the data is being retrieved for input card number 2.
- Line 7 identifies the version of the database adapter for Oracle. Lines 8-11 identify the information used to make the connection with the Oracle database: the host string, user ID, and password. Because this information is for an input card, there is a message indicating that a database unload is being started.
- Next, the database trace file indicates that the query successfully executed (as indicated by the following text on Line 18: `Statement execution succeeded`) and includes descriptions of the ten columns to be retrieved.
- The results of the query are written to the buffer. 543 records (rows) were retrieved using this query.

Use the Trace adapter command (-TRACE) for as many database source or targets as desired. Note that the information for later cards is appended to the default database trace file unless a file name is specified using the -TRACE filename option as described in the Resource Adapters documentation.

The exact information in the database trace file varies, depending upon the information returned by different entities including the database driver, database source versus target, and so on.

Database trace for a source with errors

The example in the preceding section provided database trace information for a database source with no problems. The following is an example of information you may see when there are errors while accessing the database source.

```
Database type is Oracle
Status returned to engine: (0) Success
No existing connection was found.
Connection to Oracle has been established.
Interface library version 6.0(140)
Data being retrieved for input card 1.
Database adapter version 6.0(140)
Starting a database unload...
Host string:
Userid      : demo
Password    : *****
Query       : SELECT * FROM BONUSS
Query size  : 20
Output is to a buffer.
Error in: oparse

        Message : ORA-00942: table or view does not exist

        Retrieved 0 records (0 bytes).

        Error returned to engine: (-17) Failed to parse SQL statement
Cleaning up and closing the transaction...
The transaction was successfully committed.
Status returned to engine: (0) Success
Commit was successful.
Database disconnect succeeded.
```

The key information in the database trace is in the line defining the query (beginning with `Query`) and the four lines beginning with the `Error in:` line. In this example, the attempt to execute the query failed because the table specified in the query listed does not exist. This would result in a map return code of 12 and a message of:

Source not available

Upon receiving this error, verify that the query is correct and that the table referenced in the query exists using the Database Interface Designer or the SQL command tool provided with your database.

Database trace for a valid target

The information included in the database trace for an output (data target) is similar to that for a database source. The following example shows the PUT > Target settings for the first output card (which is a database) in a map called **UpdateMembershipDB**.

The Update adapter command (`-UPDATE`) specifies that the rows produced by the map should update existing rows or should insert new rows, based upon the update keys configured for the table in the Database Interface Designer. The Trace adapter command (`-TRACE`) generates the database trace information for this output card.

An example of the database trace information produced for this map follows.

```
Database type is Oracle
Status returned to engine: (0) Success
No existing connection was found.
Connection to Oracle has been established.
Interface library version 6.0(140)
Loading data for output card 1.
Database adapter version 6.0(140)
Starting database load...
Host string:
Userid      : eventmgt
Password    : *****
Update mode is on.
The columns are of the following types:
Column 1 (MEMBERID) type is VARCHAR(10).
Column 2 (ATTENDEEFIRSTNAME) type is VARCHAR(30).
Column 3 (ATTENDEELASTNAME) type is VARCHAR(50).
Column 4 (TITLE) type is VARCHAR(50).
Column 5 (COMPANYNAME) type is VARCHAR(50).
Column 6 (ADDRESS) type is VARCHAR(255).
Column 7 (CITY) type is VARCHAR(50).
Column 8 (STATEORPROVINCE) type is VARCHAR(20).
Column 9 (POSTALCODE) type is VARCHAR(20).
Column 10 (COUNTRY) type is VARCHAR(50).
Column 11 (PHONENUMBER) type is VARCHAR(30).
Column 12 (FAXNUMBER) type is VARCHAR(30).
Column 13 (EMAILADDR) type is VARCHAR(50).
Column 14 (MEMBERSINCE) type is DATE.
The insert statement to be executed is:
INSERT INTO Membership VALUES
(:a00,:a01,:a02,:a03,:a04,:a05,:a06,:a07,:a08,:a09,:a10,:a11,:a12,:a13)
The update statement to be executed is:
UPDATE Membership SET ATTENDEEFIRSTNAME=:a01,ATTENDEELASTNAME=:a02,TITLE=:a03,
COMPANYNAME=:a04,ADDRESS=:a05,CITY=:a06,STATEORPROVINCE=:a07,
POSTALCODE=:a08,COUNTRY=:a09,PHONENUMBER=:a10,FAXNUMBER=:a11,EMAILADDR=:a12
WHERE (MEMBERID=:a00)5 rows inserted.

        2 rows updated.
Database load complete.
Status returned to engine: (0) Success
Cleaning up and closing the transaction...
The transaction was successfully committed.
Status returned to engine: (0) Success
```

```
Commit was successful.  
Database disconnect succeeded.
```

This sample database trace file (**map_name .dbt**) reveals important information that is highlighted in **bold** type above and some of which is described below.

- After the logon information in the sample, a message indicates that **Update mode is on**. This will affect how the rows produced are handled. Because update mode is on (enabled by the **-UPDATE** adapter command), rows in the table are inserted or updated based upon the update keys defined in the Database Interface Designer.
- After the connection is made, the database trace shows the column definitions for the output table.
- Next, the database trace file displays the actual SQL statements to be executed using the data produced for the output card. Because update mode is enabled, the database trace file displays both an **INSERT** statement and an **UPDATE** statement. The **UPDATE** statement is executed for each row in the output for which a corresponding row is found in the database table, using the update key defined (which, in this example, is the **MemberID** column). The **INSERT** statement is executed for all rows in the output for which a corresponding row does not exist in the table.
- After the SQL statements in the sample, the database trace file indicates the number of rows that were both inserted (5) and updated (2).
- Finally, the database trace file indicates that database load was executed successfully (**Database load complete**) and that the database changes were committed (**The transaction was successfully committed**).

Database trace for a target: missing required value

The following database trace file example was produced by using almost the same map as in the preceding example except for the usage of a different input data file. This database trace file illustrates the information you might see if a database error occurred while attempting to insert or update rows in a table.

```
Database type is Oracle  
Status returned to engine: (0) Success  
No existing connection was found.  
Connection to Oracle has been established.  
Interface library version 6.0(140)  
Loading data for output card 1.  
Database adapter version 6.0(140)  
Starting database load...  
Host string:  
Userid      : eventmgt  
Password    : *****  
Update mode is on.  
The columns are of the following types:  
Column 1 (MEMBERID) type is VARCHAR(10).  
Column 2 (FIRSTNAME) type is VARCHAR(30).  
Column 3 (LASTNAME) type is VARCHAR(50).  
Column 4 (TITLE) type is VARCHAR(50).  
Column 5 (COMPANYNAME) type is VARCHAR(50).  
Column 6 (ADDRESS) type is VARCHAR(255).  
Column 7 (CITY) type is VARCHAR(50).  
Column 8 (STATEORPROVINCE) type is VARCHAR(20).  
Column 9 (POSTALCODE) type is VARCHAR(20).  
Column 10 (COUNTRY) type is VARCHAR(50).  
Column 11 (PHONENUMBER) type is VARCHAR(30).  
Column 12 (FAXNUMBER) type is VARCHAR(30).  
Column 13 (EMAILADDR) type is VARCHAR(50).  
Column 14 (MEMBERSINCE) type is DATE.  
The insert statement to be executed is:  
INSERT INTO Membership VALUES  
(:a00,:a01,:a02,:a03,:a04,:a05,:a06,:a07,:a08,:a09,:a10,:a11,:a12,:a13)  
The update statement to be executed is:  
UPDATE Membership SET FIRSTNAME=:a01,LASTNAME=:a02,TITLE=:a03, COMPANYNAME=:a04,  
ADDRESS=:a05,CITY=:a06,STATEORPROVINCE=:a07, POSTALCODE=:a08,COUNTRY=:a09,  
PHONENUMBER=:a10,FAXNUMBER=:a11,EMAILADDR=:a12 WHERE (MEMBERID=:a00)  
Error in: oexec  
  
Message : ORA-01400: cannot insert NULL into  
("EVENTMGT"."MEMBERSHIP"."LASTNAME")  
  
The following values were being inserted:  
Column 1 MEMBERID : D190-0002  
Column 2 FIRSTNAME : Leverling  
Column 3 LASTNAME : NULL  
Column 4 TITLE : Vice President-New Products  
Column 5 COMPANYNAME : Northwind Traders  
Column 6 ADDRESS : 722 Moss Bay Blvd.  
Column 7 CITY : Kirkland  
Column 8 STATEORPROVINCE : WA  
Column 9 POSTALCODE : 98033  
Column 10 COUNTRY : USA  
Column 11 PHONENUMBER : (206) 555-3412  
Column 12 FAXNUMBER : (206) 555-3413  
Column 13 EMAILADDR : jleverling@northwind.com
```

```

Column 14 MEMBERSINCE : 1999-07-09 22:03:03

Failed to insert a row (rc = -9).

Failed after 1 rows inserted.

Database load complete.

Error returned to engine: (-9) Failed to execute the SQL statement

Cleaning up and closing the transaction...

Transaction rollback was successful.

Status returned to engine: (0) Success

Commit was successful.

Database disconnect succeeded.

```

In this example, the **bold** text is information that can be provided when an error occurs. The first lines of this example provide the same information as was in the database trace file with no errors: connection parameters, update mode indicator, table column descriptions, SQL statements to be used, and so on.

However, after the UPDATE statement, the database trace file provides details to assist in determining the problem. The Message line indicates that an attempt was made to insert a row with a NULL value for the **EVENT.MEMBERSHIP.LASTNAME** column that cannot contain a NULL. To further identify the row in the output data causing the error, the database trace file lists the values for all of the columns in the row causing the error.

The remaining lines in the database trace file display information about the disposition of the entire database card transaction. The transaction failed after one row was inserted. The database adapter returns an error code of -9 to the Launcher with a corresponding error message of Failed to execute the SQL statement. If you produced the audit log, you would see the following line in the execution summary section:

```

<TargetReport card="1" adapter="DB" bytes="1218" adapterreturn="-9">
  <Message>Failed to execute the SQL statement</Message>
  <TimeStamp>22:07:32 January 8, 2004</TimeStamp>
</TargetReport>

```

Subsequently, the database trace file communicates that the database adapter is cleaning up and closing the transaction. Because the **OnFailure** setting was set to **Rollback** for this output card, the final entry in the database trace reveals that the transaction rollback was successful.

Database trace for a target - using the bad data adapter command (-BADDATA)

About this task

Use the Bad Data adapter command (-BADDATA) for a target (or in a PUT function). If any inserts, updates, or procedure calls fail to execute, the data that could not be processed is written to a file that you specify and processing continues.

The following database trace example was produced using the same map and data as in the preceding example, now additionally using -BADDATA.

```

.
.
Error in: oexec
Message : ORA-01400: cannot insert NULL into
("EVENTMGT"."MEMBERSHIP"."LASTNAME")
The following values were being inserted:
Column 1 MEMBERID : D190-0002
Column 2 FIRSTNAME : Leverling
Column 3 LASTNAME : NULL
Column 4 TITLE : Vice President-New Products
Column 5 COMPANYNAME : Northwind Traders
Column 6 ADDRESS : 722 Moss Bay Blvd.
Column 7 CITY : Kirkland Column 8 STATEORPROVINCE : WA
Column 9 POSTALCODE : 98033
Column 10 COUNTRY : USA
Column 11 PHONENUMBER : (206) 555-3412
Column 12 FAXNUMBER : (206) 555-3413
Column 13 EMAILADDR : jleverling@northwind.com
Column 14 MEMBERSINCE : 1999-07-09 22:03:03
Failed to insert a row (rc = -9).
4 rows inserted.
2 rows updated.
1 rows were rejected.
Database load complete.
Warning returned to engine: (1) One or more records could not be processed
Cleaning up and closing the transaction...
The transaction was successfully committed.
Status returned to engine: (0)
SuccessCommit was successful.
Database disconnect succeeded.

```

All of the information in this trace file example is the same as the previous example except for the bold lines of text. In this example, because -BADDATA was used, the database adapter goes on to process all of the other rows produced for the target. Upon completion, four rows were successfully inserted, two rows were successfully updated, and one row was rejected. The rejected record was saved in the file specified with the Bad Data adapter command (-BADDATA). In this example, the rejected record was saved to a file called badstuff.txt.

The remaining lines in the database trace file indicate information about the disposition of the entire database card transaction. Notice that the use of -BADDATA allows the card to successfully complete. The map successfully completes and the transaction is committed.

If you produced the audit log, you would see the following line in the execution summary section:

```

<TargetReport card="1" adapter="DB" bytes="1218" adapterreturn="1">
  <Message>One or more records could not be processed</Message>
  <TimeStamp>22:07:32 January 8, 2004</TimeStamp>
</TargetReport>

```

Database trace for a target with -UPDATE off

The following example illustrates the database trace that can result from a common error encountered when developing maps have database targets.

```

Database type is ODBC
Status returned to engine: (0) Success
No existing connection was found.
Connection to datasource OracleProd has been established.
Interface library version 6.0(140)
Loading data for output card 1.
Database adapter version 6.0(140)
Starting database load...
Datasource: OracleProd
Userid   : EVENTMGT
Password : *****
Update mode is off.
The columns are of the following types:
Column 1 (MEMBERID) type is VARCHAR, precision is 10.
Column 2 (ATTENDEEFIRSTNAME) type is VARCHAR, precision is 30.
Column 3 (ATTENDEELASTNAME) type is VARCHAR, precision is 50.
Column 4 (TITLE) type is VARCHAR, precision is 50.
Column 5 (COMPANYNAME) type is VARCHAR, precision is 50.
Column 6 (ADDRESS) type is VARCHAR, precision is 255.
Column 7 (CITY) type is VARCHAR, precision is 50.
Column 8 (STATEORPROVINCE) type is VARCHAR, precision is 20.
Column 9 (POSTALCODE) type is VARCHAR, precision is 20.
Column 10 (COUNTRY) type is VARCHAR, precision is 50.
Column 11 (PHONENUMBER) type is VARCHAR, precision is 30.
Column 12 (FAXNUMBER) type is VARCHAR, precision is 30.
Column 13 (EMAILADDR) type is VARCHAR, precision is 50.
Column 14 (MEMBERSINCE) type is TIMESTAMP, precision is 19.

The insert statement to be executed is:
INSERT INTO Membership VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?)
Error in SQLExecute
Message: ORA-00001: unique constraint (SYSTEM.UNIQUE_MEMBER_ID) violated
SQL State: 23000
The following values were being inserted:
Column 1 MEMBERID : 1978-1964
Column 2 ATTENDEEFIRSTNAME : Jean
Column 3 ATTENDEELASTNAME : Fresnière
Column 4 TITLE : Marketing Assistant
Column 5 COMPANYNAME : Mère Paillarde
Column 6 ADDRESS : 43 rue St. Laurent
Column 7 CITY : Montréal
Column 8 STATEORPROVINCE : Québec
Column 9 POSTALCODE : H1J 1C3
Column 10 COUNTRY : Canada
Column 11 PHONENUMBER : (514) 555-8054
Column 12 FAXNUMBER : (514) 555-8055
Column 13 EMAILADDR : fresnierej@paillarde.org
Column 14 MEMBERSINCE : {ts '1999-07-09 23:18:04'}
Failed to insert a row (rc = -9).

Failed after 2 rows inserted.

Database load complete.

Error returned to engine: (-9) Failed to execute the SQL statement

Cleaning up and closing the transaction...

Transaction rollback was successful.

Status returned to engine: (0) Success

Commit was successful.

Database disconnect succeeded.

```

If you forget to specify the usage of the update setting (using **-UPDATE** either in the PUT > Target > Command setting in the Map Designer or Integration Flow Designer or in the command line), you may receive a database error resulting from the attempt to insert a row with a duplicate index.

The first entry in this example highlighted in **bold** type indicates that update mode is off. Because update mode is off, the database adapter attempts to insert a row into the database table for each row produced by the map. The **Message** entry contains the message returned by the database driver to the database adapter describing the cause of the error. In this example, the row in error would violate the **UNIQUE_MEMBER_ID** constraint defined for the table. The next lines show the column values for the row in which the error occurred and the final result of the database operation.

There are several possible methods for resolving this problem. Depending upon the desired behavior, you might:

- Enable update mode when executing the map by using the **-UPDATE ON** or **-UPDATE ONLY** adapter command.
- Use the Delete adapter command (**-DELETE**) to remove all rows from the output database table before inserting the rows resulting from map execution.
- Build logic into your map to ensure that there is no existing row in the table prior to generating an output row to be inserted. This might be accomplished by either defining a query for the table being used as an input against which a **LOOKUP** or **SEARCHUP** function is performed or by using the **DBLOOKUP** function to check for an

existing row.

- Use -BADDATA so that all rows with a unique **MEMBERID** are inserted. Those rows that would result in duplicate rows are then saved to a specified file.

Database trace for a target: DBLOOKUP/DBQUERY functions

The following is a selection from a database trace file (**map_name dbl**) for an output card using the GET function to execute a DELETE statement based upon a value in the input.

```
..Starting a database unload...
Host string:
Userid      : eventmgt
Password    : *****
Query       : DELETE FROM REGISTRATION WHERE REGISTRATIONSTATUS = 'Overflow'
Query size : 71Output is to a buffer.
Statement execution succeeded.
Retrieved 0 records (0 bytes).
Size of DBLOOKUP data is 0.
Warning returned to engine: (2) No data found
Cleaning up and closing the transaction.....
```

The following SQL statement to be executed:

```
DELETE FROM REGISTRATION WHERE REGISTRATIONSTATUS = 'Overflow'
```

and the resulting successful execution statement:

```
Statement execution succeeded
```

are included in this trace file as shown in the example above. Note that no records were retrieved because the DELETE statement does not return data.

The next series of statements are produced for an output card using a DBQUERY function to obtain several columns from a database table based upon specified values in the **MEMBERID** column in the input data.

```
..Starting a database unload...
Host string:
Userid      : eventmgt
Password    : *****
Query       : SELECT FIRSTNAME, LASTNAME , EMAILADDR FROM EVENTMGT.MEMBERSHIP
              WHERE MEMBERID = 'D191-0001'
Query size : 108
Output is to a buffer.
Statement execution succeeded.
The columns are of the following types:
  Column 1 (FIRSTNAME) type is VARCHAR(30).
  Column 2 (LASTNAME) type is VARCHAR(50).
  Column 3 (EMAILADDR) type is VARCHAR(50).
Number of buffers in fetch array = 492
Writing results to a buffer.
Retrieved 1 records (40 bytes).
Size of DBQUERY data is 40.
The following data was returned from a DBQUERY:
```

```
Nancy|Davolio|nancyd@cascadecoffee.com
```

```
Status returned to engine: (0) Success
Status returned to engine: (0) Success
Interface library version 6.0(140)
Data being retrieved for DBQUERY function.
Database adapter version 6.0(140)
Starting a database unload...
Host string: Userid      : eventmgt
Password    : *****
Query       : SELECT FIRSTNAME, LASTNAME , EMAILADDR FROM EVENTMGT.MEMBERSHIP
              WHERE MEMBERID = 'D191-0002'
Query size : 108
Output is to a buffer.
Statement execution succeeded.
The columns are of the following types:
  Column 1 (FIRSTNAME) type is VARCHAR(30).
  Column 2 (LASTNAME) type is VARCHAR(50).
  Column 3 (EMAILADDR) type is VARCHAR(50).
Number of buffers in fetch array = 492
Writing results to a buffer.
Retrieved 1 records (42 bytes).
```

```
Size of DBQUERY data is 42.
```

```
The following data was returned from a DBQUERY:
```

```
Janet|Leverling|jleverling@northwind.com
```

In this example, as in the previous one, the database trace file reports the actual query to be executed. In this example, each DBQUERY returns a single row containing the three columns from the **Membership** table for the specified **MEMBERID**. The data returned by the DBQUERY function conforms to the **Delimited Row group format** type tree definition, meaning that there will be a pipe character (|) delimiter between columns and a carriage return/line feed terminator for each row of data.

The size of the data returned by the DBQUERY is also listed. Note that the size of the DBQUERY data is specified as 42. This value represents the length of the data, including the carriage return/line feed that follows the row of data. There will be a similar set of entries in the database trace file for each DBQUERY executed.

Tracing database errors only

Depending upon the number of database sources and targets, the volume of database functions that are executed, and the database trace settings that are used, the database trace file can contain a large amount of data. Even when no database errors occur, the database trace file can become very large.

Use the Trace Error adapter command (-TRACEERR) to minimize the amount of information contained in the database trace file (**map_name.dbt**). This file contains only the database errors occurring during the map execution.

When -TRACEERR is specified, the database trace file is produced using the full name of the MDQ with a **.dbt** file name extension instead of the usual trace (**.mtr**) extension. It is created in the directory in which the map is located. Use the -TRACEERR *filename* syntax as described in the *Resource Adapters documentation* to specify the name for the database trace file.

A full database trace is most beneficial during map development; however, it is preferable to use -TRACEERR in a production environment because only the database errors are reported.

For example, if you generate the full database trace for a map called **DBUpdate** that has three database input cards, it would produce the following 88-line **MultiDB's.dbt** file.

```
Database type is MS SQL Server
Status returned to engine: (0) Success
No existing connection was found.
Error in: dbopen
Error   : 5701
State   : 2
Message : Changed database context to 'master'.
Error in: dbuse
Error   : 5701
State   : 1
Message : Changed database context to 'pubs'.
A transaction was started.
Connection to SQL Server has been established.
Interface library version 6.0(140)
Data being retrieved for input card 1.
Database adapter version 6.0(140)
Starting a database unload...
Server\\Database: HP_NT\\pubs
Userid      : sa
Password    :
Query       : SELECT * FROM Authors
Query size   : 21
Output is to a buffer.
Statement execution succeeded.
The columns are of the following types:
Column 1 (au_id) type is varchar(11).
Column 2 (au_lname) type is varchar(40).
Column 3 (au_fname) type is varchar(20).
Column 4 (phone) type is char(12).
Column 5 (address) type is varchar(40).
Column 6 (city) type is varchar(20).
Column 7 (state) type is varchar(2).
Column 8 (zip) type is varchar(5).
Column 9 (contract) type is bit.
Number of buffers in fetch array = 1
Writing results to a buffer.
Retrieved 23 records (1810 bytes).
Status returned to engine: (0) Success
Database type is MS SQL Server
Status returned to engine: (0) Success
Interface library version 6.0(140)
Data being retrieved for input card 2.
Database adapter version 6.0(140)
Starting a database unload...
Server\\Database: HP_NT\\pubs
Userid      : sa
Password    :
Query       : SELECT * FROM Publishers
Query size   : 24
Output is to a buffer.
Statement execution succeeded.
The columns are of the following types:
Column 1 (pub_id) type is char(4).
Column 2 (pub_name) type is varchar(40).
Column 3 (city) type is varchar(20).
Column 4 (state) type is varchar(2).
Column 5 (country) type is varchar(30).
Number of buffers in fetch array = 1
Writing results to a buffer.
Retrieved 8 records (305 bytes).
Status returned to engine: (0) Success
Database type is MS SQL Server
Status returned to engine: (0) Success
Interface library version 6.0(140)
Data being retrieved for input card 3.
Database adapter version 6.0(140)
Starting a database unload...
Server\\Database: HP_NT\\pubs
Userid      : sa
```

```

Password      :
Query         : SELECT * FROM Title
Query size    : 19
Output is to a buffer.
Error in: dbsqlexec

        Error   : 208
        State   : 1
        Message : Invalid object name 'Title'.
        Error in: dbsqlexec

        Error   : 10007
        Message : General SQL Server error: Check messages from the SQL Server.

        Failed to execute statement.

        Invalid object name 'Title'.
Retrieved 0 records (0 bytes).
Error returned to engine: (-9) Failed to execute the SQL statement
Cleaning up and closing the transaction...
The transaction was successfully committed.
A transaction was started.
Status returned to engine: (0) Success

```

After reviewing this file, notice that the query for input card number 3 (SELECT * FROM Title) failed because it references an object name (the table name **Title**) that does not exist. Alternatively, you could use the Trace Error adapter command (-TRACEERR) for each of the three input cards and produce the following database trace file:

```
Invalid object name 'Title'.
```

This file can be used along with the following execution audit log information:

```

<SourceReport card="1" adapter="DB" bytes="1810" adapterreturn="0">
  <Message>Success</Message>
  <TimeStamp>01:02:00 January 9, 2004</TimeStamp>
</SourceReport>

<SourceReport card="2" adapter="DB" bytes="305" adapterreturn="0">
  <Message>Success</Message>
  <TimeStamp>01:02:00 January 9, 2004</TimeStamp>
</SourceReport>

<SourceReport card="3" adapter="DB" bytes="0" adapterreturn="-9">
  <Message>Failed to execute the SQL statement</Message>
  <TimeStamp>01:02:00 January 9, 2004</TimeStamp>
</SourceReport>

```

Use this information to determine whether the query defined for input card 3 references a table or view that does not exist.

Viewing database source and target data

When debugging a map that uses database sources or targets, you cannot view the database source or target data in the Map Designer by selecting Run Results from the View menu. Because the data retrieved from a database for an input or written to a database as an output is a snapshot of the data at a given time, it is not available to the Map Designer after the map runs.

However, you can use **Backup** settings to capture the data retrieved from or written to the database for debugging purposes.

- [Using backup settings](#)
- [Capturing data not processed](#)

Using backup settings

Backup settings are used to determine when, where, and how the data for a specific card should be copied to a specified backup file. These settings are configured in the **Input Card** and **Output Card** settings in the Map Designer and the Launcher or in Command Settings for the Integration Flow Designer. For more information about these settings, see the Map Designer documentation.

Capturing data not processed

The Bad Data adapter command (-BADDATA) can be used for a target (or in a **PUT** function). If any inserts, updates, or procedure calls fail to execute, the data that could not be processed is written to a file you specify and processing continues.

The -BADDATA command can be specified:

- as part of the **PUT** Target > Command setting for a card target
- as part of the command line for a database target referenced in a **RUN** or **PUT** function
- on the command line or in a command file.

The rejected record(s) is/are saved in the file specified with the -BADDATA adapter command.

When the `-BADDATA` adapter command is used and one or more rows are rejected, the database adapter returns an adapter return code of 1 with a message of One or more records could not be processed as shown in the following TargetReport excerpt in the execution summary section:

```
<TargetReport card="1" adapter="DB" bytes="1218" adapterreturn="1">
  <Message>One or more records could not be processed</Message>
  <TimeStamp>22:07:32 January 9, 2004</TimeStamp>
</TargetReport>
```

Database audit files

Additional troubleshooting and diagnostic information is available in the database adapter audit. Use the Audit adapter command (`-AUDIT`) to create a file that records the adapter activity for each specified database activity. This command can be used for a source or target, or in a `DBLOOKUP`, `DBQUERY`, `GET`, or `PUT` function. This adapter command can be specified for individual input and output cards on a card-by-card basis or, optionally, as a global audit that encompasses all database activity for the entire map.

The default is to produce a file named `audit.dbl` in the directory in which the map is located. Optionally, you can append the audit information to an existing file or specify a name or the full path for the file. For more information, see the Resource Adapters documentation.

The database adapter audit file provides the following details for each database access:

- **Execution Time (Audit Time)**
This is the amount of clock time (in seconds) the database adapter takes to execute the database action (for example, the retrieval of all rows for a data source, a single instance of `ADBLOOKUP` function, and so on).
- **Adapter Return Code (AC)**
This is the adapter return code as a result of executing the database action.
- **Connection**
This information identifies the connection used for each database action. This can be helpful in determining ways to configure your map to minimize the number of database connections that must be made.
- **Map Name (Map)**
This is name of the compiled map file.
- **Access Type (Input Card, Function, or Output Card)**
This information identifies the type of database action (for example, whether the type is for an input card, output card, or for a `DBLOOKUP`, `DBQUERY`, `GET`, or `PUT` function).
- **Additional Information**
This information is provided (when appropriate) for each database action such as the SQL statement associated with an input card, the name of the table or stored procedure for a database target, or the SQL statement executed by any of the functions for interfacing with data in a database (for example, `DBLOOKUP`, `DBQUERY`, `PUT`, and `GET`).

DBMS trace utilities and SQL command tools

The utilities and tools that are part of your relational database management system (RDBMS) will also be helpful during the troubleshooting process. For example, a map fails at runtime because a table name was invalid in an input that is a database source. Try to execute that same query using the tools included with your database to determine whether the query will run natively.

- [Trace utilities](#)
- [SQL command tools](#)

Trace utilities

For example, when using ODBC data sources on a Windows platform, enable ODBC tracing from the ODBC Data Source Administrator window. This creates a log of the calls to ODBC drivers as you use the Database Interface Designer to define databases and queries, and when you use a Launcher to run maps that have database sources or targets. Similar tracing tools are available for most of the other database systems. For information about the tracing capabilities available for your database system, see the documentation for your RDBMS.

SQL command tools

Each database management system includes some form of an SQL command tool. For example, ODBC data sources can be accessed using tools such as Microsoft Query to test queries and view information about your tables, views, and stored procedures. Oracle databases can be accessed using SQL Plus; Microsoft SQL Server data sources can be accessed with ISQL.

These tools provide the ability to determine the problem by testing your queries against the database using the same drivers being accessed by the database adapters.

For example, if you run a map that executes a query for a data source and it fails because one or more of the column names was invalid, you could copy the query text from the Database Interface Designer and test it using the SQL command tool for your database. Then, you could modify the query as necessary to get it to work correctly. When you achieve the expected results, copy the SQL statement back to the query defined in the Database Interface Designer.

Similarly, if a database insert or update operation fails, you could try entering the corresponding INSERT or UPDATE statement into your database system's SQL command tool to help determine the cause of the failure.

REST API V2.0

- **[Deployment APIs](#)**

Deployment in v2 uses SQLite files generated by tx-server during deployment. These files contain all the objects (flows, maps, and files) defined in the definition of the package being deployed.

- **[Map and Flow run APIs](#)**

These APIs allow you to run maps or flows that have URLs specified for them in the package definition.

- **[Flow Schedule APIs](#)**

Flow Schedule APIs allow you to control the scheduler properties for a flow.

Deployment APIs

Deployment in v2 uses SQLite files generated by tx-server during deployment. These files contain all the objects (flows, maps, and files) defined in the definition of the package being deployed.

After the file has been sent to the tx-rest server, it is merged into the master SQLite file for that server which can be found at the location specified for the rest.persistence.catalog property in config.yaml.

Packages can be started, paused, and stopped. They are automatically started when they are deployed. After they are started, the flows, maps, and files within the SQLite file are unpacked to the file system under a directory named after the package located inside the directory specified for the property rest.persistence.maps in config.yaml. If multiple locations are specified, the first one is used.

When stopped, the files are removed from the file system and the flows and maps cannot be run until it is started again.

If the package is paused, maps and flows can still be invoked through the REST API but listeners are paused until the package is un-paused.

These APIs in this table can be called:

API	Description
GET /v2/packages/{package_name}	Returns detailed information about the specified package.
DELETE /v2/packages/{package_name}	Deletes the specified package. This API has a Boolean query parameter stop that if set to true will allow you to delete a package that has been started, otherwise the package must first be stopped.
GET /v2/packages	Returns basic information about each deployed package. Has a Boolean query parameter details . If set to true will return more detailed information for each package.
POST /v2/packages	Deploys a package. Takes an SQLite file containing a package as the request body. Has a Boolean query parameter start which will cause the package to automatically start after it is deployed
PUT /v2/{package_name}/actions/start	Starts the specified package if it is stopped or paused.
PUT /v2/{package_name}/actions/stop	Stops the specified package if it is started or paused.
PUT /v2/{package_name}/actions/pause	Pauses the specified package if it is started.
PUT /v2/{package_name}/actions/resume	Un-pauses the specified package if it is paused.

Map and Flow run APIs

These APIs allow you to run maps or flows that have URLs specified for them in the package definition.

- **[PUT /v2/run/url](#)**

Synchronously run a map or flow whose package has been started. The result contains the data of each output card that was set as a REST output for a map, or each flow output terminal defined for a flow.

- **[POST /v2/run/url](#)**

This API asynchronously runs a map or flow whose package has been started.

PUT /v2/run/url

Synchronously run a map or flow whose package has been started. The result contains the data of each output card that was set as a REST output for a map, or each flow output terminal defined for a flow.

If there is only one, it will be returned as a stream of bytes, otherwise, the multiple outputs will be joined into a multipart/form-data response. These requests will timeout after a timeout length specified in the synchronousTimeout property in config.yaml.

The status and log can also be specified as values for the return query parameter for both maps and flows, which return a status message about the run and the audit for a map, respectively.

For a flow, status and log both just return a status message, but with log specified, that status message contains the flow audit. Maps can also use the value trace which returns the map trace. Multiple values for the return parameter can be specified at one time.

POST /v2/run/url

This API asynchronously runs a map or flow whose package has been started.

The result contains an href that gives information on the status of the run when a GET request is sent to it and an id for a run instance which can be used with the map and flow instance APIs defined in this table:

Map and flow APIs	Description
GET /v2/instances/{id}	Get the status of an asynchronous map or flow run. The id is the one returned from running a map or flow asynchronously.
DELETE /v2/instances/{id}	Delete the instance of an asynchronous map or flow run, these instances will persist until this API is invoked. The id is the one returned from running a map or flow asynchronously
GET /v2/instances/{id}/outputs/{output_number}	Get the data for an output card or output flow terminal of an asynchronous map or flow run. The id is the one returned from running a map or flow asynchronously, and the output_number is the number of the output to get the data for.
/v2/instances/{id}/log	Get the log of an asynchronous map or flow run. The id is the one returned from running a map or flow asynchronously
/v2/instances/{id}/trace	Get the trace of an asynchronous map run. The id is the one returned from running a map asynchronously

It is important to note that status and log can also be specified as values for the return query parameter for both maps and flows, which return a status message about the run and the audit for a map, respectively.

For a flow, status and log both return a status message, but with log specified, that status message contains the flow audit. Maps can also use the value trace which returns the map trace. Multiple values for the return parameter can be specified at the same time.

Each REST input and output for maps or flow input and output terminal for flows, can be overridden by specifying the input and output query parameters. The accepted values for these parameters are in the form <card/terminal-number>;<adapter-name>;<adapter-cmd>. Multiple overrides can be provided by separating each with a comma.

Flow Schedule APIs

Flow Schedule APIs allow you to control the scheduler properties for a flow.

- [POST /v2/schedule/url](#)
Set scheduler properties for a flow. Any flow variables that the flow should use can be passed in the flow_vars query parameter which takes flow variables as a JSON string.
- [DELETE /v2/schedule/url](#)
Unschedule a flow. When this is invoked on a flow with a schedule, it no longer runs according to its schedule until new schedule properties are set to it.

POST /v2/schedule/url

Set scheduler properties for a flow. Any flow variables that the flow should use can be passed in the flow_vars query parameter which takes flow variables as a JSON string.

For example, {"myVariable":"sampleValue","otherVariable":"otherValue"}

The request body should contain a JSON object specifying the desired scheduler properties to be set. These properties are listed below:

- start – A string representing the time of day that the flow should start being executed which should be a string in the form "HH:MM". If specified, 'end' must also be specified.
- end – A string representing the time of day that the flow should stop being executed. This is a string in the form HH:MM. If specified, start must also be specified.
- days – An array of integers between 1 and 7 which represent the days on which the flow should be run, with Monday being a 1 and Sunday being a 7. If not specified, the flow will run every day.
- unit - The unit of time for the scheduler. Can be hours, minutes, or seconds. If specified, interval must also be specified.
- interval – An integer that together with unit determines how often the flow will execute. If specified, unit must also be specified.
- skipifbusy – A Boolean expression, which is treated as true by default, which determines whether the flow should execute if an instance of it is already being executed.
- timezone – A string representing the timezone of the start and end times.

DELETE /v2/schedule/url

Unschedule a flow. When this is invoked on a flow with a schedule, it no longer runs according to its schedule until new schedule properties are set to it.

Flow functions

Using the Flow functions, you can specify a value for each of the flow variables defined for a flow, and additionally, alter an already-specified value for a flow variable, if required. Flow variables act as an alternative means to pass information through and through a flow instance—thereby eliminating the limitations and cumbersomeness of nodes being the only gateway for passage of information.

Flow variables are name-value pairs that persist until the flow-instance that is using these variables is executed. Flow variables are case-sensitive. So, you need to be particular about the case you are using while defining a flow variable. Flow variables are also instance-specific; that is, no two flow-instances that are running simultaneously can share the same set of variables. However, a set of flow variables defined for one flow instance can be used by another as long as the other instance is not executed during the runtime of the instance for which the variables are originally defined.

See the Flow variables documentation contained in the Functions and Expressions sections of this online documentation.

Using a Command Server

To use the Command Server to execute a map, you will prepare the maps for execution.

For installation information, see the [release notes](#). Before using the Command Server after you have installed it, test your installation by running an example map. To see how to test your installation on a specific platform, see the platform-specific topic.

- [Command Server Overview](#)
- [Executing a map](#)
- [Troubleshooting tools](#)
- [External interfaces for applications, EXIT and RUN](#)

Maps can interface with other maps, functions in libraries, or programs using the `EXIT` function, the `RUN` function or an adapter as a data source or target.

Command Server Overview

The Command Server is used to develop, test, and execute maps in development environments. It can also be used to execute commands in production environments.

Maps are developed and tested using the Design Studio software on a Windows system-based development computer. When a map's development is complete, the compiled map file (`.mmc`) is, by default, ready to be executed by a Windows system-based Command Server. For example, you can run a map from the Map Designer interface or a Windows system-based Command Server, a Launcher, or the Platform API that has the same byte-order and character set.

The following list summarizes the steps you will do to run maps with the Command Server.

1. Create your map
2. Build or build for specific platform
3. Transfer if necessary
4. Execute with the Command Server

To account for environmental differences, some platforms require the map to be compiled or built for that specific platform. The **Build for Specific Platform** command is available from the Map menu in the Map Designer. After a map is built, it can be transferred to any destination computer hosting the Command Server. Using your preferred file transfer method, the file must be transferred in BINARY mode to ensure that the content of the compiled map arrives in its proper format on the destination platform.

Maps already transferred to their destination platform can be executed:

- from the command line
- as part of a series of maps in a command file
- from a program that runs an external program
- using a batch file or shell script
- as a task within a scheduler
- [Building a platform-specific map](#)

Building a platform-specific map

After creating a map in Map Designer, you can build a map for a specific execution platform. Select a specific platform under Map > Build for Specific Platform.

When a map is built for a specific platform, the name of the compiled map file has a platform-specific file name extension that indicates the execution platform for the map. The following table lists each platform and the corresponding platform-specific filename extension.

Platform	Map File Name Extension
HP-UX (Itanium processor)	.hpi
IBM® AIX® (RS/6000® processor)	.aix
IBM z/OS®	.mvs
Linux® (Intel processor)	.lnx

```
Microsoft Windows  
  .mmc  
pLinux  
  .pnx  
Sun Solaris (SPARC processor)  
  .sun  
zLinux  
  .znx
```

After transferring a platform-specific built map, it is strongly recommended that you rename the compiled map by using the .mmc file extension. Renaming it is not necessary; however, it is helpful because an .mmc extension identifies the file name in mapping functions and the renamed map can be executed without specifying the extension.

Executing a map

After a map has been built for a specific execution platform and transferred, the map can be executed using a Command Server.

- [Using execution commands](#)
 - [Input data validation](#)
 - [Output objects built](#)
-

Using execution commands

Using the execution commands and options, you have precise control over how a map executes. For example, you can override sources and targets specified in the map, create trace and audit log files, and control a variety of characteristics of the run-time environment.

Some execution commands are not available on all platforms. For information about execution commands and their syntax, see the *Execution Commands* documentation.

Input data validation

When a map begins execution, the input data must first be validated to ensure that the data for each input conforms to the definition of the corresponding input card. If it does conform, the data is valid. If the data does not conform, it is invalid. If data is determined to be invalid, a run-time return code and corresponding message is issued.

If invalid data is encountered, you may still want the map to continue processing or you may even want to map the invalid data. This can be accomplished during map development by using the **Restart** attribute when defining the input data and using the **REJECT** function when mapping the output data. In addition, functions are available to test for the presence of errors. For information about using the **Restart** attribute, see the *Type Designer* documentation. For information about using the error-related functions, see the *Functions and Expressions* documentation.

The validation process assumes that source data is a data stream. To find data objects in the input stream, type tree definitions are used including information about syntax items, data patterns, restrictions, component rules, and size. To process your input, the data must be able to be accessed at a designated position, for a specified length.

As data is validated, information about the data is recorded into workspace in memory or work files that guide the output process. The workspace information controls the data objects appearing in the data stream, whether they are related to other data objects, and what (if any) the relationship is. This enables you to specify any input within a map rule regardless of where the data appears in the input stream. It also enables you to use rules about the presence of data so that, for example, you can count objects or test their presence.

Output objects built

After all the input data has been validated and the input data is valid, output objects are generated based on each output card specification, including the map rules. If the output has an initiator defined, it is built first. Then, each output is built, one rule at a time, beginning with the first rule of the first output card. The entire first rule is evaluated, even if it refers to other maps and functions. When the first output is built, it is a complete data object, identified and treated as any other input data object is identified and treated. This enables you to refer to earlier outputs in a map rule.

As outputs are built, syntax objects are inserted—such as a delimiter, initiator, terminator, or release character—according to the type definition, and whatever remaining bytes of a fixed-length item are filled with pad characters. The execution process continues as the rules are evaluated and output data is built through the completion of the last rule of the last card.

Troubleshooting tools

When a map begins to run, the first thing that occurs is the validation of the input data—performed by comparing the data to the type tree definition. After data validation completes, the map rules are evaluated for the output cards and output objects are built.

When a map executes, invalid or unknown input data may be encountered and you may want to analyze what happened during the validation process. You may also want to analyze the output data if the map did not produce the expected results.

Many debugging aids are available to assist you through the troubleshooting process including:

- A Command Server display of ongoing execution information

- Command Server return codes and messages
 - Map audit log files, including data audit, execution audit, map settings, and card settings information
 - Map trace files
 - If an adapter is used, adapter-specific return codes and messages, audit log files, and trace files
- [Command Server display](#)**
- [Command Server return codes and messages](#)**
- [Map audit logs and trace files](#)**
-

Command Server display

Unless suppressed using execution commands, the Command Server displays several pieces of information when you run a map including the Command Server name, number of input objects found, number of output objects built, map execution time and a message indicating the map status. This information displays using a Command Server window on all Windows operating systems or using a console display on UNIX operating systems.

Ongoing feedback is shown during map execution and upon map completion. The messages that result (each message has a corresponding return code) indicate whether the map has completed and display the mapping results.

Command Server return codes and messages

After a map completes, a message indicates the execution results. These messages are also recorded in the appropriate audit logs.

Some return codes are only warnings indicating that the map did complete, but some problems occurred. You can change the default behavior for the warning codes by specifying **Fail on Warnings** or **Ignore Warnings** through the map settings or by using the **-F** or **-Z** execution command.

Map audit logs and trace files

Audit logs and trace files provide information that can be used effectively for debugging and troubleshooting purposes. There are many situations that occur during map execution in which you would want the activities associated with the processing of your data to be recorded in an audit log file or trace file. The auditing and tracing of the map execution process can be accomplished at various levels by specifying that the appropriate audit logs and trace files be enabled for maps, input cards, output cards, or particular adapters. These audit log and trace files may also be defined further to include summary or detail information about execution statistics, data information, or adapter activity.

For information about map audit logs and trace files, see the *Map Designer* documentation.

For information about audit log and trace files for an adapter, see the adapter-specific documentation about enabling and viewing these adapter-specific log files.

- [Map audit log](#)
 - [Map trace files](#)
-

Map audit log

When the **MapAudit Log Switch** setting is set to **ON**, you can select to have either execution information or data information recorded. You can also specify to have both execution and data information recorded for a complete picture of what transpired during the execution of a map. The map audit logs can be enabled from the Map Settings window in the Map Designer, Integration Flow Designer, Command Server for Windows, or on any platform using execution commands on the command line.

The default name for an audit log that is not unique is the full name of the map with the .log extension. The default name for a unique audit log is `Mer_mapname_processkey_mapcounter_hostname.log`. The default audit log directory is the directory of the compiled map file.

The **MapAudit** log can contain three different sections: **BurstAudit**, **SummaryAudit**, and **SettingsAudit**. The sections produced depend on the settings for **MapAudit**.

BurstAudit

The information in the **BurstAudit** section of the **MapAudit** setting is configurable using the Data Audit Settings tab in the Organizer. See the *Map Designer* documentation for information about configuring these options and interpreting the information in the **BurstAudit** section.

SummaryAudit

When the **SummaryAudit's Execution** setting is **Always**, **OnError**, or **OnWarningsOrError**, an **ExecutionSummary** section is included in the log file. The **ExecutionSummary** section of the audit log file provides a summary of the return codes, sources, targets, and work areas for the map. This execution information also includes:

- The command string to use if you were running the map from the command prompt.
- The number of input and output objects.
- Information about data and work files.
- Execution time

See the *Map Designer* documentation for more information.

An ExecutionLog per burst

When the **BurstAudit Execution** setting is **Always**, **OnBurstError**, or **OnBurstWarningOrError**, the log file will contain a section for each burst within the map. If all inputs have a **CardMode** of **Integral**, there will be a single **Burst** section. The following excerpt is from the **tryaudit.log** file, created from running the **tryaudit.mmc** compiled map, which is included in the *install_dir\examples\general\audit* folder and presents the **Burst** section:

```
<Burst count="1">

<DataLog>

<input card="1">
  <object index="1" level="1" size="10113" status="E00">
    DTX_MapAudit General Element
  </object>
  <object index="1" level="0" size="10113" status="V01">
    Audit_Log
  </object>
</input>

</DataLog>

<ExecutionLog burstreturn="28" ElapsedSec="3.9280">

<inputstatus card="1" bytes="10113" adapterreturn="0" contentreturn="28"/>
<inputstatus card="2" bytes="1067" adapterreturn="0" contentreturn="0"/>
<outputstatus card="1" bytes="610" adapterreturn="0" contentreturn="0"/>
<outputstatus card="2" bytes="1055" adapterreturn="0" contentreturn="0"/>
<outputstatus card="3" bytes="10938" adapterreturn="0" contentreturn="0"/>
<outputstatus card="4" bytes="0" adapterreturn="0" contentreturn="0"/>

</ExecutionLog>

</Burst>
```

The **ExecutionLog** section identifies the return code and elapsed time for the burst, as well as the status of each input or output, including both an adapter return code, if applicable, and a content return code.

ExecutionSummary per map

The **ExecutionSummary** section provides information at the map level. The following excerpt presents the **ExecutionSummary** section:

```
<ExecutionSummary MapStatus="Valid" mapreturn="0" ElapsedSec="3.9610"
  BurstRestartCount="0">
  <Message>Input type contains errors</Message>

  <CommandLine>'C:\install_dir\examples\general\audit\tryaudit.mmc'</CommandLine>
  <ObjectsFound>111</ObjectsFound>
  <ObjectsBuilt>10</ObjectsBuilt>

  <SourceReport card="1" adapter="File" bytes="10113" adapterreturn="0">
    <Message>Data read successfully</Message>
    <Settings>C:\install_dir\examples\general\audit\goodaud.log</Settings>
    <TimeStamp>07:03:10 December 11, 2003</TimeStamp>
  </SourceReport>

  <SourceReport card="2" adapter="File" bytes="1067" adapterreturn="0">
    <Message>Data read successfully</Message>
    <Settings>C:\install_dir\examples\general\audit\maperrcd.dat</Settings>
    <TimeStamp>06:11:46 May 29, 2003</TimeStamp>
  </SourceReport>

  <TargetReport card="1" adapter="File" bytes="610" adapterreturn="0">
    <Message>Data written successfully</Message>
    <Settings>C:\install_dir\examples\general\audit\result.txt</Settings>
    <TimeStamp>09:27:11 January 23, 2004</TimeStamp>
  </TargetReport>
  .
  .
  <WorkArea type="File">
    <inputarea card="1" Path="C:\install_dir\examples\general\audit\tryaudit.I01"
     TimeStamp="09:27:11 January 23, 2004" bytes="67522"/>
    <inputarea card="2" Path="C:\install_dir\examples\general\audit\tryaudit.I02"
     TimeStamp="09:27:11 January 23, 2004" bytes="67522"/>
    <outputarea card="1" Path="C:\install_dir\examples\general\audit\tryaudit.001"
     TimeStamp="09:27:11 January 23, 2004" bytes="65536"/>
    <outputarea card="2" Path="C:\install_dir\examples\general\audit\tryaudit.002"
     TimeStamp="09:27:11 January 23, 2004" bytes="65536"/>
    <outputarea card="3" Path="C:\install_dir\examples\general\audit\tryaudit.003"
     TimeStamp="09:27:11 January 23, 2004" bytes="65536"/>
    <outputarea card="4" Path="C:\install_dir\examples\general\audit\tryaudit.004"
     TimeStamp="09:27:11 January 23, 2004" bytes="65536"/>
  </WorkArea>
</ExecutionSummary>
```

As shown above, the execution log can provide some high-level debugging information, such as:

- **Map return code and message.** The map return code and message indicate how the mapping operation completed and whether there were any problems. For example, a map return code of 0 and message of **Map completed successfully** indicate that no execution errors were encountered. This information helps in

analyzing the source information in the log.

- **SourceReport and TargetReport.** For each source or target, the **ExecutionSummary** includes information indicating the adapter, the size of the data for the source or target, the adapter return code and message, and so on. For example, if using a database adapter, a return code of -17 and a message of Failed to parse SQL statement for an input would indicate that some error occurred when the adapter attempted to execute the appropriate SQL statement to fetch the data from the database.
- **WorkArea.** For each input or output for which a WorkArea is created, the **ExecutionSummary** includes information, such as the location and size.

The **ExecutionSummary** is a good place to start when diagnosing map execution problems because you can quickly determine the sources or targets in error. Using the information in the **ExecutionSummary**, you can produce more detailed troubleshooting information for only those sources or targets that experienced problems.

MapSettings section

The **MapSettings** section contains a list of all map settings, including the settings for **MapAudit**, **MapTrace**, **WorkSpace**, **Century (SlidingCentury)**, **Validation (CustomValidation)**, **Retry (MapRetry)**, and **Warnings**. The following excerpt presents the **MapSettings** section:

```
<MapSettings>

<MapAudit switch="ON">
    <Log executionsummary="Always" databurst="Always"
        database_sizevalidation="WrongSize" executionburst="Always"
        mapsettings="Always" datasettings="OnError"/>
    <Location type="File" action="Create">
        <Directory type="Map">C:\install_dir\examples\general\audit</Directory>
        <FileName type="Default" prefix="MapName">tryaudit.log</FileName>
    </Location>
</MapAudit>

<MapTrace switch="OFF"></MapTrace>

<WorkSpace>
    <Location type="File" action="Delete">
        <Directory type="Map">C:\install_dir\examples\general\audit</Directory>
        <FileName type="Default" prefix="MapName"></FileName>
    </Location>
    <Paging>
        <PageSize>64</PageSize>
        <PageCount>8</PageCount>
    </Paging>
</WorkSpace>

<SlidingCentury switch="OFF"></SlidingCentury>

<CustomValidation switch="ON">
    <OnValidationError>Continue</OnValidationError>
    <RestrictionError>Validate</RestrictionError>
    <SizeError>Validate</SizeError>
    <PresentationError>Validate</PresentationError>
</CustomValidation>

<Retry switch="OFF"></Retry>

<BurstRestart switch="OFF"></BurstRestart>

<Warnings type="Every" action="Warn"></Warnings>

</MapSettings>
```

This information can be useful during debugging to determine why execution occurred in a certain way.

DataSettings section

The DataSettings section contains a list of all **InputData** and **OutputData** settings depending on the adapter used, including the **FetchAs (CardMode)**, **WorkArea**, **Backup**, **Command (SourceAdapterCommand or TargetAdapterCommand)**, **OnSuccess**, **OnFailure**, **Retry (AdapterRetry)**, **Scope (AdapterScope)**, and so on. The following excerpt presents the DataSettings section:

```
<DataSettings>

<InputData card="1" CardMode="Integral" WorkArea="!Reuse">
    <Backup switch="ON" when="Always" action="Create">
        <Path>C:\install_dir\examples\general\audit\Mer_tryaudit_10441074864820_5.ll.bak
    </Path>
    </Backup>
    <Source adapter="File" OnSuccess="Keep" OnFailure="Rollback" Scope="Map"
        Warnings="Ignore">
        <Resource>C:\install_dir\examples\general\audit\goodaud.log</Resource>
        <Retry switch="OFF"></Retry>
    </Source>
</InputData>
.

.

<OutputData card="1">
    <Backup switch="OFF"></Backup>
    <Target adapter="File" OnSuccess="Create" OnFailure="Rollback" Scope="Map"
        Warnings="Ignore">
        <Resource>C:\install_dir\examples\general\audit\result.txt</Resource>
        <Retry switch="OFF"></Retry>
    </Target>
</OutputData>
```

```

</Target>
</OutputData>
.
.
</DataSettings>
```

This information can be useful during debugging to identify whether data should be copied to a backup file, whether the changes to a database target should be committed if the map fails, and so on.

Map trace files

Map trace files can be used for diagnosing invalid data or incorrect type definitions. When the MapTrace Switch setting is set to **ON**, you can specify the location and file name and select to have information captured for inputs and outputs including what level of detail you want recorded.

When you enable the **InputContentTrace** setting to trace your input data, the log file provides information about the data objects found, why data was found to be invalid, sizes and counts of data objects and their position in the data stream. When **OutputTrace** is enabled, the information recorded in the trace file reflects which output objects were built and which output objects evaluate to **NONE**.

You can choose to trace the input data, the output data or both. For additional information about map trace files, see the *Map Designer* documentation.

Note: You should enable the trace for debugging purposes only because enabling a map trace may affect execution performance.

External interfaces for applications, EXIT and RUN

Maps can interface with other maps, functions in libraries, or programs using the **EXIT** function, the **RUN** function or an adapter as a data source or target.

You can implement an adapter or **EXIT** function through a:

- library interface using a function in a library
- program passing a command line

You can invoke a user exit by specifying the **EXIT** function on a map rule, or by specifying an external library or application on a source or target card.

Platform-specific limitations can determine the availability and usage of these interface methods on a particular platform.

For example, the following table summarizes by platform the availability (Y) of the **RUN** function, the library interface, and program interface.

Table 1. External interfaces for user exits and RUN function

Platform	User Exit: External Library	User Exit: Application	RUN Function
HP-UX (PA-RISC)			Y
RS/6000 AIX	Y		Y
Windows			Y
z/OS Batch	Y	Y	Y
z/OS CICS		Y	Y

For information about the **RUN** and **EXIT** functions, see the *Functions and Expressions* documentation.

Windows platforms

There are several activities you can do with the Command Server that you have installed on your Windows environment.

You can:

- test your Windows Command Server installation by running the sample map provided
- use the Command Server to run your maps
- [Running the map examples](#)
After successfully installing the software, you can run the map examples included in the installation to test the installation to ensure the Command Server is functioning properly.
- [Using the Command Server](#)
The Command Server provides a window that shows ongoing information during map execution.
- [Help](#)

Running the map examples

After successfully installing the software, you can run the map examples included in the installation to test the installation to ensure the Command Server is functioning properly.

To run the map example

1. Run the setup script.
2. Execute the map:

- Go to `install_dir/examples/general/map/sinkmap`
- From the command line, type:

```
dtx sinkmap
```

The Command Server updates the screen while the map executes. If the map ran successfully, the following message is displayed:

```
Map completed successfully
```

If the map does not complete successfully, see [Using a Command Server](#) for troubleshooting information.

Using the Command Server

The Command Server provides a window that shows ongoing information during map execution.

To use the Command Server to run a map

When you open the Command Server application, a window opens indicating that the Command Server is ready to be run; however, no maps are currently loaded:

1. From the Map menu, select Run.
2. Choose a compiled map.

The Command Server window displays status information such as the number of objects found and built, as well as execution time.

Note: If the map does not complete successfully, see [Using a Command Server](#) for troubleshooting information.

To override map settings, right-click an executable map in Design Studio and edit the map settings. The map is ready to be run according to the newly entered override settings. The values entered in the Map Settings do not update the compiled map file, and are only recognized for this one-time execution. To permanently change any values for map settings, use the Map Designer to modify the settings and recompile the map.

Help

Help is available for the Command Server to provide access to descriptions and procedures including information about execution commands, return codes, functions, and so on. You can access Help by selecting Contents from the Help menu.

When using the command line, Help for Command Server is available by typing:

```
DTX -?
```

Note: Ensure your current directory is where the Command Server is located or specify the relative path or full path as applicable when entering the Help command.

z/OS Batch

There are several activities you can do with the Command Server that you have installed on your z/OS Batch environment.

You can:

- Test your z/OS Batch Command Server installation by running the sample map provided
- Use the Command Server to run your maps

- [Running the map examples](#)

After successfully installing the software, you can run the map examples included in the installation to test the installation to ensure the Command Server is functioning properly.

- [Using the Command Server](#)

- [z/OS Batch execution commands](#)

- [Troubleshooting](#)

There are three files that are essential to troubleshooting efforts.

- [VSAM considerations](#)

- [Using the EXIT function](#)

The EXIT function calls a user-written program to return data that is mapped to an item.

- [Using the RUN function](#)

The first argument to the RUN function is the name of the map to be executed.

- [Reusing work files](#)

- [Output file same as input file](#)

- [Appended output](#)

- [Resource names](#)

Running the map examples

After successfully installing the software, you can run the map examples included in the installation to test the installation to ensure the Command Server is functioning properly.

Using the Command Server

The Command Server for z/OS Batch requires JCL coding to execute as a batch job.

Note: Maps must always be built for z/OS Batch and transferred as BINARY for execution on the z/OS platform. **The file transfer must not use the CRLF option because map corruption may result.**

- [Execution command line \(JCL PARM statement\)](#)

- [Identifying data sources and targets](#)

On z/OS, data sources and targets that exist as datasets are identified by DDNAME. Sources and targets must be defined by DD statements in the execution JCL before a map can be executed.

- [Names to avoid](#)

- [Temporary datasets](#)

- [JCL](#)

The following sample JCL can be used as a guide to help develop JCL for your maps meeting your mapping requirements and the standards at your z/OS site.

- [Overriding Language Environment \(LE\) runtime options](#)

You can set the Language Environment (LE) runtime options in the executable DTXCMDSV program.

Execution command line (JCL PARM statement)

The execution command line is coded in the PARM statement of the JCL. The command line comprises the DDNAME of at least one map followed by one or more commands that affect the execution of that same map. The map DDNAME is required; execution commands are optional.

The first argument in a command line string is always the DDNAME that identifies the map to be executed. The map DDNAME may indicate a sequential dataset that contains the z/OS Batch-specific map (Example 1), a PDS member that contains the z/OS Batch-specific map (Example 2), or a member name (Example 3).

- [Example 1](#)

- [Example 2](#)

- [Example 3](#)

- [Example 4](#)

Example 1

```
//STEP1 EXEC PGM=DTXCMDSV,PARM='MYMAP -TS'  
//MYMAP DD DSN=USER.MAP1,DISP=SHR
```

Example 2

```
//STEP1 EXEC PGM=DTXCMDSV,PARM='MYMAPPDSDS(MAP1) -TS'  
//MYMAPPDSDS DD DSN=USER.MAPPDS,DISP=SHR
```

Example 3

```
//STEP1 EXEC PGM=DTXCMDSV,PARM='MYMAP -TS'  
//MYMAP DD DSN=USER.MAPPDS(MAP1),DISP=SHR
```

You can control the execution of a map by specifying one or more of the execution commands described below in [z/OS Batch Execution Commands](#). These commands must follow the map whose execution they affect.

You can execute multiple maps in one job step. Example 4 shows the syntax.

Example 4

```
//STEP1 EXEC PGM=DTXCMDSV,  
// PARM='MAP1DD -<opt1> -<opt2> MAP2DD -<opt1> -<opt2>'
```

The PARM statement may be continued onto multiple lines; however JCL syntax limits it to a total of 100 characters. However, this JCL limitation can be overcome by using the Command Dataset execution command (-@).

During execution, the map is loaded once and is removed from storage only when the task ends or when the map being executed is different than the last executed map. This means that when maps are executed repeatedly, they are not reloaded each time.

Identifying data sources and targets

On z/OS, data sources and targets that exist as datasets are identified by DDNAME. Sources and targets must be defined by DD statements in the execution JCL before a map can be executed.

When you create a map to be run using the Command Server for z/OS Batch, define source and target names associated with input and output cards. If a map is tested on a Windows platform before being executed on a z/OS platform, data source and target names typically use the Windows file naming convention of *drive:\path\filename.extension*. When a map executes on z/OS Batch, source and target names are converted to valid z/OS DDNAMES by removing *drive*, *path*, and *extension*, converting *_* characters to *#* characters, and translating the remaining characters to uppercase.

Consider the following example. A map is to be executed on z/OS Batch. It has one input and one output-both of them sequential datasets. The map was developed on a Windows platform; for testing purposes the data source was initially defined as C:\My\Data\In_file.txt and the data target was defined as F:\Outfile.dat. When the map executes on z/OS Batch, source and target names are converted to DDNAMES. The data source name becomes IN#FILE and the data target name becomes OUTFILE. The execution JCL must contain the following DD statements:

```
//IN#FILE DD DSN=MY.INPUT,...  
//OUTFILE DD DSN=MY.OUTFILE,...
```

The DDNAME associated with a data source or target may be changed at run time using a card override on the command line passed in the PARM statement. Input and output card override commands are discussed in [z/OS Batch Execution Commands](#).

The use of instream datasets for data sources is supported. If an instream dataset is used, ensure that no sequence numbers exist in columns 73 through 80; sequence numbers will be interpreted as data by the map. The **GETANDSET** function may not be used with instream datasets.

Note: If you are testing a map on a computer that is targeted to z/OS Batch, the process can be simplified by specifying data source and target names that, when converted, will resolve to the expected z/OS DDNAMES.

Names to avoid

When defining data sources and targets in the Design Studio, use names that resolve to unique DDNAMES upon conversion. You should also ensure that a converted name is not reserved. The following DDNAMES are reserved for use on z/OS:

DDNAME	Description
DTXDEBG	Aid in problem resolution
DTXAUD	Audit Log
DTXLOG	Execution log
DTXTRCE	Trace output
DBTRACE	DB adapter trace output
MQTRACE	WebSphere MQ adapter trace output
DTXMRN	Fixed DDNAME for the .mrn file.
SYSTMP <i>n</i>	where <i>n</i> is a number between 01 and 99. Reserved for statically allocated temporary datasets.
STEPLIB	Reserved by the operating system or Language Environment or both
SYSPRINT	Reserved by the operating system or Language Environment or both
SYSOUT	Reserved by the operating system or Language Environment or both
CEEDUMP	Reserved by the operating system or Language Environment or both

Temporary datasets

During mapping, temporary datasets are allocated as needed, depending on two factors:

- how the map is constructed
- the record format (RECFM) of the map's input and output files

One temporary dataset is allocated in all cases, which is a general use work file. One temporary dataset is also allocated for each input. Depending on how the map is constructed, it may create a temporary dataset for each output to serve as its work file.

In addition, one temporary dataset is always allocated for each input or output defined with an RECFM that is not F (fixed unblocked) or FBS (fixed blocked standard).

Because you can develop a map with no inputs, and output work files may not be used in some situations, the minimum number of possible temporary files for a map execution is 1. The maximum possible number is $1 + (2 * \text{number of inputs}) + (2 * \text{number of outputs})$. Temporary files will be allocated using this formula where you have specified in the map to create output work files during its execution and all the inputs and outputs are formatted with a RECFM other than F or FBS.

By default, temporary datasets are dynamically allocated on disk. Dynamically allocated datasets require no DD statements in the JCL and can store up to approximately 3.1 MB of data. This is based on the settings specified in the **ALLOCxx** PARMLIB member. These datasets are deleted at the end of mapping and have system-generated dataset names with low-level qualifiers of **CTMFnnn**, where *nnn* is a number indicating the sequence in which the dataset was created. For example, the first temporary dataset created during a mapping has a qualifier of **CTMF000**; the second, **CTMF001**, and so on.

If the space allocation for a dynamic temporary dataset is insufficient, override it with a static allocation in the JCL. Use the following DD statement as a template:

```
//SYSTMP<nn> DD DSN=&&TEMP<nn>,
//                      DISP=(NEW,DELETE,DELETE),
//                      DCB=(RECFM=FBS,LRECL=<lrecl>),
//                      UNIT=<unit>,
//                      SPACE=(<space>)
```

The DDNAME for a statically allocated temporary dataset must use the form, **SYSTMPnn**, where *nn* is a number between 01 and 99 corresponding to the sequence number of the temporary dataset. For example, if the third temporary dataset created by a map (the system-generated dataset name with a low-level qualifier of **CTMF002**) runs out of space (this will be indicated by an **IEC030I x37** message in JES log), override the dynamic allocation with a **SYSTMP03** DD statement that provides sufficient space.

The dataset name for a static allocation must begin with **&&** which indicates a temporary dataset. The disposition must be (NEW,DELETE,DELETE). The RECFM should be FBS (fixed blocked standard) although F (fixed unblocked) can be used. You can choose the logical record length, although larger record lengths may increase IO efficiency. Do not use the BUFNO parameter.

To statically allocate all the temporary datasets that can possibly be used by a map, code DD statements **SYSTMP01** through **SYSTMPnn**, where *nn* is equal to $1 + (2 * \text{number of inputs}) + (2 * \text{number of outputs})$. This will provide adequate space.

Note: Statically allocating temporary datasets may increase map performance.

Static temporary dataset allocations are reused when multiple maps are executed from the command line. Consider the following scenario: the command line executes MAPA followed by MAPB and MAPC. MAPA uses seven temporary files, MAPB nine, and MAPC four. In this example, you must statically allocate not 20 temporary datasets ($7 + 9 + 4$), but nine temporary datasets, the maximum number needed by any map in the series. Each successive map execution reuses the existing static temporary file allocations beginning with SYSTMP01.

The resulting execution log reports complete temporary dataset usage information including DDNAME, dataset name, DCB, and size.

JCL

The following sample JCL can be used as a guide to help develop JCL for your maps meeting your mapping requirements and the standards at your z/OS site.

The z/OS Batch Command Server is called DTXCMDSV.

```
/** The program to be executed is DTXCMDSV. The command line is passed in the
/** PARM statement. The map to be executed is MYMAP, which has one input,
/** MYINPUT and one output, MYOUTPUT. The map and all its datasets are
/** accessed through ddnames.
*/
//STEP1 EXEC PGM=DTXCMDSV,PARM='MAPPDS (MYMAP) /P256:12 /TS'
/*
/** The ddname MAPPDS identifies the partitioned dataset
/** where IBM Transformation Extender maps are stored. MYMAP is the member.
*/
//MAPPDS DD DSN=MY.MAPPDS,DISP=SHR
//DTXLOG DD SYSOUT=* Execution log
//DTXAUD DD SYSOUT=* Audit
//DTXTRACE DD SYSOUT=* Map Trace
//DTXDEBUG DD SYSOUT=* Map Debug listing
//DTXMRN DD DSN=USER.TEST(RES1MRN),DISP=SHR
//SYSPRINT DD SYSOUT=* Used by LE
//SYSOUT DD SYSOUT=* Used by LE
//CEEDUMP DD SYSOUT=* Used by LE
/*
/** STEPLIB identifies the load library where IBM Transformation Extender was
/** installed.
/** It identifies the Language Environment library if it is not in the link list.
*/
//STEPLIB DD DSN=MY.LOADLIB,DISP=SHR
//          DD DSN=CEE.SCEERUN,DISP=SHR
/*
/** The map's output.
*/
//MYOUTPUT DD DSN=MY.OUTPUT,
//                      DCB=(RECFM=VB,LRECL=256),
//                      UNIT=SYSDA,SPACE=(TRK,(5,1)),
//                      DISP=(NEW,CATLG,DELETE)
/*
/** The map's input.
*/
//MYINPUT DD DSN=MY.INPUT,DISP=SHR
/*
/** Static temporary dataset allocations.
*/
//SYSTMP01 DD DSN=&&TEMP01,DISP=(NEW,DELETE,DELETE),
//                      DCB=(RECFM=FBS,LRECL=23476),
//                      UNIT=VIO,SPACE=(TRK,(5,1))
//SYSTMP02 DD DSN=&&TEMP02,DISP=(NEW,DELETE,DELETE),
//                      DCB=(RECFM=FBS,LRECL=23476),
//                      UNIT=VIO,SPACE=(TRK,(5,1))
//SYSTMP03 DD DSN=&&TEMP03,DISP=(NEW,DELETE,DELETE),
```

```

//          DCB=(RECFM=FB5,LRECL=23476),
//          UNIT=VIO,SPACE=(TRK,(5,1))
//SYSTMP04 DD DSN=&#39;TEMP04',DISP=(NEW,DELETE,DELETE),
//          DCB=(RECFM=FB5,LRECL=23476),
//          UNIT=VIO,SPACE=(TRK,(5,1))
//SYSTMP05 DD DSN=&#39;TEMP05',DISP=(NEW,DELETE,DELETE),
//          DCB=(RECFM=FB5,LRECL=23476),
//          UNIT=VIO,SPACE=(TRK,(5,1))
//
```

Overriding Language Environment (LE) runtime options

You can set the Language Environment (LE) runtime options in the executable DTXCMDSV program.

Use the **PARM** field or the **CEEOPTS DD** statement in your JCL to set the LE options.

For information about how to use the LE runtime options, see the following publications:

- SA22-7561-0X z/OS Language Environment Programming Guide
- SA22-7562-0X z/OS Language Environment Programming Reference

The LE options that you specify on either the **PARM** field or the **CEEOPTS DD** override both the system and program-defined options. A common use of these LE runtime options with IBM Transformation Extender is to override the **STACK** and **HEAP** runtime options.

As an example, if you specify the following LE options on the **CEEOPTS DD** statement of your JCL, it produces both an LE Options and Storage report. These reports can assist you in tuning the product's interaction with LE. The reports are directed to the **SYSOUT DD** statement.

```

//CEEOPTS DD *
  RPTOPT(ON)
  RPTSTG(ON)
/*

```

To use the DTXCMDSV executable program, specify it in your JCL as shown in the following example:

```

//DTX EXEC PGM=DTXCMDSV,REGION=0M,
//      PARM='REVERSEB /VX0D,X0A REVERSEI /VX0D,X0A REVERSEO'

```

z/OS Batch execution commands

To properly enter the z/OS Batch execution commands, you must observe the command syntax as well as the following general rules:

- Each command must begin with a hyphen (-) or a forward slash (/).
- At least one space between commands is required (for example: -A -v).
- Do not put spaces within commands, except for -I, -O and -V, which require a space before the source or target name.
- Execution commands are not case sensitive.
- When specifying commands, the entire command string (the map name and its associated commands and options) may be on one line or several lines if the -@ command is used.

Only the execution commands listed in the following table are supported under z/OS Batch. The majority of these commands are described in the *Execution Commands* documentation. If there is a difference in how the command is used for z/OS Batch, the exception is noted in the Syntax column. Commands only used by z/OS Batch are marked "Unique to z/OS Batch" in the Syntax column. z/OS Batch-specific commands are not listed in the *Execution Commands* documentation, so are described here.

The Syntax displayed is for z/OS Batch usage.

Command	Syntax
MapAudit	-A[D[R W]] [E[R W]] [B[R W]] [C[R W]] [P[R W]] [[M][S]]
Sliding Century	-D [ccyy 0]
	Exception: None
Fail on Warnings	-F[!][warning_code[:warning_code...]]
	Exception: None
Ignore Validation Options	-G[R] [S] [B]
	Exception: None
Input Dataset Override (-I)	-I[F]card_num[W !W][B] ddbname
	Unique to z/OS Batch
Input Echo Override	-IEcard_num[Ssize]
	Exception: For z/OS Batch, this command cannot be used to designate the source (source) or change the specification for using workfiles (W !W)
List	-L
	Exception: None

[No DTXLOG \(-NL\) or \(/NL\)](#)

-NL|/NL
Unique to z/OS Batch

[Output Dataset Override \(-O\)](#)

-O[F] *card_num* [+|!+] [B] *ddname*
Unique to z/OS Batch

Output Echo Override

-OE *card_num* [S]
Exception: For z/OS Batch, the X option is not available to exclude this card from echo.

WorkSpace Paging

-P*size:count*
Exception: None

MapTrace

-T[I|C*card_num*|I_{from}[:*to*]||O|OC*card_num*][S]
Exception: For z/OS Batch, the =dir option is not available to change from the default the location in which the trace file is created.

Stop Validation

-V
Exception: None

[Workfiles in Memory \(-WM\)](#)

-WM
Exception: None

Ignore Warnings

-Z[!][warning_code[:warning_code...]]
Exception: None

[Print Uppercase \(-+\)](#)

-+
Unique to z/OS Batch

[Command Dataset \(-@\)](#)

-@*ddname*
Unique to z/OS Batch

[Problem Resolution \(-\\$\)](#)

-\$
Unique to z/OS Batch

[Record Separators \(/V\)](#)

/V[separator_list...] *ddname*
Unique to z/OS Batch

The following commands are unique to z/OS Batch.

- [Input Dataset Override \(-I\)](#)
- [No DTXLOG \(-NL\) or \(/NL\)](#)
- [Output Dataset Override \(-O\)](#)
- [Workfiles in Memory \(-WM\)](#)
- [Print Uppercase \(-+\)](#)
- [Command Dataset \(-@\)](#)
- [Problem Resolution \(-\\$\)](#)

Use the Problem Resolution execution command (-\$) only under the direction of Customer Support.

- [Record Separators \(/V\)](#)

Input Dataset Override (-I)

Use the Input Dataset Override execution command (-I) to override the specified input card source with a dataset only for this execution.

-I [F] *card_num*[W|!W] [B] *ddname*

Option

Description

F

Optional for compatibility with previous versions.

card_num

The number of the input card to override.

W

Reuse the existing work file for this data source. If the work file does not exist at execution time, create it.

!W

Do not use the existing work file, but create a new work file each time the map is run.

B

If map execution is **not** successful, roll back any changes made to the dataset and reinstate the dataset to the original state prior to map execution.

ddname

Specify the DDNAME of the dataset used as the input source.

No DTXLOG (-NL) or (/NL)

Use the No DTXLOG execution command (-NL) to suppress the creation of a **DTXLOG**. When you use it, no informational data will be produced from **DD DTXLOG**. If you pass the -NL command on the command line through the **RunMap()** function, **DD RUNLOG01 (RUNLOG02 and so forth)** information will be suppressed.

-NL | /NL

Output Dataset Override (-O)

Use the Output Dataset Override execution command (-O) to override the specified output card target with a dataset for this execution only.

-O [F] *card_num*[+|!+] [B] *ddname*

Option

Option	Description
F	Optional for compatibility with previous versions
<i>card_num</i>	The number of the output card to override
+	Append the data for the current map execution to an existing dataset.
!+	Do not append to an existing dataset, but create a dataset each time the map is run.
B	If map execution is not successful, rollback any changes made to the dataset and reinstate the dataset to the original state prior to map execution.
<i>ddname</i>	Specify the DDNAME of the dataset used as the output target.

Workfiles in Memory (-WM)

Use the Workfiles in Memory execution command (-WM) to override the default allocation of work files on disk and to allocate them as memory files.

-WM

Print Uppercase (+)

Use the Print Uppercase execution command (+) to cause all printed output, execution logs, audit logs, and trace logs to be written in uppercase.

-+

Command Dataset (-@)

Use the Command Dataset execution command (-@) to specify a DDNAME defining a dataset that contains map names and execution commands. This command is unique to z/OS Batch. If this command is used, it can only appear in the PARM statement and it must be the only command coded in the parm statement (for example: EXEC PGM=DTXCMDSV, PARM='-@DTXPARM'). Using the command dataset eliminates the 100-byte limitation of the JCL PARM statement. Within the command dataset, map names and execution commands may be coded free-form. The command dataset may reside on disk or it may be defined instream in the JCL. If defined instream, ensure that there are no sequence numbers in columns 73 through 80. A sequence number, because it does not begin with the switch indicator (- or /), is interpreted by the command parsing logic as a map name.

-@ *ddname*

Option	Description
<i>ddname</i>	The DDNAME of the command file dataset.

Problem Resolution (-\$)

Use the Problem Resolution execution command (-\$) only under the direction of Customer Support.

This command produces output that is unintelligible to the user but is required by Customer Support for debugging.

Ensure you use the hexadecimal value **0x5B** for the '\$' character on US EBCDIC systems. Regardless of the character displayed in the JCL, using Hex **0x5B** is essential to make this option work and produce the debug output if Support requests it.

Record Separators (/V)

Use the Record Separators execution command (/V) to enable special handling of datasets with variable length records (datasets defined with RECFM=V).

/V[separator_list...] ddname

Option	Description
separator_list	Specify a comma-delimited string of separator bytes that can be expressed in hex format (for example: /VX5B,X6C,X7C) or character format (for example: /V\$,%,@). The default record separator if none is indicated on the /V command as a single byte, hex15.
ddname	The DDNAME of the input or output dataset

This command functions differently for data sources and targets. If specified for an input dataset, the indicated (or default) record separator string is appended to each variable record after it has been read. If specified for an output dataset, the record separator string is interpreted as marking the end of a variable length record in the output.

For inputs, this command causes a string of one or more record separator bytes to be appended to each variable record to ensure that the data maps properly on z/OS Batch. For example, suppose you are mapping an input file on Windows that consists of lines of text delimited by carriage-return/line-feed (0D 0A) pairs. Even though you cannot see them in a text editor, these delimiters are embedded in the data and the map expects them to be there.

Now you need to run this same map on z/OS Batch. However, when you upload the input file to a z/OS dataset of variable length records using the CRLF option, the record separators embedded in the data are removed during the transfer. Consequently, when the file is read in for mapping under z/OS Batch, the record separators must be reinstated or the data will not map correctly. The solution is to use the command /VX0D, X0A to append a hex 0D 0A record separator string to the end of each variable record.

If a dataset of variable length records already contains the separators defined to the map embedded in the data, this option should not be used.

For outputs, this command specifies that the indicated (or default) record separator byte or bytes are used to mark the ends of variable length records in the output. When this command is used, the map must embed record separator strings in the output data. These separator strings indicate the places where variable length record boundaries are to occur in the final output. The record separator strings are removed when the variable records are created. They do not appear in the output in its final form.

Output variable records wrap at the maximum record length defined for the dataset. If an output dataset is defined in the JCL with DCB=(RECFM=VB, LRECL=80) and either by accident or design a map produces an output record which is actually 120 bytes in length, two variable records are produced in the output dataset—one that is 80 bytes in length and one that is 40 bytes in length.

Troubleshooting

There are three files that are essential to troubleshooting efforts.

These files have DDNAMES as indicated in the following table:

DDNAME	File	Description
DTXTRCE	Trace File	Produced as a result of specifying the Trace execution command (-T)
DTXAUD	Map Audit Log	Produced as a result of enabling the Map Audit settings for a map and running the map with the Audit execution command (-A)
DTXDEBG	Problem Resolution	Produced as a result of specifying the Problem Resolution execution command (-S) as instructed by Customer Support.

- EBCDIC-encoded XML trace files**

The IBM Transformation Extender batch command server can generate trace log files in EBCDIC encoding. You can display EBCDIC-encoded trace files by using the vi editor or the ISPF application of the IBM® Time Sharing Option/Extensions (TSO) facility.

EBCDIC-encoded XML trace files

The IBM Transformation Extender batch command server can generate trace log files in EBCDIC encoding. You can display EBCDIC-encoded trace files by using the vi editor or the ISPF application of the IBM® Time Sharing Option/Extensions (TSO) facility.

To configure EBCDIC-encoded trace log files, add the following DD statement to the JCL step that invokes IBM Transformation Extender:

```
//CEEOPDS DD *  
ENVAR ("DTX_XML_LOG_ENCODING_IBM1047=TRUE")  
/*
```

VSAM considerations

The z/OS Batch Command Server supports the use of VSAM keyed-sequenced (KSDS), entry-sequenced (ESDS) and relative-record (RRDS) datasets for both inputs and outputs subject to the following conditions:

- VSAM datasets must be pre-defined and, if used for input, pre-loaded using IDCAMS or some other utility.
- The z/OS Batch Command Server opens input and output VSAM files for both reading and writing.
- When used as input, all VSAM datasets are processed sequentially; as output, KSDS files are updated and ESDS and RRDS files are appended.
- The use of SHAREOPTIONS 3 or 4 is not recommended. The library IO routines supplied by Language Environment do not check SHAREOPTIONS at dataset open time and do not provide support for read and write integrity for shared VSAM files.

Using the EXIT function

The **EXIT** function calls a user-written program to return data that is mapped to an item.

The **EXIT** function is documented in your *Functions and Expressions* documentation. However there are special considerations when using this function with the z/OS Batch Command Server.

A user exit is invoked by calling the **EXIT** function in a mapping rule. The **EXIT** function expects three arguments:

```
EXIT ("DLL_load_module_or_non_null",
      "DLL_function_or_load_module", "parameters")
```

If the user exit program is a DLL, the first argument must contain the name of the DLL load module and the second argument must contain the name of the function within the DLL.

If the user exit program is not a DLL, the first argument must contain some value that is not a null string ("")-for example, one or more spaces-and the second argument must contain the name of the load module.

The third argument is a text item that contains parameter information passed to the exit program.

Your user exit programs must reside in one of the load libraries referenced by the STEPLIB DDNAME in the JCL used to run the z/OS Batch Command Server. Any static data areas declared within an exit program, such as COBOL working-storage or C static variables, persist across multiple calls to the exit.

For non-C language (for example, COBOL) exits, four arguments are passed that contain the address of:

- A full word that contains the length of the parameter text item
- A space-padded 64K buffer that contains the parameter text item
- A full word that will contain the length of the resulting text item
- A 64K buffer where the resulting text item will be placed

For C language exits, a single argument is passed, which is the address of the EXITPARAM structure as described in the *Functions and Expressions* documentation.

User exits must support the conventions of z/OS Batch standard linkage and be link-edited in 31-bit addressing mode. DLL exits must be compiled as DLL code.

For an example of a C language and COBOL user exit program, see DTXEXRME in the DTX.SDTXSAMP PDS included in the IBM Transformation Extender with Command Server installation.

Using the RUN function

The first argument to the **RUN** function is the name of the map to be executed.

On z/OS, this must be the DDNAME that identifies the deployed map. To ensure this, the z/OS Batch Command Server converts the map name to a legal z/OS DDNAME by removing drive, path, and file extension information if it exists, converting _ characters to # characters and translating the remaining characters to uppercase. For example, the map rule **RUN ("install_dir \mymaps\map_10.mmc")** executes the map identified by the DDNAME **MAP#10**.

When using the **RUN** function, the z/OS Batch Command Server execution log, audit log, trace and problem resolution output must be written to individual datasets for each level of recursion; otherwise the output from different recursion levels are mixed together and will be unintelligible. By default, output from a map executed with the **RUN** function is directed to dynamically-allocated print datasets assigned to the default SYSOUT class for the job. This may be overridden by coding additional DD statements in the JCL. For each level of recursion, four DD statements should be added to your execution JCL:

- **RUNLOG nn** z/OS Batch execution log file
- **RUNAUDnn** Data audit log file
- **RUNTRCnn** Trace file
- **RUNDBGnn** z/OS Batch problem resolution file (used with Customer Support)

where *nn* is the level of recursion (always two positions, zero-filled).

For example, if an executed map issues a **RUN** function and the map to be run does not execute any other maps, one level of recursion exists and the following statements should be added to the JCL step:

```
//RUNLOG01 DD SYSOUT=B  
//RUNAUD01 DD SYSOUT=B  
//RUNTRC01 DD SYSOUT=B  
//RUNDBG01 DD SYSOUT=B
```

A map executed using the **RUN** function is loaded once and is removed from storage when the task ends or when the map executed is different from the last map executed. Maps repeatedly executed are not reloaded each time.

Statically allocated temporary datasets are reused when a map is executed repeatedly using the **RUN** function. For example, consider the following scenario: MAPA calls MAPB 100 times in succession using the **RUN** function. MAPA uses seven temporary files and MAPB uses three. In this example it is necessary to statically allocate, not 307 temporary datasets ($7 + (100 * 3)$), but 10 temporary datasets. MAPA will use SYSTMP01 through SYSTMP07 and MAPB will reuse SYSTMP08 through SYSTMP10 each time it is executed.

Reusing work files

To reuse a work file in the z/OS Batch Command Server, you must code a DD statement in the JCL, allocating the work file as a cataloged dataset. Only input work files can be reused.

The DDNAME for the work file to be reused must be a concatenation of up to five characters of the map name, followed by the `IW` string, followed by the number of the work file. The number of the work file corresponds to the number of the input card. For example, if you are executing a map called **MYMAP** and you want to reuse the work file for input card number three, code a DD statement with the following DDNAME:**MYMAPI3**.

Work file DCB parameters should define a file with fixed-blocked standard (FBS) records. The disposition should be (MOD,CATLG,DELETE). Using MOD will ensure that if the file does not exist, it will be created. Using CATLG may cause the message NOT RECATLGD 2 to appear in the output JES log, but this message can be safely ignored. The BUFNO parameter should not be used.

The following example DD statement demonstrates how you can reuse the work file for input card number 1 of a map named **MYMAP**:

```
//MYMAPI1 DD DSN=name,  
//          DISP=(MOD,CATLG,DELETE),  
//          DCB=(LRECL=23476,RECFM=FBS),  
//          UNIT=unit,  
//          SPACE=(space)
```

Output file same as input file

There are no special z/OS Batch considerations when the file named in the output card a map is the same as the file named in its input card. Code one DD statement for both files with a disposition of `OLD` or `SHR`. If you are mapping your output back to the input, be careful using card overrides.

The following example will not work:

```
//STEP1 EXEC PGM=DTXCMDSV,PARM='MAP -IF1 INFILE -OF1 OUTFILE'  
//INFILE DD DSN=MY.IOFILE,DISP=SHR  
//OUTFILE DD DSN=MY.IOFILE,DISP=SHR
```

Instead, the JCL should read:

```
//STEP1 EXEC PGM=DTXCMDSV,PARM='MAP -IF1 IOFILE -OF1 IOFILE'  
//IOFILE DD DSN=MY.IOFILE,DISP=SHR
```

Appended output

When an output card for a map specifies that the output be appended to a file, the target file MUST previously exist. Use a disposition of `OLD` or `SHR`.

Resource names

For information about creating resource names for use on the Command Server for the z/OS Batch platform, see [Resource Names for z/OS Platform](#).

z/OS CICS

There are several activities you can do with the CICS Execution Option that you have installed on your z/OS CICS environment.

You can:

- Test your CICS Execution Option installation by running the sample maps provided in the installation.
- Use the CICS Execution Option to run your maps.

Note: The IBM Transformation Extender includes the CICS Execution Option through which you can run maps in your CICS environments. Prior to IBM Transformation Extender version 8.1, this functionality was provided in the CICS Command Server product.

- [Running the map examples](#)

After successfully installing the software, you can run the map examples included in the installation to test the installation to ensure the CICS Execution Option is functioning properly.

- [Using the CICS Execution Option](#)

The CICS Execution Option can be invoked in several ways.

- [z/OS CICS execution commands](#)

- [Specifying storage medium](#)

The z/OS CICS Execution Option can map from or to any of six different storage media.

- [Using EXEC CICS LINK passing storage buffers](#)

When the z/OS CICS Execution Option is called by an EXEC CICS LINK, it is often because you want to pass to the CICS Execution Option the address of one or more storage buffers containing input data or the address of one or more storage buffers that will contain mapped output data or both address types.

- [Using the EXIT function](#)

The EXIT function is available when entering mapping rules for fields using the Map Designer on your PC.

- [Using the RUN function](#)

- [Preloading IBM Transformation Extender maps and .mdq files in a z/OS CICS environment](#)

The CICS® preloader utility preloads IBM Transformation Extender maps and database query (.mdq) files into memory. When maps and .mdq files run frequently,

you can reduce processing by preloading them into memory. Preloaded files do not require I/O processing.

- **Abnormal termination**

If a task terminates abnormally with an internal error, the return code 16 and an error message are passed back to the calling program.

- **The DTXMPULD utility**

In addition to loading maps into the KSDS, where maps are stored for use in the z/OS CICS region as described above, the DTXMPULD utility has the capability of listing and deleting maps from the **KSDS** map.

- **Resource names**

- **Multiple IBM Transformation Extender versions**

This section presents information about running multiple versions of the IBM Transformation Extender CICS Execution Option concurrently in a single CICS region.

Running the map examples

After successfully installing the software, you can run the map examples included in the installation to test the installation to ensure the CICS Execution Option is functioning properly.

For information about the examples, see the example readme files in the DTX.SDTXSAMP PDS, as well as the IBM Transformation Extender for z/OS [release notes](#).

Using the CICS Execution Option

The CICS Execution Option can be invoked in several ways.

It can be invoked:

- From a clear terminal, enter the **DTXI** transaction and follow it with a command string containing the name of at least one map and any execution commands required for the execution of that map. Command arguments must be separated by spaces. For example, to run the map **MYMAP** from a terminal with commands specifying input and output card overrides, type:

```
DTXI MYMAP -I1 MYINPUT -O1 MYOUTPUT
```

Note: The **MYMAP** file must have been built for z/OS CICS, file transferred to z/OS, and loaded into the z/OS CICS map **KSDS**.

- Start the **DTXI** transaction from a user program by passing the transaction name **DTXI** followed by the command string containing the map name and any required execution commands in the **FROM** option of the **EXEC CICS START** command.
- Automatically initiate the **DTXI** transaction from a trigger transient data queue. The first record in the queue must contain the transaction name **DTXI** followed by the command string.
- Invoke the **DTXI** program using an **EXEC CICS LINK** and pass the link control blocks. (Use the **DTXSCTST** sample program as a guide.) These control blocks define areas for the command string, optional data buffers, and a return code and message.

Note: When **DTXI** is invoked using an **EXEC CICS LINK**, the command string supplied must be null terminated. If the calling program is COBOL, this means that the command string must be followed, in working storage, by at least one byte containing LOW-VALUES.

To run a map using the CICS Execution Option for z/OS CICS

1. Build the map for z/OS CICS. From the Map Designer select Map > Build for Specific Platform and select IBM(TM) z/OS from the drop-down list.
2. File transfer the platform-specific map to z/OS CICS as BINARY with no ASCII to EBCDIC translation and no elimination of carriage returns (CRs) and line feeds (LFs).
Note: Transfer the file exactly as it is, without any changes.
3. Modify and execute the **DTXCMJCL** JCL to load the map into the z/OS CICS VSAM map dataset see [Running the Map Examples](#).
Note: It is possible to execute multiple maps from one command line. The syntax is:

```
map1 map1_execution_commands map2 map2_execution_commands
```

... Maps are loaded once and removed from storage when the task ends or when the map executed is different from the last map executed. This means that maps repeatedly executed are not reloaded each time.

z/OS CICS execution commands

To properly enter the z/OS CICS execution commands, you must observe the command syntax as well as the following general rules:

- Each command must begin with a hyphen (-) or a forward slash (/).
- At least one space between commands is required (for example: **-A -V**).
- Do not put spaces within commands, except for **-I**, **-O** and **-V**, which require a space before the source or target name.
- Execution commands are not case sensitive.

Only the execution commands listed in the following table are supported under z/OS CICS. The majority of these commands are described in the *Execution Commands* documentation. If there is a difference in how the command is used for z/OS CICS, the exception is noted in the Syntax column. Commands only used by z/OS CICS are marked "Unique to z/OS CICS" in the Syntax column. z/OS CICS-specific commands are not listed in the *Execution Commands* documentation, so are described in this section. The Syntax displayed is for z/OS CICS usage.

Command	Syntax
<u>Audit (-A)</u>	<pre>-A [D R W]][E R W]][B R W]][C R W]][P R W]][M S][@storage_codeName]</pre>
SlidingCentury	<pre>-D [ccyy 0]</pre>
	Exception: None.

Fail on Warnings

-F[!][warning_code[:warning_code...]]

Exception: None.

Ignore Validation Options

-G[R|S|P]

Exception: None.

[Input Dataset Override \(-I\)](#)

-I[F]card_num[W|!W][B][[@storage_code]

[{:W|P[delimiter_list]}]] [name]

Unique to z/OS CICS.

Input Echo Override

-IEcard_num[Ssize]

Exception: For z/OS CICS, this command cannot be used to designate the source (source) or change the specification for using workfiles (W|!W).

[Execution Log \(-L\)](#)

-L[@storage_codeName]

Unique to z/OS CICS.

[Output Dataset Override \(-O\)](#)

-O[F]card_num[+|!+][B][[@storage_code]

[#record_length][{:W|P[delimiter_list]}]] [name]

Unique to z/OS CICS.

Output Echo Override

-OEcard_num[S]

Exception: For z/OS CICS, the X option is not available to exclude this card from echo.

WorkSpace Paging

-Psize:count

Exception: None.

[Enqueuing on Data Files \(-Q\)](#)

-Q

Unique to z/OS CICS.

-T[I|ICcard_num[Ifrom[:to]]][O|OCcard_num][S]

[@storage_codeName]

Unique to z/OS CICS.

Stop Validation

-V

Exception: None.

[Workfiles in Memory \(-WM\)](#)

-WM

Exception: None.

Ignore Warnings

-Z[!][warning_code[:warning_code...]]

Exception: None.

[Problem Resolution \(-\\$\)](#)

-\$

Unique to z/OS CICS.

[Print Uppercase \(+\)](#)

+*

Unique to z/OS CICS.

[Eliminate AICA \(ICVR Timeout\) Abends \(-In\)](#)

-In

Unique to z/OS CICS.

The following commands are unique to z/OS CICS.

- [Audit \(-A\)](#)
- [Input Dataset Override \(-I\)](#)
- [Execution Log \(-L\)](#)
- [Output Dataset Override \(-O\)](#)
- [Enqueuing on Data Files \(-Q\)](#)
- [Trace \(-T\)](#)
- [Workfiles in Memory \(-WM\)](#)
- [Problem Resolution \(-\\$\)](#)

Use the Problem Resolution execution command (-\$) only under the direction of Customer Support.

- [Print Uppercase \(+\)](#)
- [Eliminate AICA \(ICVR Timeout\) Abends \(-In\)](#)

Audit (-A)

Use the Audit execution command (-A) to control the creation of audit log information. When used with other specified options, the **Audit** settings specified with the Audit command (-A) override all **Audit** settings compiled into the map. When used without other specified options, no audit log is produced. For information about the audit log files, see the *Map Designer* documentation. The default is to write the audit information to a temporary storage queue named **DTXAUD**. You can override the default by specifying the storage medium to be used and the name. See [Specifying Storage Medium](#) for details.

```
-A [D|R|W] [E|R|W] [B|R|W] [C|R|W] [P|R|W] [[M] [S]]  
[@storage_codeName]
```

Options

Description

@storage_code

Specify one of the following codes, which represents the storage medium to be used for the audit report
T = Specify a temporary storage queue
D = Specify a transient data queue
R = Specify VSAM relative-record dataset
K = Specify VSAM key-sequenced dataset
E = Specify VSAM entry-sequenced dataset

Name

Specify the queue name or VSAM dataset name for the audit report

Name must not be preceded by any spaces.

Input Dataset Override (-I)

Use the Input Dataset Override execution command (-I) to override the specified input card source with a dataset for this execution only. Use this command to override data source specifications that are in the compiled map file for a single execution of a map.

Use this command to override the name for a specific input data source, work file behavior, or rollback.

You can also specify storage medium, record separator delimiters, behavior for data with variable length records, and so on. See [Specifying Storage Medium](#) and [Variable Length Records](#) for more information about specifying these options.

```
-I[F]card_num[W|!W][B][[@storage_code][:{W|P[delimiter_list]}]] [name]
```

Option

Description

F

Optional for compatibility with previous versions

card_num

The number of the input card to override

W

Reuse the existing work file for this data source. If the work file does not exist at execution time, a work file is created.

!W

Do not use the existing work file but create a new work file each time the map is run.

B

If map execution is **not** successful, roll back any changes made to the file and reinstate the file to the original state prior to map execution.

storage_code

Specify one of the following codes, which represent the type of storage medium to be used for input data. The default is a temporary storage queue with the name defined for the input card with the path and file extension removed.

T = Specify a temporary storage queue.

D = Specify a transient data queue.

R = Specify VSAM relative-record dataset.

K = Specify VSAM key-sequenced dataset.

E = Specify VSAM entry-sequenced dataset.

B = Specify storage buffer.

Note: If the storage buffer option (B) is selected, then specifying an override name is not applicable.

:W

Specify for a file consisting of variable length records to copy data from the input file into fixed format workspace before mapping begins. The map will take its input not from the file but from workspace. For details see [The :W Option](#).

:P

Specify that the data will be *profiled*, that is, a table will be built in memory which cross-references record identifiers to corresponding file offsets. For details, see [The :P Option](#).

Delimiter_list

Specify a comma-delimited list of character or hex values. (Xnn is the syntax for indicating a hexadecimal value.) The default delimiter is a single byte with the value hex 15, decimal 21.

name

Specify the name of the storage entity (source) containing the data for the specified input card. If *name* is omitted, the name used is the name defined for the input card with the PC-based path information and file extension removed.

Note: *name* must be preceded by one or more spaces.

Execution Log (-L)

Use the Execution Log execution command (-L) to specify that an execution log is produced. The default is to write the execution information to a temporary storage queue name **DTXLOG**. You can override the default by specifying the storage medium to be used and the name of the log. See [Specifying Storage Medium](#) for details.

-I[@storage_codeName]

Options

Description
<i>storage_code</i>
One of the following codes, which represent the type of storage medium used for the execution log
T = Specify a temporary storage queue.
D = Specify a transient data queue.
R = Specify VSAM relative-record dataset.
K = Specify VSAM key-sequenced dataset.
E = Specify VSAM entry-sequenced dataset.
<i>Name</i>
The name for the execution log
Note: <i>name</i> must not be preceded by any spaces.

Output Dataset Override (-O)

Use the Output Dataset Override (-O) execution command to override the specified output card target with a dataset for this execution of the map only. For example, use this command to override the name for a specific output data target or to specify the appending of data files, file deletion, or rollback.

You can also specify the storage medium, record length, delimiters to use for record separators, and so on. See [Specifying Storage Medium](#), [Specifying Output Fixed Format Record Length](#) and [Variable Length Records](#) for more information about specifying these options.

When using an input or output override execution command (-I) or (-O), the source or target name must be preceded by one or more spaces. If the override specifies a storage buffer, using an override name is not applicable. Names for the data and execution audit, trace, and problem resolution information are not separated from the command by spaces.

**-O[F]card_num[+|!+][B][[@storage_code][#record_length]
[:{W|P[delimiter_list]}]] [name]**

Option

Description
F
Optional for compatibility with previous versions.
<i>card_num</i>
The number of the output card to override.
+
Append the current map execution data to an existing data file.
!+
Do not append the data to an existing data file, but create a file each time the map is run.
B
If map execution is not successful, rollback any changes made to the file and reinstate the file to the original state prior to map execution.
@ <i>storage_code</i>
One of the following codes that represent the type of storage medium to be used for output data. The default is a temporary storage queue with the output card name defined without its path and file extension.
T = Specify a temporary storage queue.
D = Specify a transient data queue.
R = Specify VSAM relative-record dataset.
K = Specify VSAM key-sequenced dataset.
E = Specify VSAM entry-sequenced dataset.
B = Specify storage buffer.
If the storage buffer option (B) is specified, then specifying an override name is not applicable.
<i>record_length</i>
The fixed output record length specified for writing the data that is less than or equal to 32768. The default is to write the data as 4096-byte fixed length records. For VSAM, the record length is the record length defined for the dataset.
:W
The data is mapped into workspace and then copied from workspace to its final target after mapping completes. Variable length records are separated out during the copy operation based on a defined record separator. See The :W Option .
:P
The data <i>profiled</i> , that is, a table is built in memory that cross-references record identifiers to corresponding file offsets. For details, see The :P Option .
<i>delimiter_list</i>
A comma-delimited list of character or hex values. (Xnn is the syntax for indicating a hexadecimal value.) The default delimiter is a single byte with the value hex 15, decimal 21.
<i>name</i>
The name for the temporary storage entity (target) where data for the specified output card will be written. If <i>name</i> is omitted, the name used will be the name defined for the output card without the PC-based path information and file extension.
<i>name</i> must be preceded by one or more spaces.

Enqueuing on Data Files (-Q)

Use the Enqueuing on Data Files execution command (-Q) to specify this execution command to enable enqueueing on data sources and targets for maps that run concurrently in a z/OS CICS region and share inputs or outputs or both inputs and outputs. By default, I/O calls to and from input and output files are not serialized.

-Q

Trace (-T)

Use the Trace execution command (-T) to turn off tracing or to specify a particular card or a range of input or output data objects or both input and output data objects to trace. When the -T command is used without other options specified, no trace log is produced. You can also produce a trace summary and specify the storage medium to use for the trace report. The default is to write the trace information to temporary storage queue DTXTRCE. You can override the default by specifying the storage medium to be used and the name. See [Specifying Storage Medium](#) for details.

**-T[I|ICcard_num|Ifrom[:to]|
[O|OCcard_num][S][@storage_codeName]**

Option	Description
I	Produce detailed trace information for all output cards
ICcard_num	Produce detailed trace information for a single input card. For example, the command -TIC3 only produces detailed trace information for input card 3.
Ifrom:to	Produce detailed trace information for a range of input objects. For example, to produce trace information for objects 1000 to 1320, the option could be specified as -TI1000:1320 . The upper limit of the object range is optional such that to trace from input object 1000 to the end of the file, the option could be specified as -TI1000 .
O	Produces detailed trace information for all output cards
OCcard_num	Produce detailed trace information for a single output card. For example, the command -TOC2 produces detailed trace information only for output card 2.
S	Produce summary trace information for input and output cards as specified, providing a message for each card. For example, the message for a particular card may be Input 1 was valid, but 34 bytes of unknown data found or Output 2 built successfully .
storage_code	One of the following codes that represent the type of storage medium used for the trace report. The default storage medium is transient data queue CSML. T = Specify a temporary storage queue. D = Specify a transient data queue. R = Specify VSAM relative-record dataset. K = Specify VSAM key-sequenced dataset. E = Specify VSAM entry-sequenced dataset.
name	The name for the trace report. <i>name</i> must not be preceded by any spaces.

Workfiles in Memory (-WM)

Use the Workfiles in Memory execution command (-WM) to override the default allocation of work files on disk and allocate them as memory files.

-WM

Problem Resolution (-\$)

Use the Problem Resolution execution command (-\$) only under the direction of Customer Support.

This command produces output that is unintelligible to the user but is required by Customer Support for debugging.

-\$

Ensure you use the hexadecimal value 0x5B for the '\$' character on US EBCDIC systems. Regardless of the character displayed in the JCL, using Hex 0x5B is essential to make this option work and produce the debug output if Support requests it.

Print Uppercase (+)

Use the Print Output to Uppercase execution command (+) to cause all printed output, execution logs, audit logs, and trace logs to be written in uppercase.

+*

Eliminate AICA (ICVR Timeout) Abends (-!n)

Use the Eliminate AICA (ICVR Timeout) Abends execution command (-!n) to issue an EXEC CICS SUSPEND being issued after every *n* objects found or built by the map. The EXEC CICS SUSPEND resets the ICVR timer and eliminates AICA abends caused by long-running maps that do a large amount of processing in CPU. Be aware that

using this command could adversely affect map performance if other tasks with higher priority are executing in the z/OS CICS region at the same time as the map.

-!n

Specifying storage medium

The z/OS CICS Execution Option can map from or to any of six different storage media.

You would specify the storage media by using the following codes:

- **T** = Specify a temporary storage queue.
- **D** = Specify a transient data queue.
- **R** = Specify VSAM pre-defined relative-record dataset.
- **K** = Specify VSAM pre-defined key-sequenced dataset.
- **E** = Specify VSAM pre-defined entry-sequenced dataset.
- **B** = Specify storage buffer.

Storage media can be specified for data sources and targets as well as execution log, trace, and audit output. For a data source or target, the default storage medium is a temporary storage queue with the input or output card name defined without the PC-based path and file extension.

A particular storage medium can also be specified for data audit, execution log, trace, and problem resolution information:

**This temporary storage queue
is the default medium for the...**

DTXLOG
 execution log
DTXAUD
 audit log
DTXTRCE
 trace
DTXDEBG
 debug listing

The following examples demonstrate how to specify the storage medium option with the applicable execution commands. In these examples, it is assumed that the storage medium for input and output data files comprises one or more records of a consistent, fixed length.

Example

Example	Description
-I1@D INPQ	Specifies data for input card 1 resides in a TDQ called INPQ
-O1 OUTPUTQ	Specifies data for output card 1 is written to a TSQ named OUTPUTQ
-I3@B	Specifies data for input card 3 resides in a storage buffer
-I2@K MYKSDSIN	Specifies input 2 is a VSAM KSDS named MYKSDSIN
-O2@E MYESDSOU	Specifies output 2 is a VSAM ESDS named MYESDSOU
-O2@R	Specifies output 2 is written to a VSAM RRDS named without a PC path or file extension
-L@TDTXLOG	Specifies that an execution log is written to a TSQ named DTXLOG
-A@DDTXAUD	Specifies that an audit log file is written to a TDQ named DTXAUD
-TI@TDTXTRCE	Specifies that an input trace file is written to a TSQ named DTXTRCE
-\$@TDTXDEBG	Specifies that a debug listing is written to a TSQ named DTXDEBG

Note: When using an input or output override execution command (-I) or (-O), the source or target name must be preceded by one or more spaces. If the override specifies a storage buffer, using an override name is not applicable. Names for the data and execution audit, trace, and problem resolution information are not separated from the command by spaces.

- [Specifying output fixed format record length](#)
- [Variable length records](#)

The z/OS CICS Execution Option views inputs and outputs as byte stream files rather than as collections of records.

Specifying output fixed format record length

Output data to transient data queues and temporary storage queues is, by default, written as 80-byte fixed length records. Use the Output File Override execution command (-O) to override the default and specify a different fixed output record length that is less than or equal to 32768.

The following examples demonstrate how to specify the fixed output record length using the Output File Override (-O) execution command:

Example

Example	Description
-O2@D#72 OUTQ	

Data is written to a transient data queue named **OUTQ** in 72-byte fixed-length records.
-01#256 MYTSQ
Data is written to a temporary storage queue named **MYTSQ** in 256-byte fixed-length records.

Note: Fixed format record length cannot be specified for input cards and storage buffers. For VSAM, the record length is the record length defined for the dataset.

Variable length records

The z/OS CICS Execution Option views inputs and outputs as byte stream files rather than as collections of records.

The CICS Execution Option requires support for random positioning to any byte offset within an input or output file. Random positioning is easily implemented for fixed length records, because any file offset can be easily converted to a record number and a byte offset from the start of the record; however, it is more difficult for variable length records.

In the CICS Execution Option, variable length records are handled either by copying the data to or from fixed format workspace or by building a table to cross-reference record identifiers to file offsets. You can designate the method to be used by specifying the :W or :P options when using the Input File Override (-I) and Output File Override (-O) execution commands.

- **The :W option**

When the :W option is specified using the Input File Override (-I) execution command, variable length records are reformatted by copying them from the input file into fixed format workspace before mapping begins and the map will take its input not from the input file but from workspace.

- **The :P option**

The :W option

When the :W option is specified using the Input File Override (-I) execution command, variable length records are reformatted by copying them from the input file into fixed format workspace before mapping begins and the map will take its input not from the input file but from workspace.

When this option is specified using the Output File Override (-O) execution command, data is mapped into fixed format workspace and then copied to its final target after mapping completes. Variable length records are separated out during the copy operation based on a defined record separator.

The following examples demonstrate how to specify the :W option using the Input File Override (-I) and Output File Override (-O) execution commands:

Example

Description

-I1:W MYINPUT

Input 1, **MYINPUT**, is a TSQ of variable length records; copy it to fixed format workspace before mapping it.

-O1:W MYOUTPUT

Output 1, **MYOUTPUT**, is a TSQ that will contain variable length records. Data is mapped to fixed format workspace and after mapping completes, variable length records are separated out and written to the queue.

Output files of variable length records must have a delimiter or delimiters at the end of each record so that the copy operation can detect where one record ends and the next begins. Output record separators are defined in and built by the map. The default delimiter expected is a single byte with the value hex 15, decimal 21. Optionally, you can use different delimiters by specifying them as a comma-delimited list of character or hex values. (*Xnn* is the syntax for indicating a hexadecimal value.)

The following example demonstrates how to specify a delimiter list using the Input File Override (-I) and Output File Override (-O) execution commands:

Example

Description

-O1:WXOD,XOA

Output data is mapped to fixed format workspace and records are copied to a TSQ of variable length records using the delimiter list of hex 0D 0A as record separators. In other words, when a 0D 0A byte pair is sensed in the output stream, it is understood that it indicates the end of a record. Record separators are removed.

Because reads against transient data queues are destructive and multiple passes may need to be made against them, data mapped to and from TDQs take place in workspace, regardless of the record format.

The :P option

When the :P option is specified using the Input File Override (-I) or Output File Override (-O) execution command, the data is *profiled*, that is, a table will be built in memory that cross-references record identifiers to corresponding file offsets. It is the responsibility of the map to place record separators in the output data so that the profiling operation can interpret where one record ends and the next begins.

The following examples demonstrate how to specify the :P option using the Input File Override (-I) and Output File Override (-O) execution commands:

Example

Description

-I1:P MYINPUT

Input 1 is a TSQ of variable length records; it will be profiled.

-O1@K:PX25 MYKSDS

Output 1, VSAM KSDS **MYKSDS**, contains variable length records; build a profile of the output using hex 25 as a record separator.

When using the :W or :P options, a delimiter list can also be specified for record separators to be used for input cards.

This can be useful in a situation in which you defined a map on the PC that reads a text file as input. A text file, by definition, consists of lines delimited by carriage-return/line-feed pairs (0D 0A). If you transfer this data to a z/OS environment as TEXT, the record separators are removed and variable length records are created. Mapping this data is a problem because the map as defined expects each record to be terminated by a carriage-return/line-feed pair that no longer exists. You can specify the following command string:

```
-I1:PX0D,X0A MYINPUT
```

This example command string would result in the bytes 0D 0A being added to the end of each record at the time it is read in.

Using EXEC CICS LINK passing storage buffers

When the z/OS CICS Execution Option is called by an EXEC CICS LINK, it is often because you want to pass to the CICS Execution Option the address of one or more storage buffers containing input data or the address of one or more storage buffers that will contain mapped output data or both address types.

Suppose that you want to call the CICS Execution Option from a z/OS CICS COBOL program using an EXEC CICS LINK and execute a map named **MAPA**. This map has one input and one output, both of which exist as storage buffers. To do this, the COMMAREA passed to the CICS Execution Option should be defined as:

```
01 DTX-LINK-BLOCK-B.
  05 MRLB-VERSION          PIC X(04) VALUE 'B002'.
  05 MRLB-PARM-COUNT       PIC 9(08) COMP VALUE 0.
  05 MRLB-PARM-LIST-ADDRESS    POINTER VALUE NULL.
  05 MRLB-RETURN-CODE      PIC 9(04) COMP VALUE 0.
  88 MRLB-NORMAL-RETURN    VALUE 0.
  05 MRLB-RETURN-MESSAGE    PIC X(80) VALUE SPACE.
  05 MRLB-MAPPING-RETURN-CODE PIC 9(04) COMP VALUE 0.
  88 MRLB-NORMAL-MAP-RETURN VALUE 0.
```

Next, code a working-storage area similar to the following example to hold the command line. The command line has card overrides for input 1 and output 1 indicates that the source and target exist as storage buffers.

```
01 MAP-COMMAND-LINE
  05 MAP-COMMAND-STRING   PIC X(16) VALUE 'MAPA -I1@B -O1@B'.
  05 FILLER               PIC X(01) VALUE LOW-VALUE.
```

MRLB-PARM-LIST-ADDRESS in DTX-LINK-BLOCK-B should contain the address of a parameter list such as the following example. The parameters need not be in this exact order but the command line must be the first parameter in the list.

```
01 PL00-PARM-LIST.
  05 PL01-MAP-COMMAND     POINTER.
  05 PL02-I1-BUFFER-PARM  POINTER.
  05 PL03-O1-BUFFER-PARM  POINTER.
```

PL01-MAP-COMMAND should contain the address of the null-terminated Command Line, MAP-COMMAND-LINE, defined above. PL02-I1-BUFFER-PARM should contain the address of the following working storage area:

```
01   DTX-DATA-BUFFER.
  05 MRDB-DATA-LENGTH        PIC S9(8) COMP.
  05 MRDB-DATA-ADDRESS       POINTER.
  05 MRDB-DATA-NAME-AREA    PIC X(04).
  05 FILLER REDEFINES MRDB-DATA-NAME-AREA.
    10 MRDB-DATA-NAME-L2      PIC X(02).
    10 MRDB-NULL-TERMINATOR-L2 PIC X(01).
    10 FILLER                 PIC X(01).
  05 FILLER REDEFINES MRDB-DATA-NAME-AREA.
    10 MRDB-DATA-NAME-L3      PIC X(03).
    10 MRDB-NULL-TERMINATOR-L3 PIC X(01).
```

MRDB-DATA-LENGTH should contain the actual length of the data in the buffer for input 1. MRDB-DATA-ADDRESS should contain the address of the buffer for input 1. MRDB-DATA-NAME-AREA should contain the null-terminated string "I1", indicating that this buffer is to be used as input 1.

PL02-O1-BUFFER-PARM should contain the address of an identical working storage area initialized as follows: MRDB-DATA-LENGTH should contain the full size of the buffer allocated for output 1. MRDB-DATA-ADDRESS should contain the address of the output buffer. MRDB-DATA-NAME-AREA should contain the null-terminated string "O1", indicating that this buffer will hold the mapped data for output 1.

MAPLB-PARM-COUNT should be set to 3 (1 for the command line + 1 for the input buffer + 1 for the output buffer). If more or fewer parameters are to be passed, make sure that you adjust the value of MAPLB-PARM-COUNT accordingly. See the **DTXSCTST** sample program.

Using the EXIT function

The **EXIT** function is available when entering mapping rules for fields using the Map Designer on your PC.

Use this function to exit to a user-written program to get data for mapping to a field. See the *Functions and Expressions* documentation. However, there are special considerations when using this function in conjunction with the z/OS CICS Execution Option.

A user exit is invoked by calling the **EXIT** function in a mapping rule using the Map Designer. The **EXIT** function expects three arguments, as seen in the following example:

```
EXIT ("non_null_value", "load_module_name", "parameters")
```

When using the CICS Execution Option:

- The first argument contains a value that is not a null string (a single space is sufficient).
- The second argument contains the name of the exit program (the load module name).
- The third argument is a text item containing parameter information for the exit program. This parameter text item may be up to 64K bytes in length, and the EXIT function returns a resulting text item, which also may be up to 64K bytes in length. Execution of this mapping rule causes the text item returned by the program *exit-name* to be placed in the field to which you are mapping data.

Exit programs must be defined as z/OS CICS programs. Any static data areas declared within an exit program, such as COBOL working storage, do not persist across multiple calls to the exit. Four arguments are passed to the user exit program in the COMMAREA:

- The address of a full word that contains the length of the parameter text item.
- The address of a space-padded buffer of up to 64K bytes that contains the parameter text item.
- The address of a full word that contains the length of the resulting text item.
- The address of a buffer of up to 64K bytes into which the resulting text item will be placed.

For an example of source code for a COBOL user exit program, see DTXSCXIT in the DTX.SDTXSAMP PDS included in the IBM Transformation Extender installation.

Using the RUN function

The first argument of the **RUN** function is the name of the map to be executed. Under z/OS CICS this must be the name of the map as it exists in the map KSDS. At run-time, the CICS Execution Option converts the map name specified in the **RUN** function by removing any drive, path, and file extension information if it exists. For example, the following map rule will execute the map **MAP_10**:

```
RUN ("install_dir\mymaps\map_10.mmc")
```

When using the **RUN** function, by default, the CICS Execution Option execution log, audit log, trace, and problem resolution output are written to temporary storage queues **RUNLOGnn**, **RUNAUDnn**, **RUNTRCnn**, and **RUNDBGnn**, respectively, where *nn* is the recursion level of the map greater than or equal to 01. You can override these targets as described above.

A map executed using the **RUN** function is loaded once and is removed from storage when the task ends or when the map executed is different from the last map executed. This means that maps repeatedly executed are not reloaded each time.

Preloading IBM Transformation Extender maps and .mdq files in a z/OS CICS environment

The CICS® preloader utility preloads IBM Transformation Extender maps and database query (.mdq) files into memory. When maps and .mdq files run frequently, you can reduce processing by preloading them into memory. Preloaded files do not require I/O processing.

The CICS preloader loads maps and .mdq files from a partitioned data set (PDS or PDSE) or from a z/OS® flat file.

- [Configuring the CICS preloader utility](#)
- [Map and .mdq file DD statements in the CICS startup JCL](#)
The CICS preloader loads maps and .mdq files from a partitioned data set (PDS or PDSE) or from a z/OS® flat file.
- [Running the IBM Transformation Extender CICS preloader utility](#)
The name of the preloader utility is DTXnnnLD, where *nnn* is the IBM Transformation Extender version. For example, the name of the preloader for IBM Transformation Extender V8.4.1 is DTX841LD.

Configuring the CICS preloader utility

To configure the CICS® preloader utility:

1. Create a preloader control file.
2. In the preloader control file, identify the maps and .mdq files that you want to load.
3. Add DD statements to the CICS startup JCL for:
 - The preloader control file
 - The maps and .mdq files that you want to load

These steps are described in detail in the topics that follow.

- [The preloader control file](#)
A control file identifies the IBM Transformation Extender maps and database query (.mdq) files that the IBM Transformation Extender CICS preloader utility loads into memory. You allocate the DTXnnnIN preloader control file in the CICS startup JCL.
- [Identifying the maps and .mdq files in the DTXnnnIN preloader control file](#)
In the DTXnnnIN CICS preloader control file, identify the IBM Transformation Extender maps and database query (.mdq) files that you want to preload into memory.

The preloader control file

A control file identifies the IBM Transformation Extender maps and database query (.mdq) files that the IBM Transformation Extender CICS preloader utility loads into memory. You allocate the DTXnnnIN preloader control file in the CICS startup JCL.

The *ddname* of the preloader control file resource must be DTX*nnn*IN, where *nnn* is the IBM Transformation Extender version, for example DTX841IN. The preloader control file must have a fixed-block record format (**RECFM=FB**).

For example, the following DD statement allocates a preloader control file that is named KEYWORD.MAP in the CICS startup JCL:

```
//DTX841IN DD DSN=KEYWORD.MAP DISP=SHR
```

The preloader control file cannot be the D*nnn*MAP VSAM file (where *nnn* is the IBM Transformation Extender version).

Identifying the maps and .mdq files in the DTX*nnn*IN preloader control file

In the DTX*nnn*IN CICS preloader control file, identify the IBM Transformation Extender maps and database query (.mdq) files that you want to preload into memory.

1. Edit the DTX*nnn*IN preloader control file.
2. Specify each map file and .mdq file in the format:

```
MAPNAME=MAP_TO_LOAD  
MDQNAME=MDQ_FILE_TO_LOAD
```

where *MAP_TO_LOAD* or *MDQ_FILE_TO_LOAD* is the name of the map or .mdq file. The contents of the preloader control file must be in uppercase. You can specify multiple *KEYWORD=FILE_NAME* pairs in the preloader control file. Specify multiple pairs on separate records or lines, or as a single space-delimited or comma-delimited record. For example:

```
MAPNAME=MAP_TO_LOAD,MDQNAME=MDQ_FILE_TO_LOAD,MAPNAME=MAP_TO_LOAD
```

Map and .mdq file DD statements in the CICS startup JCL

The CICS preloader loads maps and .mdq files from a partitioned data set (PDS or PDSE) or from a z/OS® flat file.

- **Map or .mdq files in a z/OS flat file**

The name of the map or .mdq file must be the *ddname* of the DD statement that allocates the resource in the CICS startup JCL. The data set name (DSN parameter) is the name of the z/OS flat file that contains the map.

- **Map or .mdq files in a PDS or PDSE**

When a map or .mdq file is a partitioned data set (PDS or PDSE), its DD statement determines how you specify it in the preloader control file.

Map or .mdq files in a z/OS flat file

The name of the map or .mdq file must be the *ddname* of the DD statement that allocates the resource in the CICS startup JCL. The data set name (DSN parameter) is the name of the z/OS flat file that contains the map.

For example, the PORDER.MAP file is a native z/OS flat file that contains the POMAP.MVS map. The SKUDB.MDQ file is a native z/OS flat file that contains the SKUMDQ.MDQ file. Use the following DD statements in the CICS startup JCL:

```
//POMAP DD DSN=PORDER.MAP, DISP=SHR  
//SKUMDQ DD DSN=SKUDB.MDQ, DISP=SHR
```

In the preloader control file, specify the map and .mdq files to preload:

```
MAPNAME=POMAP  
MDQNAME=SKUMDQ
```

Map or .mdq files in a PDS or PDSE

When a map or .mdq file is a partitioned data set (PDS or PDSE), its DD statement determines how you specify it in the preloader control file.

- If the *ddname* of the DD statement is the map or .mdq file name, the preloader control file specifies only the map or .mdq file name. For example, the following DD statement loads the TEST1 map from the TX.MAPS PDS:

```
//TEST1 DD DSN=TX.MAPS(TEST1),DISP=SHR
```

The preloader control file must specify the following MAPNAME:

```
MAPNAME=TEST1
```

- If the *ddname* of the DD statement is not the map or .mdq file name, the preloader control file specifies both the *ddname* and the map or .mdq file name. For example, the following DD statement loads the TEST1 map from the TX.MAPS PDS:

```
//MAPLIB DD DSN=TX.MAPS,DISP=SHR
```

In this case, the preloader control file must specify both the MAPLIB *ddname* and the TEST1 map name:

```
MAPNAME=MAPLIB(TEST1)
```

Running the IBM Transformation Extender CICS preloader utility

The name of the preloader utility is DTX nnn LD, where nnn is the IBM Transformation Extender version. For example, the name of the preloader for IBM Transformation Extender V8.4.1 is DTX841LD.

You can run the utility only when no other mapping process is active. The DTX nnn IC and the DTX nnn IK programs must have a RESCOUNT of 0. If a mapping process is active, the DTX nnn LD utility fails and writes a message to SYSPRINT DD.

You can execute the DTX nnn LD utility:

- As a transaction (DTLD)
- By linking to the DTX nnn LD program
- By adding the DTX nnn LD program as a second-phase initialization PLTPI

You can run the CICS preloader utility multiple times. Each time it runs, it frees the maps from memory and reloads them from disk.

If you do not preload a map, load it into the D nnn MAP VSAM file (where nnn is the IBM Transformation Extender version) to run it.

Abnormal termination

If a task terminates abnormally with an internal error, the return code 16 and an error message are passed back to the calling program.

Diagnostic information is written to the execution log if one is being produced or to the system transient data queue **CSML** if there is no execution log. Diagnostic information for a task can be identified by the task ID that occupies the first eight bytes of each line of output. All temporary files in workspace are deleted before the task ends.

The DTXMPULD utility

In addition to loading maps into the KSDS, where maps are stored for use in the z/OS CICS region as described above, the DTXMPULD utility has the capability of listing and deleting maps from the **KSDS** map.

The command syntax is as follows:

```
DELETE=map_name_or_mask  
LIST=map_name_or_mask
```

Either or both of these can be used in combination with the INPUT command that specifies maps to be loaded into the VSAM **KSDS**.

The parameter *map_name_or_mask* can be used to specify a single map or a collection of maps if you use the wildcard character *. For example, DELETE=MAPA will delete only MAPA from the **KSDS** map. DELETE=MAPA* deletes all maps with names beginning with MAPA. LIST=* lists all maps stored in the VSAM dataset. DELETE=* deletes all maps.

To use either the DELETE or LIST, the DDNAME MAPKSDS must identify the map **KSDS**:

```
//MAPKSDS DD DSN=map_KSDS,DISP=SHR
```

If the INPUT command is not used, no output is generated for loading into the map **KSDS**. Consequently, the SORT and REPRO steps of the **DTXCI** JCL should not be executed.

Resource names

For information about creating resource names for use on the CICS Execution Option for the z/OS CICS platform, see [Resource Names for z/OS Platform](#).

Multiple IBM Transformation Extender versions

This section presents information about running multiple versions of the IBM Transformation Extender CICS Execution Option concurrently in a single CICS region.

This feature will help you migrate to additional releases of the CICS Execution Option without the need to define and configure additional CICS regions. You will also be able to do a staged migration, or phase-approach, to a new release while your existing release is running in that same CICS region.

- [Enabling the multiple IBM Transformation Extender feature](#)

This feature is enabled by making the names of the files and programs you are currently installing unique to the names used in other releases you have installed previously.

- [Using the multiple installations](#)

After the new release of the CICS Execution Option has been installed, you need to be able to distinguish between the multiple installations and then invoke them.

Enabling the multiple IBM Transformation Extender feature

This feature is enabled by making the names of the files and programs you are currently installing unique to the names used in other releases you have installed previously.

It is done by embedding the product version number, *nnn*, in the resource name, D*nnn*MAP. An example is to specify D810MAP as the resource name you are currently installing, while MERCMAP is the resource name used in your previous install.

With this resource naming convention, you must update the CICS resource definitions for each release of CICS Execution Option and duplicate the associated VSAM datasets.

The result is that a completely separate copy of the product is installed and none of its resources are shared with any other version of the product that exists in the same CICS region. The user Dynamic Storage Area used by CICS Execution Option components will effectively be doubled.

Using the multiple installations

After the new release of the CICS Execution Option has been installed, you need to be able to distinguish between the multiple installations and then invoke them.

There are two ways to identify and run multiple installations of the CICS Execution Option depending on which method you use to invoke the CICS Execution Option.

- If you invoke the CICS Execution Option through the EXEC CICS START command, Transient Data Queue trigger, or by entering the transaction ID at a terminal
 - use the transaction ID **D_{nnn}** (where *nnn* is the version of the release you have installed) and **MERC** or **DSTX** depending on the specific release you currently have installed.
- If you invoke the CICS Execution Option through the EXEC CICS LINK command
 - use the DTX*nnn*KI (used with XPLINK) or DTX*nnn*CI program names (where *nnn* is the version of the release you have installed) for the new version of the CICS Execution Option and MERCCICS or DSTXCICS depending on the specific release you currently have installed.
Note: If you do not need to use the multiple CICS Execution Option feature, you can use the alias name, **MERCCICS** for the program name for backward compatibility.

Resource names for z/OS platform

These are the details for creating resource names for use on z/OS Batch and CICS platforms.

Resource names created in the Resource Registry can be used in maps running on z/OS platforms. However, there are a few things to take into consideration:

- Resource values must conform to z/OS rules for their types, such as DDNAMES and Database names.
- Virtual server defined for the z/OS platform must be named **MVS SERVER**
- Resource name file (**.mrn**) must be copied to z/OS platform
- Name and location of resource name file are specified in either **DTXMRN** DDNAME for Batch region or **DTXMRN** VSAM dataset definition for CICS region
- Resource configuration file (**.mrc**) is not required
- **Resource name aliases**
Because the z/OS platform does not recognize file path information, the IBM Transformation Extender automatically truncates the path specified in input and output cards when the map is executed.
- **Virtual server names**
When creating virtual servers to represent the z/OS deployment environment, **MVS SERVER** (case-sensitive) must be defined as the virtual server name.
- **Specifying resource name file for batch**
- **Specifying resource name file for CICS**
- **Resource configuration files**

Resource name aliases

Because the z/OS platform does not recognize file path information, the IBM Transformation Extender automatically truncates the path specified in input and output cards when the map is executed.

Therefore, it is not necessary to create a resource name that has a value defined for a file path.

- **Resource name limitations for files and DDNAMES**
To conform to z/OS conventions, the IBM Transformation Extender truncates file and resource names to 8 characters when a map is executed.

Resource name limitations for files and DDNAMES

To conform to z/OS conventions, the IBM Transformation Extender truncates file and resource names to 8 characters when a map is executed.

Therefore, resource names and their values have an 8-character limitation. Because this includes the percent (%) symbols that enclose the resource name when using it in a source or target resource, the resource name must not be longer than 6 characters. For example, the resource name **MyFile** is specified as **%MyFile%** when used in the **Command** setting of an input or output card.

The resolved resource value is actually a DDNAME that is defined on the z/OS Batch platform. The resolved value must conform to the rules for DDNAMES.

See "[DDNAMES](#)" for more information about defining DDNAMES for resolved resource values.

Virtual server names

When creating virtual servers to represent the z/OS deployment environment, **MVSSERVER** (case-sensitive) must be defined as the virtual server name.

MVSSERVER is the only server name the IBM Transformation Extender recognizes.

Even if other virtual servers are also defined in the same **.mrn** file, **MVSSERVER** is the only name recognized by the z/OS version of the IBM Transformation Extender as being the active server.

Specifying resource name file for batch

When associating a resource name file for the z/OS Batch version of the Command Server, the name and location of the **.mrn** file are specified in the **DTXMRN** DDNAME definition in a JCL (Job Control Language) file.

- [**DDNAMES**](#)

DDNAMES

DTXMRN is the only DDNAME recognized by the Command Server that contains the **.mrn** file specifications. A DD statement specifies the file name or path of the **.mrn** file.

To specify a resource name file for Batch

1. Create the resource name **.mrn** file in the Resource Registry.
2. Copy the **.mrn** file to the z/OS platform.
3. Update the JCL file.
 - Specify the **DTXMRN** DDNAME definition.
 - Include a DDNAME definition for each resolved resource value used.

Specifying resource name file for CICS

When associating a resource name file for the z/OS CICS version of the IBM Transformation Extender, the name and location of the resource name file are specified in the **DTXMRN** Virtual Storage Access Method (VSAM) dataset.

- [**VSAM**](#)

VSAM

JCL files are used when executing the IBM Transformation Extender in the CICS region just as they are for Batch. However, instead of specifying the **.mrn** file in a **DTXMRN** DDNAME definition in the same JCL file in which the job steps that invoke the IBM Transformation Extender are also specified, the **.mrn** file is specified in a **DTXMRN** VSAM dataset definition submitted in a separate JCL file. The **DTXMRN** VSAM dataset definition includes the name and location of the **.mrn** file.

To specify a resource name file for CICS

1. Create the resource name **.mrn** file in the Resource Registry.
2. Copy the **.mrn** file to the z/OS CICS platform and then load it into a VSAM dataset.
3. Update the separate JCL file.
 - Specify the **DTXMRN** DDNAME definition to define this VSAM dataset to the CICS region.
 - Include a CICS resource definition for each resolved resource value used.

Only one **DTXMRN** VSAM dataset can be used while the CICS region is running, therefore, only one resource name file can be referenced during this duration. Because of this, it is suggested that you define all resource names and their associated values that will be used on the z/OS CICS version of the IBM Transformation Extender in one resource name file.

Resource configuration files

A resource configuration file (**.mrc**) is not used by the z/OS version of IBM Transformation Extender to determine which **.mrn** file to use and what server is active. Instead, the **.mrn** file is specified in either a DD statement for the **DTXMRN** DDNAME or in the **DTXMRN** VSAM dataset definition, and **MVSSERVER** is recognized as the only active server for the z/OS version of IBM Transformation Extender.

Because a resource configuration file is not required, it is not necessary to specify an **.mrc** file in the Map Designer or the **config.yaml** file.

Launcher introduction

The Launcher automates the execution of systems of maps and can control multiple systems.

On Windows platforms, the Launcher runs as a multi-threaded service. On UNIX platforms, the Launcher runs as a multi-threaded *daemon*. For platform and version-specific information, see the [Release Notes](#).

Note: When running the Launcher on z/OS® through z/OS UNIX System Services (z/OS UNIX), follow the instructions for UNIX. The z/OS environment is treated as another UNIX system. Any differences between the standard UNIX and z/OS usages are noted where appropriate.

- **[Installation considerations](#)**

See the [release notes](#) for installation details.

- **[Optional local z/OS administration interfaces](#)**

There are optional local z/OS administration interfaces available for configuring and managing the Launcher on z/OS operating systems.

- **[Launcher overview](#)**

The Launcher runs systems of maps that are created and generated using the Integration Flow Designer (IFD).

- **[Using the Launcher](#)**

To use the Launcher, you must complete a series of steps.

Installation considerations

See the [release notes](#) for installation details.

Note: Ensure that the hostname 'localhost' resolves to 127.0.0.1 or Launcher execution (**launcher.sh**) will produce unexpected results.

Note: <install_dir> refers to the directory where your product is installed.

Optional local z/OS administration interfaces

There are optional local z/OS® administration interfaces available for configuring and managing the Launcher on z/OS operating systems.

The two optional local z/OS administration interfaces ("[Local z/OS Administration Interfaces](#)") are:

Launcher overview

The Launcher runs systems of maps that are created and generated using the Integration Flow Designer (IFD).

These systems of maps that are generated specifically to run in the Launcher are called system files (**.mst**), sometimes referred to as Launcher control files.

When the Launcher starts running, it is initialized with **.mst** files in the deployment directory.

The new HTTP Listener component of the Launcher provides secure, two-way communication between external clients and the Launcher. An external HTTP client can trigger a map in a Launcher system by sending a request to the Launcher HTTP Listener. Regardless of whether it runs successfully, the map returns a response through the HTTP Listener to the external HTTP client. If the map fails or the output card does not return a response to the client, the input card automatically returns a response. For more information on the HTTP Listener, see [Launcher HTTP Listener overview](#).

You can specify a deployment directory from the Launcher Administration interface. (For more information about this, see "[Launcher Administration](#)".)

Note: When using the z/OS® local administration interfaces, the deployment directory defaults to *install_dir/systems*.

- **[Launcher HTTP Listener overview](#)**

The HTTP Listener component of the Launcher provides secure, two-way communication between external clients and the Launcher. The HTTP Listener is included in the Design Studio installation. The HTTP Listener can automatically recognize an MTOM message and store the attachments in a user-designated location.

Launcher HTTP Listener overview

The HTTP Listener component of the Launcher provides secure, two-way communication between external clients and the Launcher. The HTTP Listener is included in the Design Studio installation. The HTTP Listener can automatically recognize an MTOM message and store the attachments in a user-designated location.

An external HTTP client can trigger a map in a Launcher system by sending a request to the Launcher HTTP Listener. Regardless of whether it runs successfully, the map returns a response through the HTTP Listener to the external HTTP client. If the map fails or the output card does not return a response to the client, the input card automatically returns a response.

The HTTP Listener executable name is **httpslsnr**. This functionality supports the following features:

- Allows you to configure HTTP Listeners in Launcher Administration.
 - Add a new HTTP Listener
 - Remove an existing HTTP Listener
 - Update an existing HTTP Listener configuration
- Runs HTTP Listeners that are enabled during the start up of the Java Launcher.
 1. The Java Launcher spawns HTTP Listener processes.
 2. The HTTP Listener process listens for client requests based on user settings.
 3. The HTTP Listener provides status information to clients.
 4. The HTTP Listener allows clients to configure non-start up settings.
 5. The Java Launcher restarts HTTP Listener if they terminates abruptly.
- Allows you to monitor the HTTP Listeners from Management Console.
 - Monitor only those HTTP Listeners that are of interest to you
 - Provide Summary, Status, and Configuration statistics
 - Provide a manual refresh for status statistics, in addition to the automatic refresh
- Provides command line interface support for HTTP Listeners.

- Define HTTP Listeners with Launcher Administration command line utility
- List HTTP Listeners defined by the user
- Export and import HTTP Listener definitions in XML format
 - Note: The Java Launcher can start HTTP Listeners with Launcher Administration XML.
- Report the status of the HTTP Listener using Java Launcher command line utility
- List HTTP Listeners running under Java Launcher
- Provides status and configuration information updates for clients through the Management Console and web browsers.
 - View status and configuration updates
 - Audit HTTP Listener activity
- Provides command line execution.
 - Includes many command line options for easy configuration and fast startup
 - Provides monitoring, auditing, and tracing from the command line console
 - Registers HTTP Client requests quickly regardless of the volume of requests
- Provides remote control access.
 - Handles updates and real-time monitoring from a browser on a remote machine
 - Refreshes the web browser automatically in custom intervals depending on the window visibility
- Allows real-time monitoring of activity.
 - Reports the total number of HTTP client requests and the type of Launcher response
 - Reports the average map execution times per HTTP Adapter watch
 - Tracks the total number of connections and averages the elapsed time for the establishment of sessions
 - Reports the names and sizes of the audit and trace files
 - Displays statistics on command line console, web browser, and Management Console
- Auditing
 - Reports each HTTP Client request and Launcher response in a XML schema format
 - Logs individual map completion times
 - Tracks each connection type and the elapsed time for session establishment
 - Activates on-demand and can be periodically archived once auditing is disabled
- Expanded Tracing
 - Provides unique names for the standard and verbose trace files
 - Tracks regular events in standard trace file and socket activity in verbose trace file
 - Activates on-demand and can be periodically archived once tracing is disabled
- Enhanced Triggering
 - Provides 2-way communication between the HTTP Adapter Input Card and HTTP Listener for added functionality and fast response times
 - Allows configurable HTTP response codes and messages based on map return code
 - Enables boilerplate and customized HTTP data responses that can include runtime map information
 - Guarantees timely HTTP Responses upon map failure

Using the Launcher

To use the Launcher, you must complete a series of steps.

The list below provides a high-level overview of the steps to follow after installing WebSphere® Transformation Extender.

Note: When running the Launcher daemon under z/OS® UNIX System Services (z/OS UNIX), by default, the Launcher daemon uses the value calculated by subtracting 1 from the starting listening port to stop, resume or pause the Launcher daemon. The starting listening port is configured under the General tab in the Launcher Administration interface.

Note: When using the z/OS local administration interfaces, see "[z/OS Launcher](#)" for more information.

- [**Basic steps**](#)

Basic steps

The following steps summarize the process flow of using the Launcher with the Management Tools.

1. From the Launcher Administration interface, configure the Launcher by setting up users, port numbers, and deployment directories.
2. Use the Integration Flow Designer to create a system and then generate the system (.msl) file to the deployment directory. For help on creating or generating a system file, see the *Integration Flow Designer* documentation.
3. Start the Launcher service or daemon.
4. Open a connection to the Launcher from the Management Console and begin viewing statistical data from the process that is running. From there you can also control the compound system that is running.
5. Open a connection to the Launcher from the Launcher Monitor to view watches that are running dynamically and to take snapshots.
6. Open the Snapshot Viewer to view snapshots taken by the Monitor.

Launcher architecture

The Launcher is an event-driven software model for linking and executing maps. Systems contain maps with the system workflow you specify by using the Integration Flow Designer.

A Launcher runs on a server platform and manages one or more Launcher processes (comprised of one or more systems) concurrently. Systems are asynchronous multi-threaded processes, whose atomic units are transactional maps. Systems can execute in a distributed environment where a Launcher resides on each server. When events occur, the Launcher is notified and starts maps based on that notification. Distributed data is received or routed during map execution.

Launcher functionality is based on the following premises:

- An event manager controls the initiation of maps based on sources to which a map subscribes.
 - A resource manager synchronizes shared data and timing interfaces among heterogeneous sources and targets.
 - Maps publish data that result from the content transformation of source data.
 - Maps are transactional processes with each map having its own error detection and recovery procedures.
 - The Launcher runs as a single process with multiple threads.
- [Triggers: Asynchronous communication](#)
 - [Synchronous coordination](#)
 - [System execution consistency](#)
 - [Map function and requirements](#)
 - [Defining a system](#)

Triggers: Asynchronous communication

The Launcher starts maps based on *triggers* that you identify using the IFD. Triggers are asynchronous events and time or source state changes. In some cases, there might be multiple triggers that must collectively exist before a map is initiated. The Launcher manages the coordination of these events to ensure that the correct set of circumstances have occurred before a map is initiated.

Although each instance of a map is considered a separate *watch*, the Launcher must also manage the resource interfaces used by other maps and other instances of the same map to properly synchronize these asynchronous events.

Synchronous coordination

Situations exist where synchronous interaction between maps is required. For example, you might not want to complete a transaction until you receive an acknowledgment of a particular action. Or you might want to communicate with a program or function to receive input data or send output data. Synchronous interaction can be performed in different ways:

- Communicate with other maps from a map rule during the data transformation process. Using a map rule in this manner with the Launcher manages coordination among maps to assure deadlock situations are avoided and shared resources are reliably accessed or deleted
- Synchronize coordination to interact with other applications using a resource adapter for a data source or target. For example, if you send a message by using the e-mail adapter, receipt notification might occur before the mapping process completes. When using resource adapters, you can configure recovery procedures for errors that occur during communication.

System execution consistency

System execution consistency implies data resources get to the right place at the right time with the right priority and that resources are manageable so that data or timing interfaces do not interfere with each other.

The smallest unit in a system configuration is a map. The definition of a map requires you to identify logical source and target data objects and rules for transforming source content to target content. For example, a logical data object can be an order, a ship notice, a news report, or a point-of-sale transaction. In this case, content transformation rules determine how an order and inventory schedule produces a ship notice.

If the transformation is consistent, the result is predictable. If the transformation is consistent, the same rules can execute on a variety of computing platforms; different sources and targets can be applied to the same logical data. For example, orders might be in a distributed database, inventory schedules might appear as messages, and ship notices might need to be contained in files.

Maps communicate when data can flow among them. To visualize that transaction data flow, the architecture of map interaction is defined as a system using the IFD. From a graphical diagram of system components, you can generate consistent control information to distribute maps and subsystems across different servers and execute all *atomic* maps in a system as an integrated whole with an Launcher.

Map function and requirements

The primary function of a map is to transform and route data from source formats to target formats. Sources and targets are independent objects, with respect to the metadata that specifies the content, the applications to which they might link, and the transport mechanisms used to obtain or deliver the content.

Defining a system

Using the IFD, you can define a system as a set of distributed subsystems, a set of maps, or a combination of maps and subsystems. You can create these definitions because a system is an isolated set of components whose interfaces are as explicit as the interfaces for an atomic map.

For example, this isolation enables the Launcher to manage a system as easily as it manages a map.

To complete successfully, a map needs the proper resources to be in place. For example:

- Every data source must be accessible.
- Every output must be routable to its assigned target.
- The server-required resources to perform the transformation (work files and audit logs, for example) must not be in use by any other process during the transformation process.

To accomplish this reliably, the transformation must be able to succeed or fail as a whole. That is, when a map does not complete successfully, the original data must persist in a form that is recoverable and partial results can be rolled back and made inaccessible to other maps or applications.

You can use a single map to simplify logical connections between multiple heterogeneous applications; direct links using application adapters; and indirect links using file, messaging, and database adapters.

Getting started

In this documentation, you will learn how to set up and start using the Launcher.

- [Creating systems](#)
To create Launcher processes (.msl system files) that run on your Launcher platform, you must use the Integration Flow Designer (IFD).
- [Launcher Administration](#)
- [Preparing to execute defined systems](#)

Creating systems

To create Launcher processes (.msl system files) that run on your Launcher platform, you must use the Integration Flow Designer (IFD).

Additionally, you can use the IFD to:

- Define systems and Launcher settings.
- Build maps for a target platform, if required.
- Move all files that are required for system execution (that is, ported map files, Launcher system files, and so on) to a target platform in a designated deployment directory. For example, generate the system file (.msl) to the deployment directory.

See *Integration Flow Designer* documentation for more information about systems.

Note: When using the z/OS® native administration interfaces, see "[z/OS Launcher](#)" for more information. The remainder of this topic does not apply.

Launcher Administration

The Launcher Administration is the interface to the Launcher from which you can do the following configuration tasks:

- Choose the deployment directories from which systems run.
 - Set up users and user access rights.
 - Select a listening port and port range.
 - Select a resource configuration file (.mrc).
 - Specify system properties for the Java™ Remote Method Invocation (RMI)
 - Configure automatic restart parameters.
- [Accessing Launcher Administration](#)
The Launcher Administration is included with the WebSphere® Transformation Extender package and runs on both Windows and UNIX platforms.
- [Using Launcher Administration](#)

Accessing Launcher Administration

The Launcher Administration is included with the WebSphere® Transformation Extender package and runs on both Windows and UNIX platforms.

To start the Launcher Administration (Windows)

Select Start > Programs > Transformation Extender > Launcher > Launcher Administration.

To start the Launcher Administration application (UNIX)

1. From the installation directory, type:

```
. setup
```

2. At the command prompt type:

```
launcheradmin.sh
```

Using Launcher Administration

Before starting the Launcher, it is recommended that you confirm the default settings. If you do not change settings in the Launcher Administration before starting the Launcher, the Launcher Administration uses the following default values:

- Automatic startup option is enabled.
- Automatic restart is enabled.
- Launcher service listening ports are 5015 and 5016.
- Port range is 7000 - 8000.
- There are no user profiles.
- The deployment directory is:
 - *install_dir\System* on Windows platforms
 - *install_dir/System* on UNIX platforms

At the minimum, you must set up at least one user profile before using the Launcher. The documentation describes all of the options.

Note: When you configure the Launcher to run with the multiple-processes feature enabled, you can edit some of the settings while it is running as described in "[Multiple Processes](#)".

When you configure the Launcher to run as a single Launcher process (Compound System), you cannot edit settings.

- [Automatic startup option](#)
- [Configuring automatic restart](#)

On distributed platforms, the Launcher restarts automatically when it fails. This option can reduce downtime in production environments. You can configure how long and how many times the Launcher attempts to restart, or you can disable automatic restart. Automatic restart is not supported on Enterprise (USS) platforms.

- [Connection ports](#)

- [User profiles](#)

- [Deployment directories](#)

- [Resource resolution](#)

- [Cluster support](#)

- [Firewall configuration](#)

- [HTTP Listener startup configuration](#)

An HTTP Listener routes HTTP requests from external clients to maps in the Launcher and returns a response to the external client. Use the Launcher Administration utility Options page and select Listeners to add, remove, and update HTTP Listeners.

- [Automatic restart for launchers and HTTP Listeners](#)

Launchers and HTTP Listeners can automatically restart when they unexpectedly stop running. Use the Launcher Administration utility Options page to enable automatic restart and to limit the number and duration of restart attempts.

- [About multiple Launchers](#)

Automatic startup option

In the Launcher Administration, the **Automatic startup** option is enabled by default. The **Automatic startup** option triggers the **.msl** system files located in the deployment directory to start running automatically upon starting the Launcher service or daemon.

Note: One or more system files appear in the Management Console as one "CompoundSystem". See "[Multiple Processes](#)" for additional information when using the multiple-processes feature.

If you disable the **Automatic Startup** option and then start the Launcher service, you must manually start the system from the Management Console. See "[Management Console](#)" for more information about the Management Console.

Configuring automatic restart

On distributed platforms, the Launcher restarts automatically when it fails. This option can reduce downtime in production environments. You can configure how long and how many times the Launcher attempts to restart, or you can disable automatic restart. Automatic restart is not supported on Enterprise (USS) platforms.

1. In the Options tab of Launcher Administration, select Restart.
2. Click Enable Configuration to enable or disable automatic restart. Restart is enabled by default.
3. In the Retry Attempts field, specify the maximum number of times the Launcher is to attempt to restart after a failure.
The default value is two restart attempts. If the specified number of restart attempts has not occurred when the Retry timeout period elapses, the Launcher continues to retry until it reaches the specified number of attempts.
4. In the Retry Timeout field, specify the maximum amount of time, in minutes, that the Launcher is to attempt to restart after a failure.
The default value is 10 minutes. There are five minutes between each restart attempt.

Related reference

- [Launcher Administration command line syntax](#)

Connection ports

A **Listening Port** is a network port that listens for incoming requests. The default listening ports are **5015** and **5016**. The second listening port is reserved for communicating with the Launcher Monitor and is displayed in a read-only field. The Launcher Administration application uses an increment of 1 to determine the second listening port number. The system-generated number for this port is based on the **Listening Port** value plus **1**. For example, if you change the listening port to **3050**, the second listening port number automatically changes to **3051**.

Note: When running the Java™ Launcher on UNIX platforms and using the stop, pause or restart command with the **launcher.sh** shell script command, the Java Launcher internally uses a port setting that is 1 less than the first **Listening Port** value. The default internal **Port** is **5014**, which is **1** less than the first **Listening Port**, **5015**.

The **Port Range** values determine the port range that is used for systems running in the Launcher. There are *three* ports used for each Launcher running. The default range is **7000-8000**. You can change the port range to any value except the **Listening Port** number plus 1 (for example, **5016**), which is reserved for communication with the

Launcher monitor.

Note: When using WebSphere® Transformation Extender for z/OS®, the specified **Listening Port** and **Port Range** ports should be added to the **TCPIP.ETC.SERVICES** file. These ports cannot be currently in use by another application. The purpose of specifying them in the file is to document that these ports are reserved and should not be used by another application. It does not prevent other applications from using them.

This is true for all platforms, although the services file name is different.

For more details, see the *IBM® z/OS Communications Server: IP Configuration Reference* documentation.

Note: See "[Multiple Processes](#)" for additional information about connection ports when using the multiple-processes feature.

Note: When running the Java Launcher on UNIX and LINUX platforms, the specified **Listening Port** and **Port Range** ports should be added to the etc/services file.

User profiles

Before starting the Launcher, you must create at least one user profile for Launcher authentication.

To add a user profile

- From the Access tab, click Add.

The User Properties window opens.

- Complete each field:

- User Name** - the user's name (up to 32 alphanumeric characters).
- Login Name** - the user's login ID for the network (up to 9 alphanumeric characters).
- Password** - the password for the **Login Name** (up to 12 alphanumeric characters).

By default, each action category (**Start/Stop; Pause/Resume; Monitor**) is set to **Revoke**, which means that this user does not have any access rights.

Note: The access rights that you grant to a user apply to all systems running in the Launcher.

- Under each action category (**Start/Stop; Pause/Resume; Monitor**), click **Revoke** to view a drop-down menu that contains the two options: **Grant** or **Revoke**.

- After you select the security level for each category, click OK to save changes.

The new user appears in the profiles list on the Access tab.

Note: You must set up users with the appropriate access rights in the Launcher Administration interface to view systems running in the Management Console.

About Access Permission

By default, the Launcher process has Local System account privileges on Windows platforms because it is run as a service. However, Local System account privileges are insufficient for performing tasks associated with mapped network drives.

If you have an issue with access permission on mapped network drives, the solution is to configure the Launcher service to log on using a specific account.

To configure user account information

- Open the Windows **Computer Management** console.
- Locate **Services** and double-click **WebSphere® Transformation Extender Launcher**.

The Properties window is displayed, from which you can make account changes.

Deployment directories

The deployment directories are where the **.msl** files are placed to run in the Launcher. The default deployment directory (**Systems**) is located in the **WebSphere® Transformation Extender** installation directory; however, you can modify the default path or add new deployment directories as needed.

Note: You can create several deployment directories, however all systems will run simultaneously. The statistics that are displayed in the Management Console ("Management Console") will include all systems running.

To add a deployment directory

- From the Deployment Directories tab, click Add.
The Select window opens.
- Navigate to the deployment directory.
- Click once on the directory and click Select.
The new directory is displayed in the directory list.

To remove a deployment directory

From the Deployment Directories tab, select a path from the directory list and click Remove.

The path is removed from the directory list.

Resource resolution

In the Launcher Administration, you can choose a resource configuration file (**.mrc**) that contains specifications for the Launcher. For example, the file can contain information such as an active virtual server and its associated resource name file (**.mrn**).

For more information about resource configuration files, see the *Resource Registry* documentation.

Cluster support

The Launcher service uses the Java™ Remote Method Invocation (RMI) to communicate with clients. When the Launcher service is running in a cluster environment or if there are machines with multiple network interface cards (NIC), you must manually specify the RMI system properties. You can do so from the Launcher Administration application Advanced tab.

The Launcher service sets these properties before running the RMI server. In a cluster environment, you must enter `java.rmi.server.hostname` in the Property column and the Launcher service host name in the Value column.

For example, `java.rmi.server.hostname` can be the Property. The host name **mycluster.fl.company.com** (under Value) can be the "value" of the property, which indicates where the Launcher service is running.

Firewall configuration

To use a firewall, settings in both the Launcher Administration and Management Console must be configured. Configuration for assigning port numbers for the firewall must be enabled for both.

By default, the Launcher opens anonymous ports for 2-way communication with the Management Console. If there is a firewall in place, fixed port numbers are required. To assign specific port numbers for the Launcher, you must enable the **Enable Configuration** option and either accept the default port numbers or enter specific ones.

To assign fixed port numbers in the Launcher Administration for firewall use

1. From the Launcher Administration Options tab, select the **Firewall** option.
Firewall configuration options appear.
2. Enable the **Enable Configuration** option.
The **Server Port** and **Client Port** fields become active.
3. Enter specific port numbers in the **Server Port** and **Client Port** fields, or accept the default values (**4634** and **4635**).

Firewall Option

Value

Enable Configuration

This option is turned off by default.

Select the check box to configure firewall ports.

When this option is enabled, the **Server Port** and **Client Port** fields become active.

Server Port

Enter a server port number or accept the default value of **4634**.

Client Port

Enter the client port number or accept the default value of **4635**.

4. Click OK to save the changes.

Note: If you are using a firewall, ensure that the Enable Configuration firewall option is also enabled in the Management Console.

To assign fixed port numbers in the Management Console for firewall use

1. From the Management Console, select Tools->Options.
The Options window opens.
2. From the **Options** list, select Firewall.
The override configuration options appear.
3. Enable the **Enable Configuration** option.
The **Client Port** field becomes active.
4. Enter the client port number in the **Client Port** field, or accept the default value (**7777**).

Firewall Option

Value

Enable Configuration

This option is turned off by default.

Select the check box to configure firewall ports.

When this option is enabled, the **Client Port** field becomes active.

Client Port

Enter the client port number or accept the default value of **7777**.

5. Click OK to save the changes.

HTTP Listener startup configuration

An HTTP Listener routes HTTP requests from external clients to maps in the Launcher and returns a response to the external client. Use the Launcher Administration utility Options page and select Listeners to add, remove, and update HTTP Listeners.

HTTP Listener settings

Name

A unique name of up to 25 characters to represent a listener.

Mode

When enabled, the listener starts automatically when the Launcher starts. By default, the listener is disabled.

HTTP port

The port that the listener uses to communicate with external clients. The listener monitors the HTTP port for incoming HTTP requests and responds to the external client through the HTTP port. Each listener must use a unique HTTP port. If you configure multiple listeners to use the same HTTP port, only one listener starts when the Launcher starts. The default HTTP port is 5017.

Launcher port

The port that the listener uses to communicate with the Launcher. Each listener must use a unique Launcher port. If you configure multiple listeners to use the same Launcher port, only one listener starts when the Launcher starts. The default Launcher port is 5018.

HTTP certificate

Specifies the server certificate used for SSL communication with external HTTP clients on the client port. If you omit it, SSL communication between the HTTP Listener and external HTTP clients is disabled.

Launcher certificate

Specifies the server certificate used for SSL communication with HTTP adapters on the Launcher port. If you omit it, SSL communication between the HTTP Listener and maps is disabled.

Log level

The trace setting for the HTTP Listener. The Launcher creates a unique trace file name by appending the current date, timestamp, and computer host name to the HTTP Listener name (for example, lsnr_a_socket07-06-16-02-11-14-PM_myhostname.log). Logs are created in the installation directory on Windows systems and in the \bin directory on UNIX systems.

None

Tracing is not enabled for the HTTP Listener. This is the default setting.

Standard

Logs the HTTP Listener activity. The log file name is *listenername_timestamp_hostname.log*.

Verbose

Logs the HTTP Listener activity and socket activity. The log file name is *listenername_socket_timestamp_hostname.log*.

Audit

Enables or disables an audit log of Transformation Extender map executions and each adapter connection and external client connection to the listener. The audit log is in XML format that can be queried with XSLT transformations. The audit log schema is LHL-Audit.xsd, installed in the Transformation Extender installation directory on Windows systems or the /bin directory on UNIX systems. Auditing is disabled by default. There is one audit log per listener.

Automatic restart for launchers and HTTP Listeners

Launchers and HTTP Listeners can automatically restart when they unexpectedly stop running. Use the Launcher Administration utility Options page to enable automatic restart and to limit the number and duration of restart attempts.

About multiple Launchers

For all supported Windows or UNIX platforms, multiple Launcher services or Launcher daemons that run on the same machine must run on different listening ports and have different port ranges. If this condition is violated, the Launcher services (or daemons) might not work as intended.

Run multiple Launcher services (or daemons) on separate installations of Launcher. Do not run more than one Launcher service (or daemon) on a single Launcher installation.

Preparing to execute defined systems

After you generate .msl files to the deployment directory, prepare the Launcher to execute the maps defined in those systems by performing one of the following actions:

- If the Launcher service or daemon is not currently started, start the Launcher.
- If the Launcher service or daemon is running, stop and then restart the Launcher.

To run a Launcher process (comprised of one or more systems) using the Launcher:

- The Launcher system files must be located in the designated deployment directory.
- All maps and source event gateways specified for system components defined within the system must be available.
- [Starting and stopping the Launcher](#)
- [Modifying systems while the Launcher is running](#)

Starting and stopping the Launcher

The following procedure describes the control functionality for running services on Windows platforms.

See the related reference topics for details about stopping the Launcher on a UNIX system.

Related reference

- [Starting, stopping, pausing, and resuming the Launcher service](#)

To start the Launcher service (Windows)

1. From your desktop, right-click My Computer and select Manage.
2. From the Computer Management window, expand the **Services and Applications** list and select Services.
3. In the **Computer Management** window, right-click **WebSphere® Transformation Extender Launcher** and select Properties. The Properties window opens.
4. Click Start.

To stop or restart the Launcher service (Windows)

From the **Computer Management** window, right-click **WebSphere Transformation Extender Launcher** and select **Stop** or **Restart** from the context menu.

Modifying systems while the Launcher is running

You can make changes to a Launcher system file (**.msl**) while the Launcher is running, however, you can only see the changes if the system is restarted from the Management Console. To add a new system, both the Launcher service or daemon and the Management Console need to be restarted.

The procedures in each of the following scenarios assume that a system is currently running in the Management Console.

Related reference

- [Benefits of using separate Launcher processes](#)

To modify an .msl file

If the changes are only to the contents of one particular **.msl** file that is already part of the **CompoundSystem** for a Launcher process (see ["Multiple Processes"](#) for additional information when using the multiple-processes feature), and the name of the system file is the same, then you must do the following actions:

1. Stop the **CompoundSystem** from the Management Console.
2. Overwrite the existing file in the deployment directory with the updated **.msl** file.
3. Restart the system from the Management Console.

To add or delete files

1. Stop the **CompoundSystem** from the Management Console.
2. Disconnect the Management Console from the relevant Launcher.
3. Stop the Launcher service.
4. After the service has stopped, make additions or deletions as necessary.
5. Restart the Launcher service.
6. In the Management Console, reconnect to the Launcher and restart the **CompoundSystem**.

Configuration parameters

The **config.yaml** file is the configuration file for **WebSphere® Transformation Extender** and contains specific settings for the Launcher. The **config.yaml** file is created in the <data directory>/config directory during the installation of **WebSphere Transformation Extender**. You can use a text editor to edit option values as needed.

- [HTTP Listener settings in config.yaml](#)
The HTTP Listener can automatically recognize an MTOM message and store the attachments in a user-designated location. You configure HTTP Listener settings in the config.yaml file.
- [Trace and log files naming conventions](#)
The Launcher trace and log files are located in the logs directory under *install_dir*, which is the directory where your product is installed.
- [Result codes in log files](#)

HTTP Listener settings in config.yaml

The HTTP Listener can automatically recognize an MTOM message and store the attachments in a user-designated location. You configure HTTP Listener settings in the config.yaml file.

The /runtime/HTTP Listener key of the config.yaml file is used by the Launcher HTTP Listener. You configure these settings as appropriate for your business needs.

RefreshInterval:

The time (in seconds) that the HTTP Listener Status interface automatically refreshes in a browser window (unless the page is already reloading from a prior refresh attempt). Set this parameter to 0 to disable the refresh interval.

RefreshIntervalIfHidden:

The time (in seconds) for automatic updates to occur if the sHTTP Listener Status interface is minimized or hidden behind another tabbed window.

mtomMode:

Configures the storage of MTOM attachments. If this parameter is enabled, the attachments for a MIME message that uses the XOP type are saved to the MTOM directory. For example, for this HTTP header

```
Content-Type: multipart/related; type="application/xop"
```

, any data following the `xop` is irrelevant. Set this parameter to 0 to disable it. Set this parameter to 1 to enable the storage of MTOM attachments.

mtomDir:

This is the directory in which all extracted attachments are stored. The default directory is the WebSphere® Transformation Extender logs subdirectory.

Important: Directory delimiters are substituted appropriately, depending on runtime and platform. On the UNIX platform, the drive letter and subsequent colon (for example, C:) are removed.

mtomExt:

This is the extension used for the MTOM file. The default file extension is MTOM. The file name format is: <HTTP Listener Cfg Name>-<date-time-stamp>-<session id>. <mtom_ext>

- [Enabling or disabling settings](#)
- [M4File](#)
- [Launcher](#)
- [Resource Manager](#)
- [Connection Manager](#)
- [Configuring Java-based adapters](#)

Enabling or disabling settings

If the config.yaml setting is commented out, or set to empty value, the default value is used. The default value is the value set in the original config.yaml file.

In the original config.yaml file, /runtime/Connections Manager/keep: 0 means that the default value of /runtime/Connections Manager/keep is 0. Therefore, the following three lines setting keep option are equal:

```
/runtime
  Connections Manager:
    keep: 0
    # keep: 0
    keep:
```

M4File

In the config.yaml file, the settings listed under the **runtime/m4file** key heading pertain to the file listener in the Launcher.

- [Logging](#)
- [FileListenCount](#)
- [ListenSleepTime](#)

Logging

Use the **/runtime/launcher/log** keys to define the file listener logging. See "[Troubleshooting Tip](#)" for details.

FileListenCount

This setting is the number of files to process at a time for each directory (0 meaning all). The default value for this setting is 25 (**FileListenCount=25**). For trigger uniformity, decrease this value as more directory watches are added.

ListenSleepTime

This setting is the time in milliseconds between directory processing. The default value is 50 (**ListenSleepTime=50**). If the CPU level is high when Launcher is idle, you should increase this value.

Launcher

In the config.yaml file, the Launcher settings are listed under **/runtime/launcher** key.

- [Logging categories descriptions](#)

- [ResourceCfgFile](#)
- [LoadPDH](#)
- [MaxThreads](#)
- [TriggerTime](#)
- [Launcher logging](#)
- [InitPendingHigh](#)
- [InitPendingLow](#)
- [WatchMaxThreads](#)
- [DisableMaxThreads](#)
- [InitPendingIdleMS](#)
- [RunMapCacheMaxNum](#)
- [LogFilePermissions](#)

You can use the LogFilePermissions setting in the runtime/launcher section of the **config.yaml** file to change the file permissions of the logs that are generated by the Launcher logging option.

- [MapAdapterTraceOff](#)

Enable this setting to turn off all adapter tracing for all watches, overriding any trace option that is configured on the adapter command line. When this setting is disabled, adapter tracing is controlled by the adapter command line.

- [CircularLogSize](#)

Use this option to specify the maximum size of the *system_name_timestamp_host_name.log* file.

- [CircularLogFileNum](#)

Use this option to specify the maximum number of *system_name_timestamp_host_name.log* files to keep before the oldest, non-startup log file is deleted.

Logging categories descriptions

Logging Category	Value	Description
trace	false	Disabled (default)
debug	false	Disabled (default)
info	false	Disabled (default)
warning	true	Enabled
error	true	Enabled
fatal	true	Enabled

If you turn on logging (["Logging Option"](#)) from the Management Console or the **config.yaml** file, the resulting files are created in the **runtime/launcher/log** subdirectory of the **WebSphere® Transformation Extender** installation directory.

ResourceCfgFile

A default resource configuration file for the Launcher is specified in the /runtime/launcher/ResourceCfgFile key section of the **config.yaml** file:

```
/runtime:
  launcher:
    ResourceCfgFile: resource.mrc
```

You can specify a resource configuration file for the Launcher by modifying the default file name (**resource.mrc**) to the resource configuration file that you want to use.

For more information about the resource configuration file, see the *Resource Registry* documentation.

LoadPDH

This setting disables the central processing unit (CPU) and memory counters (Windows only).

If the setting is commented out (indicated by a semicolon at the beginning of the line), or uncommented and set to **1**, the CPU and memory counters are loaded.

If you uncomment the line and set the value to **0** by a hash sign #, the function is disabled. Setting LoadPDH to **0** is generally not necessary; however, some adapters might require the disabling of counters.

LoadPDH: {0 | 1}

Option

Description

0

Disables CPU and memory counter loading.

1

Enables CPU and memory counter loading.

MaxThreads

The **MaxThreads** value is used to limit the number of map threads for all watches in an environment, provided that no other settings have conflicting values. If the following settings are not set to 0, then the **MaxThreads** value will be impacted:

- **WatchMaxThreads**
- **DisableMaxThreads**
- **Max Concurrent Map Instances** (a setting in the IFD).

MaxThreads: *n*

Option

Description

n

Maximum number of maps (threads) that the Launcher can execute concurrently. Range is 0 to 32,000. Default value is 20.

Note: For performance reasons, on the z/OS® operating system, *n* should be less than or equal to 400.

TriggerTime

Use the **TriggerTime** setting to specify file time granularity in seconds.

TriggerTime: *n*

Option

Description

n

Minimum trigger time in seconds for files. This is the minimum time that must elapse in a file's timestamp before it can cause a new event. Default value is 1.

Note: The **TriggerTime** setting can be used for files that reside in the Hierarchical File System (HFS) on the z/OS® operating system; it is not available for use with native data sets.

Launcher logging

In the `/runtime/launcher` key of the **config.yaml** file, you can specify the settings for a Launcher log file. See [Trace and Log Files Naming Conventions](#) for the specific naming conventions.

The `/runtime/launcher/LauncherLog` is set to empty value by default and the launcher log file is not created by default. To make sure that a log file is created, you must set one of the available options in the **config.yaml** file.

For example:

LauncherLog: `ewsc`

This will create a file that contains error, warning, startup, and configuration information, and will be created in the product installation directory by default.

The default syntax is:

LauncherLog: `[E] [W] [S] [C]`

Each option is described in the following table.

Option

Description

`E`

Logs map errors

`W`

Logs map warnings

`S`

Logs Launcher startup statistics

`C`

Logs configuration information

Configuration option

If you specify `c` for configuration, the following example displays how each watch is configured:

```
*** CONFIGURATION:

Time: Thu Nov 21 10:53:05 2002
-----
Watch: 1
MSL File: C:\mapdev.nn\mercntsv\systems\single-mq-to-file.msl
System: single-mq-to-file
Component: fromqueuetofile
Map: C:\testing\n.n\15010\fromqueuetofile.mmc

Map Delay: <none>
Pending Exp.: <none>
```

```

Retries:          10 every 5 seconds
Paging:          8 X 64K
Trace:           Summary
Validation:      Ignore: <none>
                  Stop On First Error: Yes
Work Area:       Default
Init High:       100
Init Low:        50
Time Trigger:    <none>
Input 1 (T):     IBM WebSphereMQ: -QMN QM1 -QN QA.QUEUE5
-MID * -QTY 1
                  Rollback: Yes, Delete: No, Reuse Work File: No
                  Retries: <none>
Output 1:         File: C:\testing\n.n\15010\output*.txt
                  Rollback: Yes, Delete: If empty, Append: No
                  Retries: <none>
Audit File:      Burst Data           Never
                  Burst Execution      Always
                  Summary Execution     Always
                  Data Settings         Never
                  Map Settings          Never

```

Startup option

If you specify **s** for startup, you will see an entry on whether the startup of the Launcher was successful. If successful, something similar to the following entry is displayed:

```
*** STARTUP:
Status:          Successful
Time:            Thu Nov 21 10:53:11 2002
```

If not successful, it might pinpoint where the startup problem is:

```
*** STARTUP:
Status:          Failure
Time:            Thu Nov 21 18:49:58 2002
Errors:          No systems to run
```

Error option

If you specify **e**, the file lists all maps that have an error status after being run:

```
*** ERROR:
System:          single-mq-to-file
Component:       fromqueueToFile
Map:             C:\testing\n.n\15010\fromqueueToFile.mmc
Start Time:      Thu Nov 21 10:54:06 2002
Instance:        1
Run Time:        0.1957 seconds
Return Code:     8 - One or more inputs was invalid
Input 1:         IBM WebSphereMQ: -QMN QM1 -QN QA.QUEUE5
-MID AMQ0 -QTY 1
Output 1:        File: C:\testing\n.n\15010\outputAMQ.txt
Audit File:      C:\testing\n.n\15010\fileAMQ.log
```

Warning option

If you specify **w**, the file lists all maps that have a warning status after being run.

```
System:          single-mq-to-file
Component:       fromqueueToFile
Map:             C:\testing\n.n\15010\fromqueueToFile.mmc
Start Time:      Thu Nov 21 11:23:54 2002
Instance:        1
Run Time:        0.0923 seconds
Return Code:     21 - Input valid but unknown data found
Input 1:         IBM WebSphereMQ: -QMN QM1 -QN QA.QUEUE5
-MID AMQ1 -QTY 1
Output 1:        File: C:\testing\n.n\15010\outputAMQ.txt
Audit File:      C:\testing\n.n\15010\fileAMQ.log
```

InitPendingHigh

Use `InitPendingHigh` to pause the listener threads if too many initialized pendings occur.

`InitPendingHigh: n`

Option

Description

`n`

Any integer value; modification not recommended.

Default setting of `0` specifies that an unlimited number of `InitPendings` might occur. For example, if you change this value to `10`, the listener thread pauses when the number of `InitPendings` reaches the limit of `10`.

InitPendingLow

Use `InitPendingLow` to resume the listener threads if paused from `InitPendingHigh`.

`InitPendingLow: n`

Option

Description

`n`

Any integer value; modification not recommended.

Default setting is `0`. The listener thread resumes when the number of occurring `InitPendings` is equal to this setting. For example, if `InitPendingHigh` is set to `10` and `InitPendingLow` is set to `5`, the listener thread pauses processing when the number of occurring `InitPendings` reaches `10` and resumes processing when the number reaches `5`.

WatchMaxThreads

The `WatchMaxThreads` option is also known as Global Max Threads per Watch.

`WatchMaxThreads: n`

Option

Description

`n`

Global limit on the number of concurrent mappings for each (*any individual*) watch.

This number cannot exceed the `MaxThreads` value, which is the total number of concurrent mappings overall.

Range is `0` to `32,000`.

Default setting is `0`.

Note: The `Max Concurrent Map Instances` setting in the IFD overrides the `MaxThreads`, `WatchMaxThreads`, and `DisableMaxThreads` Launcher settings.

DisableMaxThreads

Use the `DisableMaxThreads` setting to disable the `MaxThreads` option.

`DisableMaxThreads: {0|1}`

Option

Description

`0`

Setting is off (default)

`1`

Setting is on.

If set to `1`, the `MaxThreads` value is disabled and the `WatchMaxThreads` value is used to place a global limit on the number of concurrent map executions for each watch.

Note: The `Max Concurrent Map Instances` setting in the IFD overrides the `MaxThreads`, `WatchMaxThreads`, and `DisableMaxThreads` Launcher settings.

InitPendingIdleMS

This entry controls how often the Launcher checks to see if initialization-pending maps can be executed. This reduces waiting time, and improves performance on most platforms.

The default value is **1**. Higher values cause the Launcher to check less frequently for initialization-pending maps, which can decrease CPU usage but might introduce latency to the map execution.

InitPendingIdleMS: *n*

Option

Description

n

Time (in milliseconds) that initiation-pending threads should remain idle between mapping attempts.
Valid values are 1-10000 (one millisecond - 10 seconds).

RunMapCacheMaxNum

Use the `RunMapCacheMaxNum` setting to define the maximum number of RUN maps to be cached in memory.

To disable RUN map caching, set the value to **-1**.

RunMapCacheMaxNum: *n*

Option

Description

n

Maximum number of RUN maps to be cached in memory.
Default value is 300; the minimum value is 10.

LogFilePermissions

You can use the `LogFilePermissions` setting in the runtime/launcher section of the `config.yaml` file to change the file permissions of the logs that are generated by the Launcher logging option.

LogFilePermissions: *value*

Where *value* is one of the following:

Value	File Permissions
666	<code>rw-rw-rw-</code> This is the default file permission.
664	<code>rw-rw-r--</code>
644	<code>rw-r--r--</code>

MapAdapterTraceOff

Enable this setting to turn off all adapter tracing for all watches, overriding any trace option that is configured on the adapter command line. When this setting is disabled, adapter tracing is controlled by the adapter command line.

MapAdapterTraceOff: {0 | 1}

Option

Description

0

Disables the `MaxAdapterTraceOff` setting. This is the equivalent of commenting out the option with a preceding semicolon (;). When this option is disabled, the adapter command line controls tracing.

1

Turns off all adapter tracing for all watches, overriding any trace option that is configured on the adapter command line.

Related reference

- [Dynamic adapter tracing](#)

CircularLogSize

Use this option to specify the maximum size of the `system_name_timestamp_host_name.log` file.

When the log file reaches the maximum size, the Launcher service closes it, appends an index number to the timestamp in the log file name (`LauncherLog_timestamp_index.log`), and begins logging to a new file.

CircularLogSize: *n*

Option

Description

n

The maximum log file size. The maximum size is 100 MB. A value of 0 specifies an unlimited log file size. The default minimum size is 10 MB. If you specify a smaller size than 10 MB, the Launcher service creates a 10 MB log file.

Related reference

- [Launcher log file](#)
-

CircularLogFileNum

Use this option to specify the maximum number of *system_name_timestamp_host_name.log* files to keep before the oldest, non-startup log file is deleted.

CircularLogFileNum: *n*

Option	Description
<i>n</i>	Specifies the maximum number of non-startup log files to keep before deleting the oldest. The minimum is two log files, and the default maximum is five log files.

Related reference

- [Launcher log file](#)
-

Resource Manager

The config.yaml file contains the default Launcher startup setting values for the Resource Manager. The Resource Manager synchronizes shared data and timing interfaces among heterogeneous sources and targets. You can edit the values as needed using a text editor.

- [Logging categories](#)
 - [Disable](#)
 - [TransManager](#)
 - [Stream configuration options](#)
-

Logging categories

The logging categories provide a variety of debugging information for the Resource Manager. The Launcher uses them to determine which severity levels and components to debug.

/runtime/Resource Manager/log

For descriptions and examples of the logging categories, see [Logging Category Descriptions](#).

Disable

Use this option to enable or disable resource management. If the Resource Manager is disabled, the file access protection scheme is deactivated. This allows you to have concurrent maps writing to the same file at the same time. If you choose to disable the Resource Manager and your system relies on the file access protection, data can become corrupt. However, if you do not need resource management, disabling this option boosts performance.

Disable: *0*

Option	Description
0	Enables resource management. This is the default setting.
1	Disables resource management.

TransManager

See the *Global Transaction Management* documentation.

Stream configuration options

When the data that is passed to or from an adapter exceeds the size specified by Stream Maximum Memory Limit value, a file is created in a temporary directory for paged data. This limits the memory consumption of the process.

```
/runtime/stream/MaxMemLimit: n
```

Option

Description

n

Data size in megabytes.

If the size of data is exceeded, the data is paged to a temporary file.

Default value is 500.

```
/runtime/stream/MaxMemDirectory: /tmp
```

By default, the temporary directory is designated by the operating system.

For Windows this is typically what the environment variable **TEMP** is set to (such as **C:\TEMP**); for UNIX, this is typically **/tmp**.

Note: Ensure that the path points to a directory with sufficient file space.

Connection Manager

The **config.yaml** file contains the default product startup setting values for the Connection Manager. You can use a text editor to edit the values as needed. A value of 0 means disabled or off. A setting that is commented out is disabled.

- [How the Connection Manager works](#)
- [Connection Manager settings](#)
- [Logging categories](#)

How the Connection Manager works

The Connection Manager works in conjunction with adapters to maintain connections. To reduce overall run time and increase efficiency, the Connection Manager keeps a pool of connections to reduce the total number of connect or disconnect operations that have to be done. This means that a connection will not be disconnected when a map is complete, but sometime after that.

When a connection is needed, the Connection Manager searches its pool until it finds a connection that the adapter finds suitable (using the adapter's **CompareConnections** function). A suitable connection means that it must be connected to the appropriate resource, such as a database, message queue, and so forth.

If a suitable connection is found, Connection Manager then tries to verify that the connection is still valid. This is done by calling **ValidateConnection**. If the adapter does not implement **ValidateConnection**, the test will fail.

Connection Manager settings

The Connection Manager settings include:

- [Logging Categories](#)
- [IgnoreGPFs](#)
- [ShareConnectionThreads](#)
- [Idle](#)
- [Soft Limit \(SLim\)](#)
- [Hard Limit \(HLim\)](#)
- [Min](#)
- [Keep](#)
- [HSleep](#)
- [HLimTimeout](#)
- [PollWaitTimeMin](#)
- [PollWaitTimeMax](#)

Each setting is described here.

Logging categories

The logging categories defined in **config.yaml** using **/runtime/Connection Manager/log** provide a variety of debugging information for the Connection Manager. The runtime components uses them to determine which severity levels and components to debug.

For descriptions and examples of the logging categories, see [Logging Category Descriptions](#).

IgnoreGPFs

Use this setting to ignore system exceptions during processing so that only the map fails as opposed to the system.

IgnoreGPFs: false

Option

Description

false

Disables the option, which means the system will fail upon encountering an exception.

true

Enables the option so that exceptions are ignored on connection threads and therefore a map failure occurs instead of a system failure.

This is the default setting.

ShareConnectionThreads

The **ShareConnectionThreads** setting allows multiple connections on the same thread. There are three levels of sharing to specify the criteria used for sharing threads. Each value is described in the following table. This setting is disabled by default.

ShareConnectionThreads: n

Option

Description

0

Disables the sharing option, which means each connection gets its own thread.

1

Enables sharing for GTX only. Only connections with GTX will get combined to the same thread.

idle

Use this setting to specify global default settings for all adapters or specify settings for individual adapters.

The idle setting creates a background map thread with the purpose of periodically searching the connection pools, locating connections with idle time greater than the value specified by idle, and then removing them. This allows connections to be disconnected even when no map is running.

The Connection Manager shuts down the idle thread gracefully when running under a Launcher or Command Server. In an API environment that invokes maps that use Java-based adapters (such as JMS, Java™ Class, and JALE), you might wait additional idle seconds after a shutdown request. This delay occurs only when the custom application does not explicitly shut down WebSphere® Transformation Extender by using the Java-based terminateAPI routine or the C-based mpiTermAPI routine.

- To specify default settings for all adapters, use `idle: n`
- To specify settings for individual adapters, use `/runtime/Connections Manager/xxx/idle`

Option

Description

0

Disables the use of idle settings for adapters.

n

Number of seconds to allow a connection to be idle before automatically disconnecting.

xxx

Adapter type where xxx represents the command line alias for the adapter. Example values for typical adapters include:

DB

Database Adapters

MQS

IBM® WebSphere MQ (server) Adapters

MQSC

IBM WebSphere MQ (client) Adapters

ALE

ALE Adapters

FTP

FTP Adapters

For example, `/runtime/Connections Manager/MQS/idle: 30` sets the connection idle limit for IBM WebSphere MQ adapters to 30 seconds.

See the Resource Adapters documentation for more information.

Soft Limit (sLim)

Most servers place a limit on the number of clients that can be connected. WebSphere Transformation Extender generally shares a server with other clients, and therefore should not be able to use all available connections. The Resource Manager provides two limit parameters: sLim (soft limit) and hLim (hard limit). Either or both of these parameters can be used to specify a maximum number of connections. Existing connections are only removed when a new one is required.

The sLim setting is a soft (advisory) limit. This number can be exceeded at peak load, but sLim tries to limit the number of idle connections at all other times. Limits are only checked when opening a connection, so it is possible to have a greater number of idle connections.

The global sLim value is by default set to **4** and is used if any other limit for the number of connections is not set.

- To specify global default settings for all adapters, use `sLim: n`
- To specify settings for individual adapters, use `/runtime/Connections Manager/xxx/sLim`

Option

Description

n

	Number of idle connections. Default is 4.
xxx	Adapter type where xxx represents the command line alias for the adapter. Example values for typical adapters include:
DB	Database Adapters
MQS	IBM WebSphere MQ (server) Adapters
MQSC	IBM WebSphere MQ (client) Adapters
ALE	ALE Adapters
FTP	FTP Adapters

For example, `/runtime/Connections Manager/MQS/slIm: 30` sets the soft limit for IBM WebSphere MQ adapters to 30 idle connections.

See the *Resource Adapters Library* for adapter documentation.

Hard Limit (hLim)

The hLim setting is a hard (mandatory) limit. This number will not be exceeded. Connections are placed in a pending state rather than exceed this number.

- To specify global default settings for all adapters, use: `hLim: n`
- To specify settings for individual adapters, use: `/runtime/Connections Manager/xxx/hLim`

Option	Description
n	Maximum number of connections. Default is 0 (unlimited).
xxx	Adapter type where xxx represents the command line alias for the adapter. Example values for typical adapters include:
DB	Database Adapters
MQS	IBM WebSphere MQ (server) Adapters
MQSC	IBM WebSphere MQ (client) Adapters
ALE	ALE Adapters
FTP	FTP Adapters

For example, `/runtime/Connections Manager/MQS/hLim: 30` sets the hard limit for IBM WebSphere MQ adapters to 30 connections.

min

The min setting specifies a minimum number of connections to keep. When a limit (slim or hlim) is exceeded, idle connections will be removed until no more than this value remains (if possible).

- To specify default settings for all adapters, use `min: n`
- To specify settings for individual adapters, use `/runtime/Connections Manager/xxx/min`

Option	Description
n	Minimum number of connections to keep. The default setting is 0, which means there is no limit set.
xxx	Adapter type where xxx represents the command line alias for the adapter. Example values for typical adapters include:
DB	Database Adapters
MQS	IBM WebSphere MQ (server) Adapters
MQSC	IBM WebSphere MQ (client) Adapters
ALE	ALE Adapters
FTP	FTP Adapters

keep

Use this setting to specify the time, in seconds, an idle connection is expected to remain valid. Connections that idle longer than this are tested prior to being used. If the test fails or is not supported, the connection is not reused.

The keep setting is an efficiency measure and represents some amount of time to assume that the connection is valid without a test. If the idle time of the connection (amount of time since it was last used) is less than the keep value, then the Resource Manager does not attempt to verify connection validity.

Last used refers to the timestamp when the connection was last assigned to a map. Whether it was used successfully or not is not considered because it is assumed that an unsuccessful attempt is a temporary condition, not a permanent failure.

This setting should only be used for adapters that implement ValidateConnection.

Do **not** use the KEEP setting with the FTP adapter. When used with the FTP adapter, the KEEP setting causes unpredictable results.

- To specify global default settings for all adapters, use: `keep: n`
- To specify settings for individual adapters, use: `/runtime/Connections Manager/xxx/keep`

Option

Description

`n`

Specify how many seconds an idle connection is expected to remain valid. Default is **0**, which disables connection testing. A value of **-1** will force a test every time, but this is not recommended.

`xxx`

Adapter type where `xxx` represents the command line alias for the adapter. Example values for typical adapters include:

`DB`

Database Adapters

`MQS`

IBM WebSphere MQ (server) Adapters

`MQSC`

IBM WebSphere MQ (client) Adapters

`ALE`

ALE Adapters

`FTP`

FTP Adapters

For example, `/runtime/Connections Manager/MQS/keep: 30` sets the keep period for IBM WebSphere MQ adapters to wait 30 seconds for idle connections.

Note: As an example, if you are executing maps using an IBM WebSphere MQ (server) adapter at least once every 10 seconds, a setting of `/runtime/Connections Manager/MQS/keep: 10` results in the connection not being tested because they are used before they expire.

See the *Resource Adapters Library* for adapter documentation.

HSleep

The HSleep setting is the specified polling interval used to check resource availability. The value is the cycle time (in seconds) for checking connection availability. The default value is **2** and if specified should be as follows: $0 < \text{HSleep} < \text{HLimTimeout}$.

Note: No validation of this value is performed. Unexpected values for this setting can cause unpredictable results.

HSleep: n

Option

Description

`n`

Number of cycle time seconds for checking connection availability.

Default value is **2** seconds.

Note: The number of retries is the HLimTimeout value divided by the HSleep value.

HLimTimeout

The HLimTimeout setting specifies a maximum amount of time (in seconds) that a map waits for a new connection to become available before failing the map. If not specified, HLimTimeout defaults to infinite.

HLimTimeout: n

Option

Description

`n`

Number of seconds that a map waits for a new connection to become available before failing the map.

Default value is infinite.

PollWaitTimeMin

The PollWaitTimeMax and PollWaitTimeMin parameters enable you to configure a polling interval that is used by adapters to detect events. If the interval is not specified, the default value is used.

Note: If this value is too small, such as below 50 milliseconds, CPU usage can go up during the idle time when the Launcher is waiting for an event.

PollWaitTimeMin: n

Option

Description

`n`

Minimal polling interval in milliseconds.

Default setting is 2000.

PollWaitTimeMax

The PollWaitTimeMax and PollWaitTimeMin parameters enable you to configure a polling interval that is used by adapters to detect events. If the interval is not specified, the default value is used.

PollWaitTimeMax: *n*

Option	Description
<i>n</i>	Maximal polling interval in milliseconds. Default setting is 10000.

Configuring Java-based adapters

From the **config.yaml** file, you can add **.jar** files to the CLASSPATH variable. To do this, you must add an **/runtime/External Jar Files/jarN** key to **config.yaml** and list the applicable jar files, similar to the following example:

```
/runtime/External Jar Files:  
  jar1: c:\J2EE\lib\j2ee.jar  
  jar2: c:\mypath\myjar.jar
```

For more information, see the *Resource Adapters* documentation.

Trace and log files naming conventions

The Launcher trace and log files are located in the logs directory under *install_dir*, which is the directory where your product is installed.

The naming convention for the Launcher trace files, previously named launcher.txt, is the following:

Trace File Naming Convention	Description
CompoundSystem_date_time_hostname.txt	For all systems in a single Launcher process
<i>unique-name</i> _date_time_hostname.txt	For one or more systems run in separate Launcher processes. <unique_name> is the name configured in the Launcher Administration.

The naming convention for the Launcher log files, previously named mercntsv.log on Windows platforms and debug.log on UNIX platforms, is shown in the following table:

Log File Naming Convention	Description
CompoundSystem<date><time><hostname>.log	For all systems in a single Launcher process
<unique_name><date><time><hostname>.log	For one or more systems run in separate Launcher processes. <unique_name> is the name configured in the Launcher Administration.

The name and location of these log files within the initialization file are not used. The content of the log files from the resource manager and connections manager, previously named dtxrsmgr.log, are written to the Launcher log for each process.

Result codes in log files

The Launcher log files contain result codes from processes run by the Launcher. These codes are returned from the operating system and do not have the same meaning as the codes returned after issuing command line options.

Launcher settings

This documentation describes the settings that are configured in the IFD for maps to be run by the Launcher. These map settings, when configured from the IFD, are called *Launcher settings*.

Map settings are originally set during map creation in the Map Designer. When you change the map settings using the IFD, you are overriding the map settings. However, the overrides from the IFD only affect the map during the generation of Launcher system files (**.msl**)-the settings in the original map source file are not affected.

See the *Integration Flow Designer* documentation for information about how to configure all settings.

Note: The UNIX operating systems are case sensitive. Systems created using the IFD to be executed by a UNIX Launcher must observe the proper case for path names, map names, and so on.

- [Accessing Launcher settings](#)
- [Launcher settings window](#)

- [Trigger event settings](#)
 - [Source and target settings](#)
 - [Adapter classes](#)
 - [Adapters that support event triggers](#)
 - [Connection testing adapters](#)
 - [Subsystem component execution settings](#)
-

Accessing Launcher settings

Systems defined in the IFD contain source map components, compiled map components, and Launcher settings configured to control map execution. Systems can include options such as time and source event triggers to initiate maps and source and target settings for inputs and outputs.

The following procedure assumes that execution mode for the open system is **Launcher**.

To view Launcher settings:

From the IFD, do one of the following actions to view Launcher settings for a map component:

- Right-click on a map component and select Edit Launcher Settings.
 - Click the execution settings button of the map component icon.
- The Launcher Settings window appears.
-

Launcher settings window

The Launcher Settings window shows a collapsed tree view of the settings that can be specified for a map component in a system that has the Launcher mode.

The settings provide specific information needed by the Launcher.

- [MapServerLocation](#)
Use the Launcher settings in the Integration Flow Designer to set the MapServerLocation. The MapServerLocation setting identifies the path to the executable map on the server. The default value is the path of the selected map.
- [MapSettings > MapAudit](#)
- [MapSettings > WorkSpace](#)
- [MapDelay](#)
- [PendingExpiration](#)
- [Max concurrent map instances](#)
- [TimeEvent](#)
- [Pending instance thresholds](#)
- [MapInstance Timeout](#)
The MapInstance Timeout setting is the length of time that a map instance can run before a warning or an error is logged in the compound system log. You can specify the time limit in hours, minutes, or seconds.

MapServerLocation

Use the Launcher settings in the Integration Flow Designer to set the MapServerLocation. The MapServerLocation setting identifies the path to the executable map on the server. The default value is the path of the selected map.

When using the z/OS® Launcher, see "[z/OS Launcher](#)" for more information about specifying the location of the compiled map file as a native file system object.

Choose one of the following methods to change the default MapServerLocation setting:

- Enter the path and file name of the compiled map.
 - Click the browse button and navigate to the compiled map file.
 - Enter a resource alias that represents the path and file name of the compiled map. Enclose the alias in percent symbols (for example, %map_loc%). In the Integration Flow Designer, use Tools > Options > Miscellaneous to set the resource configuration file. See the Resource Registry documentation for details about resource aliases.
-

MapSettings > MapAudit

You can use the **MapAudit** settings to create and store map audit logs. The audit logs contain information about data audit, execution audit, map settings, and data settings.

MapAudit settings are compiled (as overrides) into the **.msl** file, which is run by the Launcher.

To enable the audit log

From the Launcher Settings window, under MapSettings > MapAudit > Switch, choose **ON**.

The audit log is created in a file in the map directory or to a location that you specify using the **AuditLocation** settings.

When you choose **Custom** for **Directory** under the **AuditLocation** setting, enter a file path in **Value**. The value you enter in **FileName** determines the audit log file name.

There are three options from which to choose for the audit log filename (AuditLocation.>.FileName): **Default**, **Unique**, and **Custom**. The following table displays the filename outcome of each option.

Value

Description

Default

The file is automatically assigned an audit log file name (based on the compiled map name) with a .log file name extension.

Unique

Generates a unique audit log name in the following format:

Mer_mapname_processkey_mapcounter_hostname.log

Run_mapname_processkey_mapcounter_hostname.log

The prefix "Mer" is used with top-level maps while "Run" is used for run maps.

mapname

The executable map name.

processkey

Unique value for each process.

mapcounter

Unique value for each map instance (within the same process)

hostname

The host name of the computer where Transformation Extender is running.

Custom

A custom filename. Enter the name in the FileName Value field.

Note: Wildcards can be used as long as an input trigger resolves them.

Note: For detailed information about map settings, see the *Map Designer* documentation.

Note: If you are using a map stored in a z/OS® dataset, see the documentation about the three data file location options DEFAULT (["DEFAULT"](#)), CUSTOM (["CUSTOM"](#)), and UNIQUE (["UNIQUE"](#)), for the format of the audit log filename generated in the HFS.

- [Unique names for audit log files](#)

Unique names for audit log files

If the log files produced for one map trigger another map, use AuditLocation.>.FileName.>Unique to select unique names for the log files. Each log file will then trigger another instance of the other map.

For the AuditLocation.>.FileName setting, the following option results apply:

- The **Unique** option produces a file in the following format:
Mer_mapname_processKey_mapCounter_hostname.log
- The **MapName** option produces a log file in the following format:
mapname.log

If you use this **MapName** option, the general **Retry** option is used to determine when one map instance will not interfere with another. If you use multiple maps that generate unique log file names and you want to use log files to trigger different maps, be sure to specify different directories for those log files.

MapSettings > WorkSpace

When a map runs, the software creates or reuses a workspace that it uses during the mapping process to find data that it needs to transform.

The **Unique** value of the **WorkFilePrefix** setting enables you to select globally-unique work file names for a map. If you want multiple instances of the same map to run at the same time, use this option. If you do not use the unique work files option, multiple instances of the same map will run sequentially.

If you do not use unique work files, the general retry setting is used (MapSettings.>.MapRetry.>.Switch = ON) when one map instance interferes with another.

Note: Selecting the **Memory** option for the **AuditLocation** setting also allows multiple instances of the same map executing simultaneously.

Unique filename format for work files:

Mer_mapname_processkey_mapcounter_hostname[I|O].cardnumber

MapDelay

The **MapDelay** setting defines the amount of time in milliseconds that the Launcher waits, after all triggers are satisfied, before running the map. The default value for the **MapDelay** setting is **0**.

Value

Description

n

Time, in milliseconds, to wait before starting the map.

Valid entries are integers 0 to 9999.

PendingExpiration

The **PendingExpiration** setting specifies the amount of time that a map can stay in an initiation pending state before expiring (running without all of its input events). The default value for the **PendingExpiration** setting is **Hour**.

Setting Value

Description

Hour

Measures the value of the **PendingExpiration** period in hours

Minute

Measures the value of the **PendingExpiration** period in minutes

Second

Measures the value of the **PendingExpiration** period in seconds

The **PendingExpiration_Value** setting specifies how many units (in hours, minutes, or seconds) to use for measuring the waiting period. Enter the number of units in the **Value** field. The default value for the **Value** setting is **0**.

If the map associated with a given watch has multiple triggers and at least one trigger occurs, the map is placed in an *initiation pending* state until all triggers occur. Maps in an initiation pending state are reevaluated each time another trigger occurs.

The initiation pending state occurs when any of the following conditions are true:

- The map is defined as single-threaded and one occurrence of the map is already running.
- The map requires an event that has not yet occurred. (This is the only initiation pending condition in which **PendingExpiration** can occur.)
- The maximum number of concurrent maps are already running.

If all preceding conditions are eliminated, the initiation pending state can be removed from the map; if the **PendingExpiration** period has not elapsed, the map can run.

Max concurrent map instances

Specifies the maximum number of concurrent mappings for each watch. In the IFD, this setting overrides the **MaxThreads**, **WatchMaxThreads**, and **DisableMaxThreads** Launcher settings.

TimeEvent

The **TimeEvent** settings initiate (or trigger) map components of a system based on time. Each map component is run based on a time event, a source event, or a combination of multiple events. If a map component is not assigned an event trigger, the Launcher will not initiate that map to run.

Note: When you analyze a system on the IFD and your system is set to Launcher mode, a warning is registered for any source or compiled map component that has no associated event triggers.

- [TimeEvent > Switch](#)
- [TimeEvent > BeginAfter](#)
- [TimeEvent > Trigger](#)
- [Trigger > Interval](#)
- [Interval > OmitDays](#)
- [Interval > OmitHours](#)
- [TimeEvent > SkipIfBusy](#)
- [TimeEvent > EventCoordination](#)

TimeEvent > Switch

The **Switch** setting enables or disables **TimeEvent** options. The default value for the **Switch** setting is **OFF**.

Value

Description

OFF

Disables **TimeEvent** options

ON

Enables **TimeEvent** options

TimeEvent > BeginAfter

The **BeginAfter** settings specify the period of time in hours, minutes, or seconds to wait (from the time the Launcher starts) before enabling the time trigger.

BeginAfter > Hour

The **Hour** setting is used to specify how many hours to wait to begin the map. The default value for the **Hour** setting is **0**.

Value	Description
<i>n</i>	Enter a number to represent the time in hours to wait to begin the map. Valid entries are integers 0 to 23.

BeginAfter > Minute

The **Minute** setting is used to specify how many minutes to wait to begin the map. The default value for the **Minute** setting is **0**.

Value	Description
<i>n</i>	Enter a number to represent the time in minutes to wait to begin the map. Valid entries are integers 0 to 59.

BeginAfter > Second

The **Second** setting is used to specify how many seconds to wait to begin the map. The default value for the **Second** setting is **0**.

Value	Description
<i>n</i>	Enter a number to represent the time in seconds to wait to begin the map. Valid entries are integers 0 to 59.

TimeEvent > Trigger

The **Trigger** setting is used to define the type of time event. You can trigger a map based on a singular time event or cycle, according to a specified time increment. The default value for the **Trigger** setting is **Once**.

Value	Description
Once	The trigger is a non-repeating, one-time-only event.
Every	The trigger occurs once for every interval defined using the Interval settings.

Trigger > Interval

Use the **Interval** setting to describe the recurring period to use for the trigger. The **Interval** setting is available only if the value of the **Trigger** setting is **Every**. The default value for the **Interval** setting is **1 Day(s)**.

Interval > At

Use the **At** settings to set a specific time for the trigger to start after the start of the interval defined in **Interval** settings. The **At** settings are only available if the value of the **Trigger** setting is **Every**. The time periods specified in the **Hour**, **Minute**, and **Second** settings describe a time based on a 24-hour clock set to system time.

At > Hour

The **Hour** setting specifies the hour in which to run the map. The default value for the **Hour** setting is **0**.

Value	Description
<i>n</i>	Enter a number representing the time in hours in which to run the map. Valid entries are integers 0 to 23.

At > Minute

The **Minute** setting specifies the minute in which to run the map. The default value for the **Minute** setting is **0**.

Value	Description
<i>n</i>	Enter a number representing the time in minutes in which to run the map. Valid entries are integers 0 to 59.

At > Second

The **Second** setting specifies the second in which to run the map. The default value for the **Second** setting is **0**.

Value	Description
<i>n</i>	Enter a number representing the time in seconds in which to run the map. Valid entries are integers 0 to 59.

Interval > OmitDays

Use the **OmitDays** setting to set the days to omit.

OmitDays > Monday

The **Monday** setting is used to omit Mondays from the time trigger schedule. The default value for the **Monday** setting is **No**.

Value

Description

No

Do not omit Mondays.

Yes

Omit Mondays.

OmitDays > Tuesday through Sunday

These settings work the same way as the **Monday** setting.

Interval > OmitHours

Use the **OmitHours** setting to set the hours to omit.

OmitHours > 0

The **0** setting is used to omit the 0 hour (midnight to 1:00 am). The default value for the **0** setting is **No**.

Value

Description

No

Do not omit this hour.

Yes

Omit this hour.

OmitHours > 1 through 23

These settings work the same way as the **0** setting (above).

TimeEvent > SkipIfBusy

The **SkipIfBusy** setting is used to ignore the time trigger if it occurs when a watch instance is already in process. The default value for the **SkipIfBusy** setting is **Yes**.

Value

Description

Yes

Ignore time trigger if a watch is in process.

No

Do not ignore time trigger if a watch is in process; initiate another watch instance.

TimeEvent > EventCoordination

The **EventCoordination** setting is used to control the coordination of time and input events. The **EventCoordination** setting is available only if the map has both a time trigger and one or more input triggers: the **SourceEvent** setting for one of the input cards is **ON**. The source trigger might have wildcard values. The default value for the **EventCoordination** setting is **All or None**.

Value

Description

All or None

If no source trigger exists when the time trigger occurs: this time trigger interval is skipped and nothing is run.

If one source trigger exists when the time trigger occurs: the map is run.

When the time trigger occurs, if multiple source events exist that satisfy a wildcard: multiple instances of the map are run for all event sets that exist.

FirstAvailable

If no source trigger exists when the time trigger occurs: then the map goes into pending state and waits for the first source trigger to appear, then the map runs.

If one source trigger exists when the time trigger occurs: the map is run.

When the time trigger occurs, if multiple source events exist that satisfy a wildcard: the map is run with the first known source event.

Pending instance thresholds

The **Pending Instance Thresholds** setting enables a listener to pause or resume for a specific watch. This setting allows high and low initialization-pending maps for any given watch to turn listeners on or off for that particular watch only.

The use of this setting can improve performance by enabling watches to run that could not run previously because of other watches monopolizing system resources.

If two watches share the same event (what an adapter is triggering), the two watches must have the same values for high and low pending instance thresholds. Otherwise, an error occurs, preventing the Launcher from starting.

During initialization, properties of the watch, including high and low parameters, are written to the debug trace (.txt) file (see [Trace and Log Files Naming Conventions](#) for the specific naming conventions).

Note: The global **InitPendingHigh** and **InitPendingLow** settings in **config.yaml** regulate all listeners and events. The **Pending Instance Thresholds** Launcher setting regulates listeners for a specific watch (or watches) based on individual watch settings for high and low pending maps. If the **InitPendingHigh** and **InitPendingLow** settings in the runtime/launcher section of the **config.yaml** file are also set, the **Pending Instance Thresholds** Launcher settings will take precedence.

Setting	Description
Hi	Value (no limit) for the watch to pause the event or events for that watch. The default value is 0 .
Low	Value (no limit) for the watch to resume the event or events for that watch. The default value is 0 .

MapInstance Timeout

The MapInstance Timeout setting is the length of time that a map instance can run before a warning or an error is logged in the compound system log. You can specify the time limit in hours, minutes, or seconds.

The map does not fail when it runs longer than specified by the MapInstance Timeout setting.

To log warnings and errors, you also must enable the LogWarning and.LogError logging options in the runtime/launcher section of the config.yaml file.

The following table shows the logging settings and sample log messages for a map instance that runs for 15 seconds:

MapInstance Timeout Setting	config.yaml File Setting	Logged Message
Warning = 10 seconds	warning=true	Map Warning: TIMEOUT watch num/card num: 15s > 10s
Error = 10 seconds	rror=true	Map Error: TIMEOUT watch num/card num: 15s > 10s

Trigger event settings

Event settings initiate map components of a system. Each map component is run based on a time event, source event, or combination of multiple events. A source event can be a file, message, or database event. If a map component is not assigned an event trigger, the Launcher does not initiate that map to run.

Note: When you analyze a system on the IFD and your system is configured for Launcher mode, a warning for a source or compiled map component that has no associated event triggers results.

A *watch* consists of the map and the set of events that initiate that map. Each event that contributes to the initiation of a map is called a *trigger*. Whenever the Launcher detects a new event trigger, the watches associated with that trigger are either initiated, placed in an initiation pending state, or maintained in an initiation pending state until all specified event triggers have occurred.

The same map can appear in different watches. For more information on this situation, see [Same Map in Different Watches](#).

- [Time events](#)
- [Source events](#)

A source event is the change in state of an input that you designate as a trigger. A source event can be associated with a file, database, or message source. A source event can trigger a map component with or without a time trigger.
- [Time and source event coordination](#)
- [Launcher input trigger files](#)

Time events

A time event is the occurrence of a specific time that you designate to initiate a watch. You can select to trigger a map based on a singular time event or cycle according to a specified time increment.

To specify an event based on time, select the **TimeEvent** setting in the **Launcher** Settings window in the IFD. See ["TimeEvent"](#) for information about how to configure these settings.

Source events

A **source event** is the change in state of an input that you designate as a trigger. A source event can be associated with a file, database, or message source. A source event can trigger a map component with or without a time trigger.

When the **SourceEvent** setting for a map input card is **ON** and the map has a time trigger, the **TimeEvent...>EventCoordination** setting is available. This setting controls the coordination of time and input events.

The following table describes the file, message, and database source events.

Event	Description
File Event	A change in state of that file, its creation, or a change in the time stamp for an existing file. If you use file triggers, you can enter the full path name for that file as the source name. You can also enter a path name relative to where the compiled map resides (that is, the .mmc file). The file name can also include wildcards, as described in Wildcards in Adapter Names .
Message Event	The occurrence of a new message in a messaging system
Database Event	Database modification that triggers a map. Database sources for certain database types can be used as source events for the Launcher, which allow insertions, deletions, or updates to database tables as the trigger mechanism that launches a map. The specific database trigger activities (that is, insertions, updates or deletions) are defined for the query in the Database Interface Designer. The event or events might be the result of a map or another application. A typical use is to configure a trigger that launch a map to run when another map has completed and modified a table. Note: For more information about database triggering, and the specific databases that support it, see <i>Database Triggers</i> in the <i>Database Interface Designer</i> documentation. For platform and version-specific information, see the release notes .

Time and source event coordination

When a map has a time trigger and one or more source triggers, use the Launcher settings to coordinate these different events. You specify how these events are coordinated using the **EventCoordination** setting under **TimeEvent** in the **Launcher Settings** window. This setting specifies how the Launcher executes when a map has both a time trigger and at least one source trigger (which may include wildcard values). The two options for **EventCoordination** are **FirstAvailable** or **All or NONE**.

The following table describes the conditions that might exist at execution time and the result for each Event Coordination option.

Condition	FirstAvailable	All or NONE
No source trigger exists when the time trigger occurs.	Map goes into pending state and waits for the first source trigger to appear, then the map runs.	This time trigger interval is skipped and nothing is run.
One source trigger exists when the time trigger occurs.	Map is run.	Map is run.
When the time trigger occurs, multiple source events exist that satisfy a wildcard.	Map is run with the first set that appeared, instances of the same map are run sequentially with each set at successive time trigger intervals, based on the order that the event set appeared.	Multiple instances of the map are run for all event sets that exist.

Launcher input trigger files

These are recommendations for creating Launcher input trigger files if you experience the following issues:

- Multiple triggering of the same file.
- Unexpected EOF (end of file) during map processing, which is reported with the following error message: Input type contains errors.

If you have experienced the above issues, the following command usage is recommended:

- Use the `rename` system call or `mv` shell command to create input files on UNIX platforms.
- Use the `rename` or `ren` command on Windows platforms to create input files.
- Do not use the `copy` command or similar non-atomic commands or functions (such as FTP) because the Launcher file adapter can detect a file before it is fully copied.

This is applicable for both Windows and UNIX platforms.

Note: To avoid the potential problems listed above, you should use a **.tmp** filename extension during file creation and rename the file afterward.

Related information

- [How to prevent Launcher from triggering multiple times on same input file](#)

Source and target settings

When you compile a map from the Map Designer, you supply a default set of sources and targets. From the IFD, you can override those sources and targets to adapt a map to the system in which that map is used.

These source and target setting options enable you to treat a map as a transaction for mission critical applications. The option settings are important when you use multiple target adapters. For example, if one output inserts data to a database correctly, but the next output fails to produce a related message, you can rollback the database insert.

Source adapters are used to read input data from a file, query a database, get a message from a queue, or receive the results of a call to an application. Target adapters are used to write output data to a file, insert or update data into a database, put a message on a queue, or call an application to send data to it.

From the **Launcher Settings** window, you can select source adapters for inputs and target adapters for outputs. Select the adapter type from the **Source** or **Target** drop-down list and then enter specific settings for the selected adapter type.

Adapter classes

The source and target adapters are organized into four adapter *classes*. Each adapter class has an icon that appears in the system diagrams in the IFD that indicates the Launcher properties of that adapter. The following table shows the Launcher properties for each adapter class. It also shows the properties available in all Server modes (Command Server, Launcher, and SDK) and those properties that are available only on the Launcher.

n

Table 1. Properties available in all server modes (Launcher, Command Server, SDK)

Properties	Application Adapters	Database Adapters	File Adapters	Message Adapters
OnSuccess action	Yes	Yes	Yes	Yes
OnFailure option	Yes	Yes	Yes	Yes
Retry option	Yes	Yes	Yes	Yes
Override option for compiled adapters	Yes	Yes	Yes	Yes

Table 2. Properties available only in the Launcher

Properties	Application Adapters	Database Adapters	File Adapters	Message Adapters
Event management	No	Yes	Yes	Yes
Resource management	No	Yes	Yes	Yes
Wildcard instantiation based on source events	No	No	Yes	Yes
Wildcard coordination with source events	Yes	No	Yes	Yes

- [Wildcards in adapter names](#)
- [File adapters](#)
- [Database adapters](#)
- [Application adapters](#)
- [Message adapters](#)

Wildcards in adapter names

Use wildcard notation to identify a source or target to coordinate data threads in the same map or among many maps. The **Launcher Settings** window allows only valid source and target names. Inconsistencies are analyzed from the IFD and only valid names are accepted. In the following list, wildcard notation refers to a name with asterisks or question marks in it.

General rules for using wildcards in adapter names:

- Directories *cannot* have wildcard notation in their names.
- A database source name *cannot* contain wildcard notation.
- If wildcards are used as sources or targets, at least one source trigger must have a wildcard.
- Question marks *cannot* be used in target file names.
- Multiple wildcards can be used in a source trigger name of a map, as long as there are no other source triggers.
- If more than one source file name contains a wildcard, that wildcard must be the same wildcard.
- Sources that are not event triggers and target names can contain at most one asterisk.

File adapters

When you select a file adapter, you supply the name of a specific file or a file name that contains a wildcard. If a source adapter is a file, you can specify that the file be a source trigger. If you use an input file with a wildcard in its name, the Launcher initiates a watch for any file that matches that name. If more than one source for the same watch has a wildcard, the first one to change state is used as the value for the other input sources and output sources that have that wildcard in its name.

Use watch directories only for trigger files to avoid the overhead caused by polling non-trigger files and to ensure the best performance of the Launcher file adapter listener. For example, if the watch directory is FULLPATH/WatchDir and the watch wildcard is set to trigger from *.txt files, do not keep configuration files (*.conf) in the watch directory.

Files used as sources or targets have the following characteristics:

- [Event triggers](#)
- [Wildcards](#)
- [File names, path names, directories, and overrides](#)
- [OnFailure and Retry settings](#)
- [Effect of OnSuccess card setting on file event triggers](#)
The OnSuccess card setting can affect performance when an input file is a source event in a Launcher system.
- [Troubleshooting tip](#)

Event triggers

- A file can be used as a source event trigger for a map
- Files used as triggers cannot have a **.tmp** extension. The Launcher uses this extension for temporary files. For example, temporary files are created when rollback options are selected.

Wildcards

- Can be used within a file name
- Cannot be used for directory names
- Can be used in Audit Log file names providing the input that a trigger has the necessary wildcards to resolve the name for resource coordination

See the *Resource Adapters* documentation for more information about wildcards.

File names, path names, directories, and overrides

- File names can be specified with a full path name or a relative path name. Relative path names are relative to the path of the compiled map.
- File names can be coordinated with other sources and destinations.
- Files can be used as an override for any source or destination type.
- If the directory for a file source that is an input trigger for a map becomes disconnected while the Launcher is running, the Launcher will attempt to reconnect to that watch directory every 10 seconds.

OnFailure and Retry settings

An **OnFailure** setting can be specified for a file that a map writes to in case the map does not run successfully. If an **OnFailure** setting of **Commit** is specified, the changes to the file remain intact, even if the map does not complete successfully. If an **OnFailure** setting of **Rollback** is specified and a map does not complete successfully, the file remains in its original state. That is, if a file originally exists, it is not updated. If a file does not originally exist, it is not created.

When the Launcher settings specify **OnFailure=Rollback**, the File adapter can move the source-event file to an error directory when a map fails. A default error directory is always available when cooperative file listening is enabled in the config.yaml file. When cooperative file listening is disabled, an error directory is available only if you configure it in the Launcher settings.

The **Retry** option applies to files that exist. **Retry** is used for files that exist, but might be in use by applications unknown to the Launcher and cause a resource conflict. For example, a file to be opened is already open for write or a file to be opened for write is already opened for read or write.

Related reference

- [Error directory configuration](#)

Related information

- [Cooperative file listening: Multiple watches monitor a single trigger directory](#)

Effect of OnSuccess card setting on file event triggers

The OnSuccess card setting can affect performance when an input file is a source event in a Launcher system.

The Launcher file listener maintains and monitors a list of the files that have triggered the Launcher to run a map. Each time the file listener receives an input file candidate for the Launcher, it compares the input file's name and timestamp against the list before it notifies the Launcher of the file.

- Files that have the **OnSuccess=Keep** card setting remain on the list after triggering an event.
- Files that have the **OnSuccess=KeepOnContent** card setting remain on the list after triggering an event if the file is not empty.
- Files that have the **OnSuccess=Delete** card setting are removed from the list after triggering an event.

Over time, files that have the **OnSuccess=Keep** card setting cause the list to grow. A longer file list takes the file listener longer to traverse and maintain, which affects the performance of the Launcher. In production environments that use the Launcher, either set the **OnSuccess** card setting to **Delete**, or configure a keep directory.

When the Launcher **OnSuccess** setting specifies **Keep** or **KeepOnContent**, the File adapter can move the source-event file to a keep directory when a map succeeds. A default keep directory is always available when cooperative file listening is enabled in the config.yaml file. When cooperative file listening is disabled, a keep directory is available only if you configure it in the Launcher settings. By using a keep directory, you can retain a source-event file without affecting the performance of the file listener.

Related reference

- [Keep directory configuration](#)

Related information

- [Cooperative file listening: Multiple watches monitor a single trigger directory](#)

Troubleshooting tip

To enable trace (logging) functionality for the file listener, you must enable the option in the **config.yaml** file.

To enable logging for the file listener

1. Open **config.yaml**, which is located in the product installation directory.
2. Scroll down to the **runtime/m4file** heading and locate the following text:

```
; used to enable logging for the file listener -- it should  
; be a valid complete path (directory and filename)  
;Logging=
```

3. Uncomment the `Logging` line by removing the semicolon in front of it.
4. Enter a complete path and filename for the log file. For example:
`Logging=install_dir \m4trace.log`
5. Save and close the file.

Database adapters

Database source adapters enable you to query a database and use the results of that query as input data. Target adapters enable you to insert or update data to a database. Connections to a particular database are generally made once for each map instance. However, if one database resource uses a rollback option and another does not, the database adapter will use a separate connection for the rollback option.

Before you specify the database to be used by the Launcher, you must first define and configure connections to the selected database. For example, before you specify the Oracle **Connect String** (in the Database Interface Designer) to be used by the Launcher on a computer installed with Oracle client, you must ensure that the value you provide for the **Connect String** matches the Oracle service name definition on that client.

Note: For more information about how to define or configure connections to a database, see the documentation for your specific operating system and database.

- [Characteristics of database adapters](#)

Characteristics of database adapters

Event triggers

Database sources can be used as source event triggers.

Wildcards

Wildcards cannot be used for database sources and targets.

Overrides

Database adapters can be used as an override for any source or target type.

Database connections

- Database connections are coordinated across sources, targets, and database functions used in a single map instance. For example, if one database target is specified with a rollback option and that database is accessed by a `DBLOOKUP` function within any output, the connection will apply for data accessed in that function.
- A connection to a database is specified for each unique combination of data-source name, user ID, password, rollback option and map instance. You can force a new connection, for example, by using different user IDs or data source names. You can take advantage of single connections for a particular map instance by keeping all the database specific parameters the same.

OnSuccess, OnFailure, and Retry settings

- The **OnFailure** setting applies to modifications made to a database source or target within a particular map, burst or card instance (depending on the setting for **Scope**). For example, an **OnFailure** setting of **Rollback** specified for a source query applies only when that query references a stored procedure, which modifies the database
- A database transaction is synonymous with a connection for the purposes of **OnFailure** and **OnSuccess** settings. The database transaction begins when the connection is established and ends when the map completes. If the map does not complete successfully and a rollback is associated with that transaction, the database transaction is rolled back. This means, for example, that if multiple sources and targets of the same map share the same connection and all have an adapter rollback specified, any modification made to any one of these sources or targets are part of the same database transaction

- If a map does not complete successfully, all DBQUERY and DBLOOKUP functions used in that map always rollback. If a DBQUERY or DBLOOKUP function fails and the map succeeds, the function returns NONE, but any modifications to a database specified in the function will rollback.
 - The **Retry** option applies when attempting to connect to a database.
 - The **OnSuccess** option for a source or target is not supported. However, SQL delete statements and stored procedures used as sources can be used for deletion purposes.
- Note: If more than one map instance deletes the same data, the first map instance that completes will delete that data. Deleted database data is not coordinated among map instances.

Application adapters

Application adapters are programs or functions called by the Launcher after a map instance has been initiated. The use and availability of both program and library function application adapters is platform dependent. If an application adapter is a function, that function must conform to a specific prototype. Its prototype, or interface syntax and semantics are defined in the *TX Programming Interface* documentation.

- [Characteristics of application adapters](#)

Characteristics of application adapters

Event triggers

Application adapters cannot be used as source event triggers, they are client applications within a map instance.

Wildcards

Wildcards can be used in an application adapter override command to coordinate trigger events with adapter arguments.

OnSuccess, OnFailure, and Retry settings

- Custom adapters can make use of the **OnFailure** and **Retry** options. The Launcher passes the information to the adapter. The adapter is responsible for responding to the rollback and retry.
- **OnSuccess** options are passed to application adapters.
- **Retry** options apply only if an adapter can be found and if applicable to that adapter. If not available on the server where the Launcher is installed, you will get a source not available or a target not available message.

Note: If you use your own application adapter, you are responsible for including functions for **OnFailure** and **Retry** options. For more information about using your own application adapter, see the *TX Programming Interface* documentation.

Message adapters

Message adapters allow you to send or receive discrete units of information, typically identified by a unique name. The naming conventions are adapter specific. For example, IBM® WebSphere® MQ names or *message identifiers* are 24-byte text items. SAP R/3 message names are called *transaction identifiers*. The storage mechanism for a message is also adapter-specific. For example, IBM WebSphere MQ uses queues to store messages.

When a source message adapter is used with an Launcher, the adapter notifies the Launcher when specific source event messages are available.

To use message adapters, you must have message adapters for the server where the Launcher is installed. For information about command options, see the Resource Adapters documentation.

- [Characteristics of message adapters](#)

Characteristics of message adapters

Event triggers

Messages can be used as source event triggers for a map.

Wildcards

Wildcards can be used for message identifiers. If the message identifier is specified as a wildcard, the value of that wildcard is derived from the value of the message identifier of a message event trigger or from a file source with a wildcard in its name.

OnFailure, and Retry settings

- The **OnFailure** option for a source or target depends on the specific messaging adapters. For example, for IBM® WebSphere® MQ the source rollback option includes an option to place the source message on an error queue.
- The **Retry** option for a source or target depends on the specific messaging adapters. In general, the command option for the adapter must be valid on the computer where the Launcher is installed. If a connection to a queue cannot be made because the specified queue is not available, you will get a source not available

message or a target not available message. If connection parameters are valid, but the message is unavailable due to, for example, a connection limit that has been reached or the message is locked, the retry will be attempted.

Adapters that support event triggers

The following list displays the adapters that can be used as event triggers by the Launcher.

Note: For platform and version-specific information, see the [release notes](#).

- BW Staging BAPI
- Connect:Direct
- Database (See note below.)
- File
- HL7 MLLP
- HTTP
- IBM® IMS TM Resource Adapter
- IBM WebSphere® MQ (client)
- IBM WebSphere MQ (server)
- JMS
- MSMQ
- Oracle AQ
- R/3 ALE
- R/3 BAPI
- R/3 BDC
- Socket
- TIB/RVTX

Note: For information about the specific databases that support event triggers, see the *Database Triggers* in the *Database Interface Designer* documentation.

Connection testing adapters

The following list of adapters supports connection testing.

Note: See the [release notes](#) for platform- and version-specific information.

• BW Staging BAPI	• MSMQ	• R/3 ALE
• DB2®	• ODBC	• Socket
• HTTP	• OLE DB/MS SQL Server	• Sybase
• IBM® WebSphere® MQ (client)	• Oracle	
• IBM WebSphere MQ (server)	• Oracle AQ	
• Informix®	• R/3 BAPI	
• JMS	• R/3 BDC	
• MS SQL Server		

Subsystem component execution settings

A subsystem component references another system that you have defined. As such, a subsystem is an executable unit. You use the IFD to specify the execution settings of subsystem components. Because the IFD treats each subsystem component as a unique instance, these execution settings are maintained separately with each component.

A subsystem component has a defined set of interfaces based on the sources and targets of the various maps that it contains. The inputs and outputs of a subsystem are derived according to the following rules:

- An input of a subsystem is a source that is not internally produced by one of its components.
- An output of a subsystem is a target that is internally produced by one of its components, and is not deleted.

You define execution settings for each subsystem component in your system by accessing the **Launcher Settings** window. To display the **Launcher Settings** window in the IFD, click the **Execution Settings** button of the subsystem component icon with the assigned server displayed.

You can also access this window by selecting the **Edit Execution Settings** command from the context menu for a subsystem component.

- [Launcher settings dialog](#)

Launcher settings dialog

Use the **Launcher Settings** window to configure the settings for the subsystem component.

- [Server](#)
- [ExecutionType](#)
- [Inputs and outputs](#)

Server

The **Server** setting is used to specify the server on which the system will be deployed. The default value for the **Server** setting is **None**.

Value	Description
Local Server	A system is going to run on the local workstation, usually where the IFD is installed. You might want to assign Local Server to systems to use your workstation as a test environment to verify execution. Or, you can plan to use Local Server as a part of your production environment for system execution.
Distributed	Distributed server; a system is composed of subsystems that are assigned to run on different servers in a coordinated manner to distribute the processing. Components in a distributed system must be subsystems, not maps.
None	No server selected; no server is assigned. The system is not executable. Systems that have an assigned server of None do not appear in the Component view in the Navigator. They can be used as subsystems within a system that has a server assigned. A subsystem with an assigned server of None derives its server from its parent system that has a server assigned. Subsystems always display in the Navigator hierarchical view unless the parent system also has a server of None .
Custom	User-defined server

ExecutionType

The **ExecutionType** setting is used to specify the execution mode for the system. The default value for the **ExecutionType** setting is **Launcher**.

Value	Description
Launcher	System will be deployed for execution by the Launcher.
Command	System will be deployed for execution by the Command Server.

Inputs and outputs

Expanding the **Input(s)** and **Output(s)** settings shows the map cards associated with the maps in the subsystem. The values shown are based on the settings specified in the map but the **Launcher Settings** window enables you to override these values. The values you can override are:

- **FetchAs** (for input cards)
- **WorkArea**
- **Backup**
- **Source** (for input cards)
- **Target** (for output cards)

See the *Map Designer* documentation for more information about card settings.

Map initiation scenarios

This documentation describes map initiation scenarios and coordination.

- [Coordinating and using wildcards](#)
- [Multiple map instances in the same watch](#)
- [Same map in different watches](#)
- [Cooperative file listening: Multiple watches monitor a single trigger directory](#)

With cooperative file listening, you can scale your solution to the volume of input files. Multiple watches can run on multiple Launchers and monitor the same trigger directory for arriving source-event files. The File adapter ensures that only one watch triggers on a particular source event. As the volume of incoming source events increases, you can add Launchers to the same computer or to different computers.

- [Different maps that use the same source](#)
- [Different maps use the same target](#)
- [One map's target is another map's source](#)
- [HTTP requests trigger maps](#)

An external HTTP client can trigger a map in a Launcher system by sending a request to the Launcher HTTP Listener. The HTTP Listener provides secure, two-way communication between external clients and the Launcher.

- [Map initiation states](#)

Coordinating and using wildcards

You can use wildcard notation to specify sources or targets that have varying names. To determine the value of a wildcard, the Launcher requires that at least one wildcard-using source is specified as a trigger event.

- [Single source name](#)
- [Matching multiple source names](#)
- [Matching source to target names](#)
- [Multiple wildcards in a source name](#)
- [Threading wildcards across multiple maps](#)

Single source name

If you want a map to run each time a file appears in a particular directory, use a wildcard as part of the source name. An example is `install_dir\receive*.any`.

In this example, each file in the `install_dir\receive` directory with an extension of `.any` starts an instance of the **Run All** map. Depending on other Launcher settings, instances of the **Run All** map can be concurrent or sequential.

For some adapters, an implicit wildcard can be used. For example, if you use the IBM® WebSphere® MQ source adapter, the message identifier is an option. In this case, if no message identifier is specified, any message on a particular queue will initiate a map. An example is `-QMN MyManager -QN MyQueue`.

If you require all message identifiers that end in `XYZ` to run the map, use a specification that is such as `-QMN MyManager -QN MyQueue -MID *.XYZ`.

Depending on other Launcher settings, instances of the **Run All** map can be concurrent or sequential.

Matching multiple source names

If more than one source name for a map has a wildcard, you can use the first source with that wildcard to set the wildcard value for the other sources required by the same map instance.

For example, consider a business using an order system that requires a credit check for all orders. Orders that pass the credit check need to be confirmed; the business does not want to mismatch an order with the wrong credit check.

In this example, the order message is a source trigger. When a new order arrives, its message identifier is used as the value of the wildcard and **CreditCheck** will not run until a credit report message with the same message identifier arrives.

Matching source to target names

If a source has a wildcard, any target that contains an asterisk is assigned the source wildcard value. Modify the credit check example used previously to include an asterisk in the target message's identifier:

The value of the Order message identifier wildcard is assigned to the **CreditStatus** message identifier wildcard. For example, if the value of Order's message identifier is `AAA1212TW`, the **CreditStatus** message identifier is assigned the value, `STATAAA1212TW`.

Multiple wildcards in a source name

If a source name has multiple wildcards and the target name has a single asterisk wildcard, the target name asterisk is replaced with the first wildcard value, next wildcard value, and any characters that exist between the wildcards.

For example, with a source name of `*a??b.c` and a target name of `*.d`, if the source file name instance is:

`xyazzzb.c`

the target name assigned by the Launcher is:

`xyazzz.d`

Threading wildcards across multiple maps

You can track a data object and other objects produced as a result of its existence. These related objects can be produced asynchronously and feed yet other transformation steps that produce more objects.

You can tag each related object to distinguish the thread of actions taken as a result of the event of the first object that started.

For example, suppose there is a travel reservation system. One component of the travel reservation system accepts travel requests with a proposed itinerary. That component sends out request messages to airlines and hotels for availability. After there is a response from an airline and hotel for that itinerary, a response is produced

so that a reservation can be made.

To track a travel request to match it to a reservation as described previously, you can design an example system to have an **Availability** map component with an input event trigger that accepts the travel requests represented by the **-QMN MyManager -QN Itinerary -MID.*** input card. The **Availability** map can send out request messages represented by the **1: AirlineRequest** and **2: HotelRequest** output cards. The responses can be represented by the **-QMN MyManager -QN Airline -MID Air*** output card from the airline and the **-QMN MyManager -QN Hotel -Htl*** output card from the hotel, which can be used as input by the **Reservation** map component with an input event trigger to produce the **-QMN MyManager -QN Response -MID *** output card.

Using wildcards, each Itinerary message identifier can be tracked by the same identifier on the Hotel queue, the Airline queue, and the Response queue.

Multiple map instances in the same watch

The same map might be initiated more than once by the same watch. For example, a map configured to run once every minute will run multiple times. You might, or might not, want multiple instances of the same map running at the same time. How Launcher settings are specified determine specific Launcher actions.

- [Using time events](#)
- [Using source events](#)

Using time events

In this scenario, a map is configured to run every minute. The map starts at 3:00 P.M. and takes over one minute to run. Another instance of that same map could be initiated at 3:01 P.M., and you could have two instances of the same map running at the same time. You can design an example **AppToApp** map with a time event trigger, **1: OrderAppln_In** input card and **1: OrderAppln_Out** output card. The map takes a minute-and-a-half to run on average. Each time it runs, it uses an application adapter to get data expected from some communication session, transforms it to its output form, and puts it in a file. Assume that this map is the only one controlled by the Launcher. Here are the different ways the map could run in real-time.

- [Case 1: Different map instances can run at the same time](#)
- [Case 2: Different map instances can run sequentially](#)
- [Case 3: Skip if Busy can be used to omit map instances from starting](#)

Case 1: Different map instances can run at the same time

If you select unique work files, different map instances can run at the same time.

Unique work files on their own do not guarantee map instances will run at the same time. For example, if the output file is appended, the resource manager will not let multiple map instances run at the same time because the result of one map instance is dependent on another.

Case 2: Different map instances can run sequentially

If unique work file names are not selected, different map instances will *always* run sequentially. A second map instance will begin running after the first map instance has completed running.

There are other situations for which maps run sequentially. For examples, see [Using Source Events](#).

Case 3: Skip if Busy can be used to omit map instances from starting

The Skip if Busy Option can be used to omit map instances from starting when another instance is in process.

In this case, it might not matter what options you select for work files and log files because map instances always run sequentially.

Note: If you want multiple instances of the same map to be able to run at the same time, be sure to select unique work files and unique log file run options. These options create different work files and log files for each map instance so that one map instance will not interfere with another. If you do not select these options, the Launcher will keep the second instance in a pending state until the first instance has completed.

Using source events

In this scenario, a map is configured to run each time a new file appears in a specified directory. A map instance starts when the first file appears and takes over a second to run. Another file appears a half a second after the first file. There could be two instances of the same map running at the same time. You can create an example **AppToApp** map with an event trigger, *install_dir\MQS\APPLN*.TXT* input card and **1: OrderAppln_Out** output card.

Assume that this map is the only one controlled by the Launcher. The following case describes different ways the map could run in real-time.

- [Case 1: Different map instances run at the same time, if all sources and targets are different](#)
- [Case 2: Different map instances run at same time or sequentially based on source and target settings](#)

Case 1: Different map instances run at the same time, if all sources and targets are different.

How run options are selected might affect the length of time a map instance is in process. Assume run options are set as one of the following combinations:

- Select unique work file names or work area in memory and select no log file.
- Select unique work file names or work area in memory and select unique log file names.

In this case, each map instance is independent and the length of time each map instance runs depends only on the amount of processing time associated with the size of the data and the mapping rules used.

If sources and targets are independent, and unique work files are used, but map instances share log files or other map system resources, you might see map instances in process run longer than expected. This occurs because the Launcher will initiate an instance based on available event triggers and then waits for map system resources to become available.

Note: Sources can be the same if they do not have the potential of a map writing data to it. See [Different Maps That Use the Same Source](#).

Note: If you use work files with the WorkFilePrefix setting of MapName (available under the Workspace setting; see [MapSettings > WorkSpace](#)), be sure to use the retry option if multiple map instances can run at the same time. If a retry is not set, the map will fail to run if these resources are in use.

You might also see this same effect if a source or target resource is required, it cannot be shared, and it is in use by an application unknown to the Launcher. In this case, be sure to use the source or target retry option, or the map instance will fail to run.

Case 2: Different map instances run at same time or sequentially based on source and target settings

Unique log file names and unique work file names do *not* guarantee map instances will run at the same time. For example, if the same input file is used each time a map runs, and it is updated by a map rule, the resource manager will not let multiple map instances run at the same time because the result of one map instance is dependent on another.

Map instances will run sequentially under the following conditions:

- If a map uses the same file or message as both a source and target
- If a source is the same file each time a map runs and a map rule is used to update it
- If a target is a file and that file is appended

If any of the previous conditions apply, you will not see two instances of the same map running at the same time.

Same map in different watches

You can have the same map in different watches. The initiations of map instances of one watch can be configured to be independent from the initiation of map instances of other watches.

For example, you want to process orders based on priority. All orders look alike before processing and are converted to the same output form. Priority orders are placed on one message queue. Normal orders are placed on another queue. This is the same map with different component names and different sources.

You want **Priority Orders** to run as fast as possible and (if more than one priority order appears at the same time) you want to process them all at the same time. Normal orders can be processed at the same time, as long as there are enough computer resources available to do so. In this case, you configure **Priority Orders** to have a higher priority than **Normal Orders**.

You also set unique work files and unique log files, so that **Priority Orders** will not need to wait for map system resources. If you just keep work files for each map in a different directory, **Priority Orders** and **Normal Orders** could run at the same time. However, this would limit **Priority Orders** instances to run sequentially. It would also limit **Normal Orders** instances to run sequentially.

Note: If you are using the log file options and are doing this for more than one map, you might want to keep these files for each watch in a separate directory. Although unique log file names are globally unique, they do not contain the name of the map. For example, if you use log files to trigger other maps, keeping these in a separate directory will make sure the logs trigger the correct map instance.

Cooperative file listening: Multiple watches monitor a single trigger directory

With cooperative file listening, you can scale your solution to the volume of input files. Multiple watches can run on multiple Launchers and monitor the same trigger directory for arriving source-event files. The File adapter ensures that only one watch triggers on a particular source event. As the volume of incoming source events increases, you can add Launchers to the same computer or to different computers.

- [Horizontal scaling](#)

Cooperative file listening scales a solution horizontally, by adding more Launchers. It is designed to address performance bottlenecks that are due to maximum CPU utilization or excessive local I/O.

- [Cooperative file listening and fault tolerance](#)

With cooperative file listening, source-event files are retained when a Launcher or a map fails.

- [Cooperative file listening and load balancing](#)

By implementing the cooperative file listening load-balancing guidelines, multiple launchers that poll the same trigger directory for the same files can process a similar number of files to ensure that uniform triggering occurs.

- [Cooperative file listening configuration](#)

- [Balance the workload across watches and scale for large data](#)

With cooperative file listening, a single watch runs on multiple Launchers. On each Launcher where the watch runs, the file listener monitors the same trigger directory for the same files. Use the Pending Instance Thresholds and the Max Concurrent Map Instances Launcher settings in the Integration Flow Designer to help to balance the workload among Launchers.

Horizontal scaling

Cooperative file listening scales a solution horizontally, by adding more Launchers. It is designed to address performance bottlenecks that are due to maximum CPU utilization or excessive local I/O.

Cooperative file listening is of limited benefit if CPU utilization on the existing computer is not maximized. Instead, scale the system vertically by adding more Launcher processes or more threads within the same Launcher process.

Similarly, Cooperative file listening does not address performance bottlenecks that are caused by contention for a shared external resource, such as an NFS file system or database.

Cooperative file listening and fault tolerance

With cooperative file listening, source-event files are retained when a Launcher or a map fails.

When a Launcher fails and cooperative file listening is enabled, the File adapter automatically saves the source-event files of the failed Launcher. Other Launchers continue to process source-event files as they arrive, so the failure of one or more Launchers or maps causes minimal processing delays.

The File adapter saves the source-event files of the failed Launcher in a subdirectory of the trigger directory. The subdirectory is WTX_port_num_hostname, where port_num and hostname are the control port and host name of the Launcher.

When the failed Launcher restarts, the File adapter automatically returns the Launcher's source-event files to the trigger directory, where any Launcher that is listening can process them.

If the failed Launcher cannot restart, you can manually move its source-event files to the trigger directory.

When a map fails and cooperative file listening is enabled, the File adapter automatically saves the source-event files to an error directory. You can use the error directory as the trigger directory for an error-processing map.

File listener messages are logged in the Launcher log file.

Cooperative file listening and load balancing

By implementing the cooperative file listening load-balancing guidelines, multiple launchers that poll the same trigger directory for the same files can process a similar number of files to ensure that uniform triggering occurs.

Cooperative file listening configuration

- [Cooperative file listening requirements](#)

Cooperative file listening requires single source-event watches and source-event files that are created by atomic operation. It supports the GPFS and NFS shared network file systems as well as POSIX-compliant local file systems.

- [Cooperative file listening config.yaml file and Launcher settings](#)

You enable cooperative file listening in the config.yaml configuration file and in the /runtime/launcherSettings.

- [Error directory configuration](#)

When cooperative file listening is enabled and the output card specifies OnFailure=Rollback, the File adapter moves the source-event file to an error directory when a map fails. The error directory is a subdirectory of the trigger directory. You can use the default error directory, or specify the error directory as an absolute file path, a relative file path, or an alias that is defined in the Resource Registry.

- [Keep directory configuration](#)

A keep directory stores source-event files after a map successfully processes them. A keep directory is always available when cooperative file listening is enabled in the config.yaml file. When cooperative file listening is not enabled, a keep directory is available if you explicitly configure its directory path in the Launcher settings. The OnSuccess setting of the input card must be Keep or KeepOnContent to save source-event files in a keep directory.

Cooperative file listening requirements

Cooperative file listening requires single source-event watches and source-event files that are created by atomic operation. It supports the GPFS and NFS shared network file systems as well as POSIX-compliant local file systems.

Single source-event watches

Cooperative file listening supports only watches that trigger on a single source-event card. Multiple source-event cards are not supported. If two Launchers use multiple source-event cards and monitor the same directory, one Launcher can process the first source-event file and another Launcher can process the second source-event file, preventing the watch from triggering.

Source-event file creation

Source-event files must be created by atomic operation. When a file is created with a non-atomic operation, its timestamp can change as the file is written, which can cause the Launcher to trigger more than once on the same file.

File system requirements

Cooperative file listening supports the General Parallel File System (GPFS) and Network File System (NFS) shared network file systems, as well as POSIX-compliant local file systems.

On Windows platforms, cooperative file listening supports file triggering on Samba-mapped UNIX drives. It does not support local or mapped NTFS or FAT drives.

Related information

- [How to prevent Launcher from triggering multiple times on same input file](#)

Cooperative file listening config.yaml file and Launcher settings

You enable cooperative file listening in the config.yaml configuration file and in the /runtime/launcherSettings.

The config.yaml configuration file setting

To enable cooperative file listening, edit the `[runtime/m4file]` section of the config.yaml configuration file and set `FileListenCooperativeListener=true`.

Launcher settings

Use the Integration Flow Designer to set the required Launcher settings for cooperative file listening. If `FileListenCooperativeListener=true` but the required Launcher settings are not configured, the Launcher does not start.

Table 1. Required Launcher settings for cooperative file listening

Option	Value
Input(s) > card_name > GET > Source	File
Input(s) > card_name > SourceEvent	On

Error directory configuration

When cooperative file listening is enabled and the output card specifies `OnFailure=Rollback`, the File adapter moves the source-event file to an error directory when a map fails. The error directory is a subdirectory of the trigger directory. You can use the default error directory, or specify the error directory as an absolute file path, a relative file path, or an alias that is defined in the Resource Registry.

Specify the error directory in the `Input(s) > card_name > SourceEvent > ErrorDirectory` field of the Launcher settings. If the fully qualified path to the error directory exceeds 260 bytes, the Launcher does not start.

If a map fails when cooperative file listening is enabled in the config.yaml file but no error directory is specified in the Launcher settings, the File adapter saves the source-event files in the default `WTX_ERRDOCS` subdirectory of the trigger directory. If the default `WTX_ERRDOCS` directory does not exist, the launcher creates it. The permissions of the default `WTX_ERRDOCS` directory are the same as the parent trigger directory.

The following table illustrates how the error directory paths resolve, relative to the `C:\myInput` trigger directory.

Table 1. Resolution of error subdirectory paths in `C:\myInput`

Specified path in ErrorDirectory field	Fully-qualified path
errors	<code>C:\myInput\errors</code>
<code>C:\errors</code>	<code>C:\errors</code>
<code>..\myErrors</code>	<code>C:\myInput\..\myErrors</code>
No directory specified	<code>C:\myInput\WTX_ERRDOCS</code>

If you disable cooperative file listening and you specify an error directory in the Launcher settings, the File adapter moves the source-event file to the error directory if a map fails and `OnFailure` is set to `Rollback`.

If cooperative file listening is disabled and you do not specify an error directory, the processing of the source-event file depends on the `OnFailure` setting.

Keep directory configuration

A `keep` directory stores source-event files after a map successfully processes them. A `keep` directory is always available when cooperative file listening is enabled in the config.yaml file. When cooperative file listening is not enabled, a `keep` directory is available if you explicitly configure its directory path in the Launcher settings. The `OnSuccess` setting of the input card must be `Keep` or `KeepOnContent` to save source-event files in a `keep` directory.

Configure the keep directory path in the Input(s) `card_name` `SourceEvent` `KeepDirectory` field of the Launcher settings. You can specify the keep directory as an absolute file path, a relative file path, or an alias that is defined in the Resource Registry. If the fully qualified path to the Keep directory exceeds 260 bytes, the Launcher does not start.

If you enable cooperative file listening but do not specify a keep directory in the Launcher settings, the File adapter saves the source-event files in the default `WTX_KEEPDOCS` subdirectory of the trigger directory. If the default `WTX_KEEPDOCS` directory does not exist, the launcher creates it. The permissions of the default `WTX_KEEPDOCS` directory are the same as the parent trigger directory.

The following table illustrates how the keep directory paths resolve, relative to the `C:\myInput` trigger directory.

Table 1. Resolution of keep subdirectory paths in `C:\myInput`

Specified path in ErrorDirectory field	Fully qualified path
backup	<code>C:\myInput\backup</code>
<code>C:\backup</code>	<code>C:\backup</code>
<code>..\myBackup</code>	<code>C:\myInput\..\myBackup</code>
No directory specified	<code>C:\myInput\WTX_KEEPDOCS</code>

To use a keep directory when cooperative file listening is disabled, specify a keep directory in the Launcher settings and set `OnSuccess` to `Keep` or `KeepOnContent`. When the setting is `Keep`, the File adapter saves the source-event file when the map succeeds. When the setting is `KeepOnContent`, the File adapter saves the source-event file when the map succeeds and the source-event file is not empty.

Balance the workload across watches and scale for large data

With cooperative file listening, a single watch runs on multiple Launchers. On each Launcher where the watch runs, the file listener monitors the same trigger directory for the same files. Use the Pending Instance Thresholds and the Max Concurrent Map Instances Launcher settings in the Integration Flow Designer to help to balance the workload among Launchers.

The Pending Instance Thresholds pause and resume the file listener of a watch. The file listener pauses when the watch's backlog of unprocessed source events reaches the high threshold, and resumes when the backlog falls below the low threshold. While the watch is paused on one Launcher, the listeners on other Launchers can process the arriving source events, which balances the workload.

Configure the high and low Pending Instance Thresholds to ensure that maps trigger uniformly across multiple Launchers. Adjust these settings so that each Launcher processes a similar number of source-event files, proportionate to the processing speed of the Launcher's server.

For example, if a watch that typically processes 10000 files runs on three Launchers, set the high Pending Instance Threshold value to 3300 to ensure that a Launcher listener pauses when it acquires a third of the source-event files. While the watch on one Launcher is paused, the listeners on the other Launchers continue to process the incoming source-event files. Set the low Pending Instance Threshold value to 1000 to resume the paused listener when the Launcher processes most of the source-event files.

Also consider the average transformation time as you adjust the Pending Instance Thresholds. In general, faster transformation needs higher Pending Instance Thresholds for optimal throughput.

- If the high and low Pending Instance Thresholds values are too low, stopping and starting the listeners frequently can impact performance.
- If the high and low Pending Instance Thresholds values are too high, one watch can accumulate the inputs while the others remain idle.

With very large data and CPU-intensive transformation, low values for the Pending Instance Thresholds can achieve scalability and load balancing. Low thresholds limit the number of source-event files that each Launcher can acquire and resume the listener only when a Launcher has CPU power available to process them.

You can accommodate CPU-intensive maps and large data by tuning the Max Concurrent Map Instances Launcher setting in the Integration Flow Designer. The Max Concurrent Map Instances setting limits the number of map instances that can run simultaneously within a Launcher process.

When you configure the Max Concurrent Map Instances setting and the Pending Instance Thresholds, take into consideration:

- Average input volume per hour
- Average time per transformation
- Number of cooperative Launchers
- Number of processors per Launcher per computer
- CPU utilization during peak load

For example, consider a system that processes very large data. Each map takes approximately 60 seconds to run and utilizes nearly 100% of one CPU. The system has three Launchers (L1, L2, and L3) that are deployed on three different host computers (H1, H2, and H3). The Launchers trigger on the same directory.

Host	Number of CPUs
H1	4
H2	2
H3	1

You might configure the Max Concurrent Map Instances and Pending Instance Thresholds as follows to balance the load and achieve optimal throughput:

Launcher on host computer	Max Concurrent Map Instances	High Pending Instance Threshold	Low Pending Instance Threshold
L1 on H1	4	8	4
L2 on H2	2	4	2
L3 on H3	1	2	1

Related reference

- [Pending instance thresholds](#)

Related information

- [Max concurrent map instances](#)

Different maps that use the same source

Different maps that use the same source can run at the same time, as long as that source is not updated by one of those maps.

For example, a patient is admitted to a hospital. The admitting department produces an admitting message. Various other departments in the hospital use the admitting message. The medical records department needs to update the patient record. The accounting department needs to track the patient's charges. The nurses' station on the floor where the patient will be staying needs to be notified. Other departments might have a need to know.

Each department uses a separate system for its internal processing. And none of these understands how to process the admitting message. The same message must be converted to many different forms, one for each department.

You could create one map and do all the conversions in one step, but some departments need the information faster than others. You therefore decide to configure a different map for each department.

For example, create an **Admit** system in the IFD with a different map for each department, **Pharmacy**, **Medical Records**, **Accounting** and **Nurses**.

In this example, each map uses the same trigger source as input. An example input trigger source is **-QMN MyManager -QN ADMIT**. Each map has different targets. Example targets are **1:HL7_Style** for the **Admit To Pharmacy** map, **1:MedicalRecord** for the **Admit To Medical Records** map, **2:InsuranceBilling** and **1:PatientCharges** for the **Admit To Accounting** map and **1:FAX** for the **Admit To Nurses** map. Each map is independent from the other; each can run at the same time. Also, more than one instance of each map could run at the same time.

- [Impact of Delete settings](#)

Impact of Delete settings

All maps that share the same source and have no Delete action specified are run first. When these maps complete, a map that specifies the Delete action for that shared resource can be initiated. Only one map should use the delete action for this type of resource to avoid a **Source not available** message.

Different maps use the same target

If different maps use the same target, the type of target influences how maps are run:

- If the target is a file or a message, the resource manager will prevent different maps that use the *same* file or message from running at the same time.
- If the target is an application, the Launcher relies on the application adapter to be multi-threaded and assumes resources for different instances of that adapter are independent.
- If the target is a database, the database adapter communicating with the specific database handles shared resources.

One map's target is another map's source

When the target of one map is used as a trigger source for another, the Launcher does not start any map until that trigger occurs. However, if a file or message source is not a trigger, the Launcher initiates a map based on other event criteria.

HTTP requests trigger maps

An external HTTP client can trigger a map in a Launcher system by sending a request to the Launcher HTTP Listener. The HTTP Listener provides secure, two-way communication between external clients and the Launcher.

You configure the HTTP Listener to monitor its ports for an incoming HTTP request for a specific URL. You design a map in a Launcher system to use the HTTP request as an input trigger. The listener sends the request to the map's HTTP adapter input card and returns a response to the external client. The response is either the output from the map's HTTP adapter output card, or the status of the map execution from the HTTP input card.

The HTTP Listener example is installed in the *tx_install_dir\examples\httplsnr* directory.

- [HTTP Listener map execution results overview](#)
Regardless of whether it runs successfully, a map returns a response through the HTTP listener to the external HTTP client. If the map fails or the output card does not return a response to the client, the input card automatically returns a response.
- [HTTP Listener map performance and throughput](#)
You can analyze the HTTP Listener status information to tune the performance of Launcher systems. The status information includes response time details for URL watches that can indicate bottlenecks during peak load periods. It also displays details about connections, responses to the HTTP client, and logging.
- [HTTP Listener connection performance](#)
The HTTP Listener status provides details about the connections between the listener and its external clients and HTTP adapters. The data can indicate the need to adjust connection settings.
- [HTTP Listener browser access](#)
You can display map execution status and configure HTTP Listener settings through a local or remote browser, as well as through the Launcher Management

- Console.
 - [HTTP Listener logging](#)
HTTP Listener trace and audit logs capture details about listener events, socket activity, connections, and HTTP communications.
 - [Supported cryptographic protocols for the HTTP Listener](#)
You can configure the cryptographic strength that the Launcher uses to communicate with its Secure Sockets Layer (SSL) clients. The Launcher supports encryption protocols that range from SSL V2.0 to Transport Layer Security V1.2 (TLS V1.2). Other supported protocols include SSL V3.0, TLS V1.0, and TLS V1.1. Also available is a NIST Special Publication (SP) 800-131A-compliant mode that selects a TLS protocol version based on the cryptographic and certificate strength that is used by the SSL client.
-

HTTP Listener map execution results overview

Regardless of whether it runs successfully, a map returns a response through the HTTP listener to the external HTTP client. If the map fails or the output card does not return a response to the client, the input card automatically returns a response.

The response from the input card can include customized HTTP status codes and a customized text page that describe the map execution results. A default set of status codes and default status page ensures that the HTTP client always receives a response from the map.

- [Status code command \(-STATUS_CODE\)](#)
When the HTTP adapter **-STATUS_CODE** command is used on a map input card, it associates one or more map error codes with an HTTP status code.
 - [Status page commands \(-STATUS_PAGE and -STATUS_PAGE_DEFAULT\)](#)
When a map or card fails, the HTTP Listener sends the external HTTP client a text file that describes the status of the map execution results.
-

Status code command (-STATUS_CODE)

When the HTTP adapter **-STATUS_CODE** command is used on a map input card, it associates one or more map error codes with an HTTP status code.

If the map fails or its output card does not produce a response, the **-STATUS_CODE** command on the input card returns the HTTP status code to the HTTP client. The `/runtime/M4HTTP` section of the `tx_install_dir\config.yaml` configuration file defines default HTTP status codes that the listener returns to the client (if the input card does not override them with the **-STATUS_CODE** command).

Related reference

- [Map execution error and warning messages](#)
 - [Status Code \(-SC or -STATUS_CODE\)](#)
-

Status page commands (-STATUS_PAGE and -STATUS_PAGE_DEFAULT)

When a map or card fails, the HTTP Listener sends the external HTTP client a text file that describes the status of the map execution results.

The HTTP adapter **-STATUS_PAGE** command sends map execution results to the external HTTP client in a text file. A map input card can use the command to create a uniquely named file that's specific to an HTTP request. The map can dynamically create the file contents by using pre-defined variables that resolve when the map runs.

The HTTP adapter **-STATUS_PAGE_DEFAULT** command returns a customized default file to the external HTTP client when the input card or map fails. When an HTTP adapter input card does not use the **-STATUS_PAGE** or **-STATUS_PAGE_DEFAULT** commands, an HTTP failure returns the generic `m4http_statuspage.html` file specified in the `/runtime/M4HTTP` section of the `config.yaml` configuration file.

Related reference

- [Status Page \(-SP or -STATUS_PAGE\)](#)
-

HTTP Listener map performance and throughput

You can analyze the HTTP Listener status information to tune the performance of Launcher systems. The status information includes response time details for URL watches that can indicate bottlenecks during peak load periods. It also displays details about connections, responses to the HTTP client, and logging.

The HTTP Listener status displays metrics that can be used to tune the Launcher performance.

Response time

The overall time to process an HTTP request for a particular URL.

Register time

The overall time to match an HTTP request with an available URL watch.

Map time

The average time to run the map and return a response to the HTTP Listener. Response time is approximately the sum of register time and map time.

The URL watch information displayed in the listener status is similar to the information in this table.:

Table 1. Example URL watch information provided as part of the HTTP Listener status

Watch #	Busy	Map Return	HTTP Response	Response Time	Register Time	Map Time	URL
0	0	0	0	0.000	0.000	0.000	/admin

Watch #	Busy	Map Return	HTTP Response	Response Time	Register Time	Map Time	URL
1	0	0	0	0.000	0.000	0.000	/admin/update
2	0	0	0	0.000	0.000	0.000	/admin/refresh
3	0	0	0	0.000	0.000	0.000	/admin/trace
4	0	0	0	0.000	0.000	0.000	/admin/trace/socket
5	0	0	0	0.000	0.000	0.000	/admin/audit
6	1	0	1	0.017	0.001	0.000	/admin/status
7	0	0	1	0.004	0.001	0.000	admin/reset
8	0	0	0	0.000	0.000	0.000	/admin/users
9	0	0	0	0.000	0.000	0.000	/admin/users
10	0	0	0	0.000	0.000	0.000	/admin/image*
12	0	1	1	0.531	0.001	0.528	/example

A significant increase in register time during peak load periods can indicate a backlog of HTTP requests. You can address a backlog by adding more copies of the map to the system (.msl) or by adding another Launcher to load-balance the HTTP requests.

The HTTP Listener status also includes the total number of success, failure, and unavailable responses that the HTTP Listener returned to the external HTTP client.

The Management Console displays this information in the Status page. You can also display it in a browser by entering the `http://Launcher_host_address:http_port/admin/status` URL. The config.yaml configuration file controls the refresh interval of the information in the browser Status page. The Launcher refresh interval controls the refresh interval of the Management Console Status page.

Related tasks

- [To set up an instance of a Launcher](#)

HTTP Listener connection performance

The HTTP Listener status provides details about the connections between the listener and its external clients and HTTP adapters. The data can indicate the need to adjust connection settings.

The Connection Information provides the total number of active and inactive connections that occurred since the listener was started. The HTTP adapter shuts down an idle connection when it exceeds the timeout limit. You can adjust adapter timeouts with the HTTP adapter **-KEEPALIVE** command and the **IdleHTTP** setting in the tx_install_dir/config.yaml configuration file.

The Connection Time indicates the total amount of time required for the HTTP Listener to connect with the external HTTP client or with the HTTP adapter.

The Management Console displays the connection information in the Status page. You can also display it in a browser by entering the `http://Launcher_host_address:http_port/admin/status` URL. The config.yaml configuration file controls the refresh interval of the information in the browser Status page. The Launcher refresh interval controls the refresh interval of the Management Console Status page.

Related reference

- [KEEPALIVE \(-KEEPALIVE\)](#)

HTTP Listener browser access

You can display map execution status and configure HTTP Listener settings through a local or remote browser, as well as through the Launcher Management Console.

To view or change the settings of a running HTTP Listener, you can use the `http://Launcher_host_address:http_port/admin` URL (for example, `http://localhost:5017/admin`). Changes that you make through the browser take effect immediately, but they only persist when the listener has been started with the **httplsnr** command. Use the Launcher Management Console to change the settings of a listener that was started with the Launcher.

HTTP Listener logging

HTTP Listener trace and audit logs capture details about listener events, socket activity, connections, and HTTP communications.

You can enable HTTP Listener tracing and audit logs through the listener Status browser or Management Console interface, the Launcher Administration utility or command, or the **httplsnr** command.

Note: Binary data is not logged.

The following table describes HTTP Listener logging.

Table 1. Listener logging details

Logging type	Logged information	Log name
Standard	HTTP Listener events such as client requests	<i>listenernname_timestamp_hostname.log</i>
Verbose	Socket activity	<i>listenernname_socket_timestamp_hostname.log</i>
Audit	Details about individual connections and map executions	<i>listenernname_audit_timestamp_hostname.log</i>

Each listener has one audit log. The audit log is in XML format that can be queried with XSLT transformations. The audit log schema is LHL-Audit.xsd, installed in the Transformation Extender installation directory on Windows systems or the /bin directory on UNIX systems.

The location of the log files depends on how you start the listener:

- When you start the listener by using Launcher Administration or the Management Console, the listener creates the log files in the `tx_install_dir/logs` directory.
- When you start the listener from the command line and specify a fully qualified log file name, the listener creates the log files in specified path. If you omit the log file name or specify a relative path to the log file, the listener creates the log in the current working directory.

Supported cryptographic protocols for the HTTP Listener

You can configure the cryptographic strength that the Launcher uses to communicate with its Secure Sockets Layer (SSL) clients. The Launcher supports encryption protocols that range from SSL V2.0 to Transport Layer Security V1.2 (TLS V1.2). Other supported protocols include SSL V3.0, TLS V1.0, and TLS V1.1. Also available is a NIST Special Publication (SP) 800-131A-compliant mode that selects a TLS protocol version based on the cryptographic and certificate strength that is used by the SSL client.

The cryptographic strength that the Launcher uses to communicate with SSL clients depends on the following factors:

- The digital certificates that are obtained.
The SSL and TLS protocols use X.509 certificates to encrypt communication between the Launcher and its SSL clients.
 - When you do not use X.509 certificates, no encryption is in effect in the TCP/IP session layer.
 - SSL encryption without NIST SP 800-131A compliance is in effect when:
 - You obtain digital certificates.
 - You do not install IBM Global Security Kit (GSKit).
- Configuring the SSL security mode (`SECURE_MODE`) setting in the config.yaml configuration file
The [SSL_SERVER] section of the config.yaml configuration file controls the level of security that the Launcher requires to communicate with SSL and TLS clients. The default setting is `SECURE_MODE=0`.

Table 1. Cryptographic strength. This table describes the levels cryptographic strength and the factors that control this strength.

Cryptographic Strength	Digital Certificates	SSL Module in Effect	SSL Security Mode Setting (config.yaml)	SSL Secure Mode Displayed in the HTTP Listener Configuration:
None	undefined	N/A	0	N/A
SSL encryption without NIST SP 800-131A compliance	defined	mercssl	0	unavailable
NIST SP 800-131A and TLS V1.2 are attempted, but the SSL client and SSL server might negotiate a less secure protocol	defined	m4gskssl	0	available
NIST SP 800-131A required	defined	m4gskssl	1	NIST SP 800-131A required
NIST SP 800-131A and TLS V1.2 are required	defined	m4gskssl	2	TLS V1.2 and NIST SP 800-131A required

Map initiation states

To be initiated, a map might require multiple events to occur and multiple resources to be available. The Launcher coordinates these events and resources for maps based on the Launcher settings established during the system creation process in the IFD.

A map is ready to run when all its triggers have occurred and all required resources known to the Launcher are available. After the map is initiated, if a source, target, work file, map file, or log file exists but is in use, the retry values set in `Source>.Retry` and `Target>.Retry` are used to determine whether to continue or complete the processing of that map instance.

- [Initiation pending state](#)
- [Resource pending state](#)
- [Map initiation tip](#)

Initiation pending state

If the map associated with a watch has multiple triggers and at least one trigger occurs, the map is placed in an initiation pending state until all triggers occur. Maps in an initiation pending state are reevaluated each time another trigger occurs.

Resource pending state

The resource pending state refers to source and target resources shared by other map instances known to the Launcher. If you are using the Launcher Status, the resource pending status count is non-zero when all triggers for a map have occurred but the resource manager has not yet released that map for initiation. In this case, at least one source or target updates or replaces a resource used by a map in process.

Map initiation tip

- If a single multi-threaded map is triggered 100 times, the map is read up front during initialization, and kept in memory for all runs. Thus, the map is only read once.

Management Tools

This documentation describes how to configure and use the Management Tools for the Launcher.

The Management Tools are a component of WebSphere® Transformation Extender.

Note: When using the z/OS® local administration interfaces, see "[z/OS Launcher](#)" for more information. The following information does not apply.

- [Overview](#)
- [Unique keys in the Launcher Management Tools](#)
There are some unique key combinations available in the Launcher Management Tools in addition to the standard Java™ navigation keys.
- [HTTP Listener default port settings](#)
- [Automatically-generated log files](#)
The Launcher automatically generates log files in the logs subdirectory of the *install_dir* during processing.
- [Management Console](#)
- [Dynamic adapter tracing](#)
- [Dynamic logging](#)
- [Logging option](#)
- [System and watch status in JSON format](#)
The Launcher heartbeat file summarizes the status of Launcher systems and watches in a JSON-formatted log file that an external program can read to automatically monitor Launcher status.
- [Launcher Monitor](#)
- [Snapshot Viewer](#)

Overview

The Management Tools provide a means to manage and view Launcher processes running in the Launcher. The Management Tools are automatically installed with WebSphere® Transformation Extender and can also be installed separately on multiple machines to allow multiple users to monitor Launcher activity remotely. You can view both Windows and UNIX Launcher activities from one station.

Note: A large number of instances of the Management Tools running will adversely affect the performance of the Launcher.

- [Management Tools for Windows platforms](#)
- [Management Tools for UNIX platforms](#)

Management Tools for Windows platforms

You can install the Management Tools separate from WebSphere® Transformation Extender on your client machines. The Management Tools installation for Windows includes the following applications:

- Management Console ("[Management Console](#)")
- Launcher Monitor ("[Launcher Monitor](#)")
- Snapshot Viewer ("[Snapshot Viewer](#)")

Management Tools for UNIX platforms

You can install the Management Tools separate from WebSphere® Transformation Extender on your client machines. The Management Tools installation for UNIX includes only the Management Console application.

Note: If you are using Native Window Manager and experience symptoms such as no title bar, no insertion point, or no pop-ups, you can work around this by reconfiguring the X Window Manager (Exceed settings).

Unique keys in the Launcher Management Tools

There are some unique key combinations available in the Launcher Management Tools in addition to the standard Java™ navigation keys.

For enhanced accessibility, the following key combinations are available in the Launcher Management Tools applications.

Application name	Navigation description	Key combination
Launcher Administration	To navigate the tabs in the Deployment Directories panel.	Shift+F2
Management Console	To navigate between panes.	Shift+F2
	To navigate the status panel tabs.	Tab

HTTP Listener default port settings

The Launcher Administration, Launcher service, Launcher, and Management Tools applications work collectively and must communicate by means of ports. Each application has default port settings, which can be changed. From the Launcher Administration interface you can configure startup, connection user security, firewall, and deployment options for the Launcher. The following list displays the default port settings for each application, and the interrelated structure of the applications.

- The Launcher process relays monitoring information over port 7001 to the Monitor.

- The Launcher process relays status information over port 7000 to the Management Console.
- The Launcher service sends control information (start, stop, pause, resume) to the Launcher process that is running over port 7002.
- The Management Console commands (start, stop, pause, resume, disconnect) are relayed to the Launcher service over port 5015 (also known as the listening port). Note: On UNIX systems, Management Console commands require both port 5015 and port 5014.
- Communications between the Monitor and the Launcher service are relayed over port 5016 (also known as the listening port).
- The Launcher HTTP Listener communicates with an external client over port 5017 and with Launcher maps over port 5018.

If you run multiple Launcher processes on one computer, they must run on different listening ports and must have different port ranges. If this condition is violated, the Launcher service might not work as intended. Also, export the display to a server that supports X Window systems for that same environment.

When you run the Launcher service through Telnet or on Transformation Extender for z/OS®, ensure that the *DISPLAY* variable points to the machine that you are using. The value of the *DISPLAY* variable can be checked by typing `echo $DISPLAY` at the Telnet prompt and can be set using the `export` command. Otherwise, if the listening port is already in use, an exception is raised instead of the regular message that indicates the listening port is in use. In all cases, an entry in the log file indicates which listening port *number* is in use.

Automatically-generated log files

The Launcher automatically generates log files in the logs subdirectory of the *install_dir* during processing.

- **[Launcher log file](#)**

When it is started, the Launcher service (or daemon) begins to write to the `LauncherLog_timestamp_hostname.log` file. The Launcher Admin process starts Launcher processes to execute launcher systems. Launcher processes write trace messages to the `system_name_timestamp_hostname.log` file. The `LauncherLog_timestamp_hostname.log` file contains session-related logging information, such as initialization settings from the Launcher Administration. The `system_name_timestamp_hostname.log` file contains system processing information such as system command lines, system commands from the Management Console, map execution status, I/O related messages, system status, and any exceptions.

- **[Management Console log file](#)**

The Launcher service (or daemon) generates the `MgmtConsoleLog_timestamp_hostname.log` to log exceptions in communication between the Management Console and the Launcher service.

- **[Communication log file](#)**

The `Communication_timestamp_hostname.log` file is the Management Console log file that the Launcher generates to log exceptions that occur when the Management Console is interacting with the Launcher.

- **[Java Management Tool log file permissions](#)**

Log files that are created by the Java™ Management Tools have the default file permissions `rwx-rw-rw-`. You can change the log file permissions by using the operating system's `umask` command before you run the Java components.

Launcher log file

When it is started, the Launcher service (or daemon) begins to write to the `LauncherLog_timestamp_hostname.log` file. The Launcher Admin process starts Launcher processes to execute launcher systems. Launcher processes write trace messages to the `system_name_timestamp_hostname.log` file. The `LauncherLog_timestamp_hostname.log` file contains session-related logging information, such as initialization settings from the Launcher Administration. The `system_name_timestamp_hostname.log` file contains system processing information such as system command lines, system commands from the Management Console, map execution status, I/O related messages, system status, and any exceptions.

When the `system_name_timestamp_hostname.log` file reaches a maximum size, the Launcher process closes it, appends an index number to log file name (`system_name_timestamp_hostname_index.log`), and begins logging to a new file. You can specify the maximum size of the `system_name_timestamp_hostname.log` file size on the `CircularLogSize` option in the `config.yaml` configuration file. The default maximum size is 100 MB, and the minimum size is 10 MB.

You also can specify the maximum number of `system_name_timestamp_hostname.log` files to keep before the oldest log file is deleted. The `CircularLogFileNum` option in the `config.yaml` file controls the maximum number of log files. The minimum is two log files, and the default maximum is five log files.

The initial `LauncherLog_timestamp_hostname.log` file that is created at Launcher service startup is never deleted. When the Launcher service restarts, it creates a log file with a new timestamp.

- **[UNIX signal-generated codes in the Launcher log file](#)**

Launcher result codes that are higher than 128 occur when the operating system sends a UNIX signal to the Launcher. The value of the Launcher result code is the UNIX signal value plus 128.

Related reference

- [CircularLogSize](#)
- [CircularLogFileNum](#)

UNIX signal-generated codes in the Launcher log file

Launcher result codes that are higher than 128 occur when the operating system sends a UNIX signal to the Launcher. The value of the Launcher result code is the UNIX signal value plus 128.

To determine the UNIX signal value, subtract 128 from the Launcher result code. For example, Launcher result code 139 corresponds to the UNIX signal value 11 (`139 - 128 = 11`). UNIX signal value 11 represents the POSIX SIGSEGV signal.

See your operating system documentation for the list UNIX signals and signal values.

Management Console log file

The Launcher service (or daemon) generates the MgmtConsoleLog_timestamp_hostname.log to log exceptions in communication between the Management Console and the Launcher service.

The log file contains the Management Console and Launcher service session-related logging information including information about Launcher service connection-related failures and Graphical User Interface (GUI) update-related exceptions.

Communication log file

The Communication_timestamp_hostname.log file is the Management Console log file that the Launcher generates to log exceptions that occur when the Management Console is interacting with the Launcher.

The Launcher populates the statistic fields in the Management Console screen directly through Communication Layer that acts as a bridge between the Management Console and the Launcher after the Launcher is running.

Java Management Tool log file permissions

Log files that are created by the Java™ Management Tools have the default file permissions **rw-rw-rw-**. You can change the log file permissions by using the operating system's **umask** command before you run the Java components.

For example, you can use the **umask** command to change the file permissions as follows:

Command	Java Log File Permission
umask 2	rw-rw-r--
umask 22	rw-r--r--

To change the file permissions for all three logs, put the umask command in the DTX_HOME_DIR/bin/common.sh shell script.

Management Console

The Management Console allows you to control and view the status of Launcher processes (.msl system files) running in the Launcher. From the Management Console you can start, stop, and pause a compound system, which is the combination of all .msl files in the deployment directory. See "[Multiple Processes](#)" for additional information when using the multiple-processes feature.

The Management Console provides a variety of figures about the maps being run by the Launcher, such as how long the Launcher has been running, how many maps have been processed, and so on.

- [Features of the Management Console](#)
- [Using the Management Console](#)
- [Monitor option](#)
All systems and compound systems are monitored by default. When you no longer need to monitor a system, you can reduce network traffic by disabling monitoring for that system.
- [Retry option](#)
- [Snapshot option](#)
- [Launcher options](#)
- [Management Console statistics](#)

Features of the Management Console

- Provides internal information for statistical analysis and troubleshooting.
- Runs on all platforms that Transformation Extender supports.
- Displays data in an easy-to-read format.
- Provides debugging information.
- Monitors the current state of an HTTP Listener or a compound system process.
- Starts, stops, and pauses a compound system without stopping the service.
- Provides both real-time and historical statistical information from the compound system running in the Launcher.
- Displays statistics for HTTP Listeners and changes HTTP Listener settings that don't affect listener start up.

Using the Management Console

The Management Console is used to display run-time statistics of the Launcher. You must complete the following steps before using the Management Console.

- [Management Console quick start list](#)
- [Management Console icon descriptions](#)
- [Required steps to perform outside of the Management Console](#)
- [Getting started](#)

- [Controlling systems](#)

Management Console quick start list

After you complete required steps outside of the Management Console, there are three main tasks to complete in the Management Console before you can begin viewing run-time statistics of the Launcher:

To prepare to view Launcher run-time statistics

1. Set up an instance of a Launcher. (See [To set up an instance of a Launcher](#).)
2. Connect to a Launcher. (See [To connect to the Launcher](#).)
3. Start the system or systems. (See [To start the system\(s\)](#).)

The procedures for each task are described in the following topics.

Management Console icon descriptions

The following icons appear in the Management Console navigation pane:

Icon	Description
	Management Console in connection mode
	Management Console in disconnection mode
	Management Console is in retry mode. (The icons periodically fluctuate)
	system stopped
	system running
	statistics
	system paused
	system stopping

Required steps to perform outside of the Management Console

- Generate a system file (**.msl**) and add it to either the **install_dir\systems** directory (which is the default) or to a deployment directory of your choice.
- Using the Launcher Administration interface, do the following activities:
 - Create a user profile with the appropriate access rights.
 - Set up the deployment directory path from where the Launcher will access the system file (**.msl**).
- Start the Launcher.

After you complete the required steps, you are ready to begin using the Management Console. Follow the steps in the following topics to quickly get started using the Management Console.

Getting started

To open the Management Console (Windows)

Select Start > Programs > Transformation Extender > Launcher > Management Console.

To open the Management Console (UNIX)

1. From the command prompt, enter the following command:

```
. setup
```

2. From the same command prompt, enter the following command:

```
mgmtconsole.sh
```

After you open the Management Console, the first thing you need to do is set up an instance of the Launcher that you're going to monitor. After the Launcher is set up, the next step is to connect to the Launcher and begin monitoring.

To set up an instance of a Launcher

Note: You must set up users with the appropriate access rights in the Launcher Administration interface to view systems running in the Management Console. If you add a new user after the Launcher service is running, you must close the Management Console and restart the Launcher before the user is recognized.

1. From the **Launcher** menu, select New to add a Launcher.
The New Launcher window appears.
2. In the **Name** field, type a unique name of up to 20 alphanumeric characters.
Note: Although the name must be unique, it can be arbitrary because it is only a label that appears on its own tab in the Management Console navigation pane.
3. In the **Host Name** field, enter the host name or IP address. An IP address can be in IPv4 (for example, 1.2.3.4) or IPv6 (for example, a:b:c:d:0:1:2:3) format.
4. In the **Port** field, enter the port number for the Launcher. Port 5015 is the default setting.
Note: Each host name must have a separate port number.
5. In the **Refresh Interval (sec)** field, type an interval time in seconds. The minimum value is 1 second, meaning that the Launcher will check for new data every second. The default setting is 5 seconds.
6. Under **Security**, type a **Login Name** and **Password** that permits access to the host. This information must correspond to a user profile set up in the Launcher Administration interface ([Launcher Administration](#)).
7. Click OK.

The New Launcher window closes and the new Launcher name is listed in the navigation pane of the Management Console window.

To connect to the Launcher

Before you create the connection to the Launcher, the Launcher must be running. (See [Starting and Stopping the Launcher](#) for instructions on starting the Launcher service). See [Multiple Processes](#) for additional information when using the multiple-processes feature.

From the navigation pane, right-click on the Launcher and select Connect.

The connection is established. The **CompoundSystem** icon is displayed in the navigation pane for a single process.

The system is labeled **CompoundSystem**, which can represent one or more system files (.msl).

To start the system or systems

From the navigation pane, right-click CompoundSystem and select Start from the context menu.

In the navigation pane, if you expand the **CompoundSystem**, you will see the **Status**, **History**, and **Configuration** icons:

Note: If you see this information upon opening the Management Console, it indicates that the **Automatic startup** option in the Launcher Administration is enabled. The statistics display in the main window.

Note: See [Management Console Statistics](#) for a description of each statistic.

Controlling systems

The Management Console enables you to control the systems that are running. If you right-click on **CompoundSystem** in the navigation pane (see ["Multiple Processes"](#) for additional information when using the multiple-processes feature), a context menu appears that has disconnect and stopping options. The following instructions describe how to disconnect from the Launcher and stop the service on a Microsoft Windows system. See the related reference topics for details about stopping the Launcher on a UNIX system.

Related reference

- [Starting, stopping, pausing, and resuming the Launcher service](#)

To stop an instance of the Launcher

1. From the Management Console navigation pane, right-click **CompoundSystem** and select Stop from the context menu.
The status of the system changes from **Running** to **Stopping** to **Stopped**.
2. Go to the **Launcher** menu and select Disconnect.
The connection is terminated and the **CompoundSystem** disappears from the navigation pane.

To stop the Launcher service (Windows)

1. From your desktop, right-click My Computer and select Manage.
2. From the Computer Management window, expand the **Services and Applications** list and select Services.
3. Right-click **WebSphere Transformation Extender Launcher** and select Stop from the context menu.

Monitor option

All systems and compound systems are monitored by default. When you no longer need to monitor a system, you can reduce network traffic by disabling monitoring for that system.

The Monitor option in the Management Console disables system statistics monitoring for a particular system. Use either of these methods to stop monitoring a system:

- In the Management Console menu, click System and deselect Monitor.

- In the system context menu, click Monitor.

Note: This option is not available when the **CompoundSystem** is stopped.

Retry option

The Management Console has a retry mode that automatically attempts to reconnect to the Launcher service in case it becomes unavailable or shuts down. This eliminates the need to manually monitor the Management Console for possible problems.

When the Launcher service fails, the Management Console attempts to reconnect to the server every 20 seconds. When the Management Console attempts to reconnect, the status bar displays the time in seconds.

To enable retry mode

From the Management Console window, select **LauncherRetry if Disconnected**.
A check mark appears next to **Retry if Disconnected**.

When this option is enabled, the Management Console goes into retry mode upon the shutdown of the Launcher service. To assist with troubleshooting, see the Launcher log file. For information about the Launcher log file, see [Launcher activity log files](#).

Because retry mode is an infinite process, you must stop the operation manually if needed.

To manually stop retry mode

From the Management Console window, select **LauncherStop Retry**.
The Management Console stops trying to connect to the Launcher.

To connect to the Launcher again, you must restart the service or daemon.

Snapshot option

The **Snapshot** option in the Management Console provides a means for saving a "snapshot" of Launcher information to a file in XML format. A "snapshot" of Launcher information refers to capturing a moment in time of status, historical, and configuration information that the Management Console provides about a **CompoundSystem** (see ["Multiple Processes"](#) for additional information when using the multiple-processes feature) running under a Launcher.

To create a snapshot file

Note: This option is not available when the **CompoundSystem** is stopped.

1. From the Management Console, select **System > Snapshot**.
The Save As window opens.
2. Specify the name of the XML file to which the Launcher **CompoundSystem** information will be stored and click **Save**.
The file is saved in XML format.

You can accomplish the same task on the command line. For information about using the Snapshot command line options, see [Snapshot commands](#).

Launcher options

In the **Options** list of the Options window, when you select Launchers, the configuration options described below are displayed. These options allow you to manually configure and override connection information for each Launcher connection in the Management Console.

- [Override Configuration options](#)

Override configuration

You can manually configure and override connection information for each Launcher connection in the Management Console. Typically, the Management Console relays control commands (start, stop, pause, resume, disconnect) to a Java-based Launcher service, which in turn executes the commands to the process running. However, you can manually override the typical process so that the Management Console bypasses the Java-based service and communicates directly with the process running. In this case, the Management Console can relay only limited control commands (stop, pause, resume) to the process running. The Launcher itself must be manually started from the CPU where it resides.

The following list describes the communication process that occurs when you use the override configuration option. The port numbers that are referenced are the default numbers.

- Control information including stop, pause, and resume commands, are communicated between the Launcher process that is running and the Management Console over port 7002.
- Status information is communicated between the Launcher process that is running and the Management Console over port 7000.
- Monitoring information is communicated between the Launcher process that is running and the Launcher Monitor over port 7001.

To access override configuration options:

1. In the Management Console, select **Tools > Options**.
The Options window opens.

2. From the **Options** list, select Launchers.
The override configuration options appear.

Override Configuration options

The Launcher override configuration options in the Options dialog are described in the following table:

Override configuration options	Value
Launchers	This field lists each Launcher connection that is configured in the Management Console. You can set override values for each Launcher listed.
Override Configuration	When this option is enabled, you can manually enter connection information for the Launcher selected that overrides the original connection information.
Host Name	Enter the host name or IP address of where the Launcher service will run.
Status Port	Enter the status port number, or accept 7000, which is the default port. This port provides status information about the process running to the Management Console.
Control Port	Enter the control port number, or accept 7000, which is the default port. This port relays stop, pause, and resume information from the process running back to the Launcher service.

Management Console statistics

The statistical information displayed in the Management Console is refreshed periodically based on the refresh interval that you set when creating a new Launcher. Statistical information is derived directly from the process that the Launcher service is running.

The Management Console displays information for systems and HTTP listeners.

- [**System statistics**](#)
The Management Console displays summary, status, history, and configuration information about a running Launcher system.
- [**Launcher HTTP Listener summary, status, and configuration**](#)
The Launcher HTTP Listener provides two-way communication between external HTTP clients and maps in the Launcher.

System statistics

The Management Console displays summary, status, history, and configuration information about a running Launcher system.

- [**Summary statistics**](#)
- [**Status statistics**](#)
- [**Historical information**](#)
- [**Configuration information**](#)

Summary statistics

From the Management Console, statistical data is displayed on the Summary tab in the main window of the Management Console. The statistics are grouped into the following three areas by default: **Performance**, **Status**, and **Historical**.

- [**Performance**](#)
- [**Status**](#)
- [**Historical**](#)

Performance

Statistic	Description
Memory Usage	Current amount of memory usage. (Supported on Windows platforms only.)
CPU Usage	Current percent of total CPU usage. (Supported on Windows platforms only.)

Status

Statistic	Description
System Status	Status of the Launcher process (.msl system file). Options include: Starting, Running, Stopping, Stopped, and Paused.
Active Component Maps	

Active Listeners Up	Current number of component map instances running
Active Listeners Down	Number of listeners running
Active Connections	Number of active connections
Start Time	Time that the Launcher started
Success Time	Cumulative amount of time for maps, which are successful
Failure Time	Cumulative amount of time for maps that failed
Up Time	Total time that the Launcher has been running
Pending Initialization	Number of maps currently waiting to begin execution
Pending Initialization Maximum	Maximum number of maps that have waited to begin execution at any one time
Pending Resource	Number of maps currently waiting for resources, such as a shared data target
Pending Connection	Number of maps waiting to connect
Pending Total Maps	Sum of Pending Initialization, Pending Resource, and Pending Connect

Historical

Statistic	
Description	
History Successes	Number of maps that have succeeded since the Launcher started
History Failures	Number of maps that have failed since the Launcher started
History Total Maps	Total number of all maps that have been executed
History Connection Failures	Total number of connections that failed
History Deadlocks Detected	Number of map failures resulting from deadlocks
History Function Failures	Number of failures in RUN, GET, PUT, DBLOOKUP, and DBQUERY

Status statistics

From the Status tab, you can view information about active components, pending components, and adapter connections.

Active Component tab

Column	
Description	
Component	Component name
Map	Name of compiled map file (.mmc)
System	System name
State	The state of the Launcher. Options include:
	<ul style="list-style-type: none"> • getting input • validating input • building input • running map • GET/PUT adapter • ending output
Card	The card number and I/O indicator
Resource	Adapter command used for this card
	<ul style="list-style-type: none"> • Pending • Adapter connections

Pending

Column	Description
Component	Component name
Map	Name of compiled map file (.mmc)
System	System name
State	Initialization, connect, or resource pending
Card	For connect and resource pending state, which card is pending
Reason	For resource pending state, reason why card is pending
Resource Type	Type of resource or adapter alias for this card
Resource Name	Name of resource (adapter command) for this card

Adapter connections

Column	Description
Adapter	Adapter type, such as database or FTP. Default represents the global default settings.
Open	Number of connections currently open
Active	Number of open connections being used in a map
Idle	Number of open connections not in use
Pending	Number of pending connection requests

Historical information

From the Historical tab, you can view information about:

- [Map Failures](#)
- [Adapter Connections](#)
- [Map Function Failures](#)

The number of rows of data varies dynamically.

- [Map failures](#)
- [Adapter connections](#)
- [Map function failures](#)

Map failures

Column	Description
Component	Component name
Map	Name of compiled map file (.mmc)
System	Name of system containing map failure
Card	Card number and I/O indicator that caused the error
Card Error	Return code and error message from adapter
Map Error	Return code and error message from map

Adapter connections

Column	Description
Adapter	Adapter type, such as database or FTP. Default represents global default settings.
Requests	Number of connection requests
New	Number of requests that required a new connection
Reused	The number of requests that reused an existing connection
Successes	Number of successful operations (GET or PUT) on the adapter
Failures	Number of failing operations (GET or PUT) on the adapter

Map function failures

Column	Description
Component	Name of the map component where the function is contained
Function Name	Name of function that failed
Argument	Map name for RUN; adapter name for GET, PUT, DBLOOKUP, and DBQUERY
Reason	Reason for function failure

Configuration information

From the Configuration tab, you can view global configuration parameters used by the Launcher.

- [System tab](#)
- [Adapter connections tab](#)

System tab

Column	Description
Maximum Concurrent Maps	Maximum number of maps that can run at the same time
Maximum Concurrent Maps Per Watch	Maximum number of maps running at the same time for each watch
Pending Initialization High	A user-defined static amount that serves as the threshold for the number of adapter listeners and time events pending initialization. After this amount is reached, maps can no longer trigger until the Pending Initialization Low amount is reached.
Pending Initialization Low	After the Pending Initialization High amount is reached, the number of items pending must reach this amount before maps can begin triggering again.

Adapter connections tab

Column	Description
Adapter	Adapter type, such as database or FTP. Default represents global default settings.
Idle Time	The period of time after which an idle connection is expected to be disconnected
Keep Time	How long an idle connection is expected to remain valid. Do not use this setting with the FTP adapter.
Keep Minimum	The minimum number of connections allowed
Advisory Limit	The maximum number of connections, but can be exceeded if necessary
Mandatory Limit	The maximum number of connections allowed

Launcher HTTP Listener summary, status, and configuration

The Launcher HTTP Listener provides two-way communication between external HTTP clients and maps in the Launcher.

Overview

To view listener details, click `Launcher > Listeners` in the Management Console and select the HTTP Listeners that you want to monitor. You can display the HTTP Listener startup settings, watch and connection status, and configure HTTP Listener settings and logging options.

Summary page

The Summary page displays the HTTP Listener startup settings defined when you create the listener. Use the Launcher Administration utility to edit these settings.

The Summary is refreshed at the refresh interval that is configured in the `config.yaml` configuration file.

Status page

The Status page displays response totals, connection data, and information about URL watches that you can use to tune the system performance of the Launcher. Also, you can enable and disable trace, socket, and audit logging on the Status.

By default, the Status is refreshed at the Launcher refresh interval. To enable manual refresh, edit the `tx_install_dir\mgmtconsole.properties` file and set `listener.status.manual=1`.

Configuration page

In the Configuration page, you can edit the HTTP Listener settings that are not required for starting the listener. The Configuration is only refreshed when you update the HTTP Listener settings.

HTTP Port

The port that the listener uses to communicate with external clients. The listener monitors the HTTP port for incoming HTTP requests and responds to the external client through the HTTP port. The HTTP port for each listener must be unique. If you configure multiple listeners with the same HTTP port value, only one listener starts. The default HTTP port is 5017.

Launcher Port

The HTTP Listener port used in the URL of HTTP adapters in Launcher systems that are going to process requests handled by this Launcher. The default Launcher port is 5017.

Launcher Policy

The method to use to load-balance HTTP requests among Launchers when there are multiple Launchers on one listener or multiple watches on the same Launcher.

- FCFS (first come first served)
- RR (round robin)
- RND (random)

Response Timeout

Amount of time in minutes and seconds (*mm:ss*) that the HTTP Listener waits for a response from the Launcher. The default value is 90 seconds (01:30).

Launcher User

User ID that the Launcher uses to connect to the HTTP Listener.

Launcher Password

Password that authorizes the User ID for the Launcher.

Admin URL (this form)

URL to access Launcher Administration form.

Admin Page (this form)

Source file name of Launcher Administration form.

Admin User

User ID to authenticate connection to Launcher Administration.

Admin Password

The password for the **Admin User** ID.

Port Monitor Cycle Time

Length of time to listen on a socket before checking for control messages. The format is *mm:ss* (minutes:seconds), where *mm:* is optional. The default value is 45 seconds.

Important: This value affects the time it takes to stop the service.

Launcher Time Limit

Time to wait for the Launcher connection when a request is received. The format is *mm:ss* (minutes:seconds), where *mm:* is optional. The default value is 5 seconds.

Client Retry-After Time

Notifies HTTP clients of the amount of time to wait before resending a request that has not been answered by the Launcher. The default value is 45 seconds.

Authenticate Clients

Globally determines whether browser requests should be authenticated. The default is authentication disabled.

User Administration

Click User Administration to add, modify or delete users for the Launcher.

SSL Status

Indicates whether SSL is available.

SSL Authentication

Indicates whether SSL communication with HTTP clients is enabled.

SSL Secure Mode

Indicates the required level of security for communication between the HTTP Listener and HTTP clients.

None

X.509 certificates are not defined. No encryption is in effect in the TCP/IP session layer.

Unavailable

SSL encryption without NIST SP 800-131A compliance is in effect.

Available

- NIST SP 800-131A and TLS V1.2 are attempted, but the SSL client and SSL server can negotiate a less secure protocol.
- NIST SP 800-131A required
 - Communication between the Launcher and SSL clients must comply with the NIST SP 800-131A standard.
- NIST SP 800-131A and TLS V1.2 required
 - Communication between the Launcher and SSL clients must use the TLS V1.2 protocol and comply with the NIST SP 800-131A standard.

HTTP Certificate

The name in the key store of the certificate from the Certificate Signing Authority that enables SSL communication between the HTTP Listener and external HTTP clients. The key store is set in the config.yaml file in the installation directory.

Launcher Certificate

The name in the key store of the certificate that enables SSL communication between the HTTP listener and the Launcher. The key store is set in the config.yaml file in the installation directory.

Related concepts

- [HTTP Listener map performance and throughput](#)
- [HTTP Listener connection performance](#)
- [Supported cryptographic protocols for the HTTP Listener](#)

Related reference

- [HTTP Listener startup configuration](#)

Dynamic adapter tracing

Dynamic adapter tracing enables you to dynamically turn on and off adapter tracing for a map running under the Launcher. Through the configuration pane of the Management Console, you can configure adapter tracing dynamically. When you turn adapter tracing on, the result is that the map will write all the trace information to a new trace file or append the trace information to an existing file.

You do not need to enable adapter tracing in the adapter command line of your map to optimize it. If you encounter an issue with the adapter, you can dynamically turn adapter tracing on in the Management Console.

- [Using the dynamic adapter trace feature](#)
- [Enabling dynamic adapter tracing](#)
- [Configuration pane](#)
- [Output files](#)

Related reference

- [MapAdapterTraceOff](#)

Using the dynamic adapter trace feature

This feature is helpful for resolving adapter issues without having to:

- bring the Launcher down to add tracing to the adapter command lines
- having to recompile and redeploy the maps
- having to restart the Launcher

The following procedure assumes you have connected to a Launcher from the Management Console and have selected the Compound System that appears in the navigation pane.

Refer to ["Multiple Processes"](#) for additional information when using the multiple-processes feature.

To use the dynamic adapter trace feature:

- Start the Launcher.

When the Launcher starts, it will initialize all trace settings to default values as configured on the adapter command line. Tracing will be done for all watches according to the adapter command line settings.

The MapAdapterTraceOff INI setting is used to override the default tracing at the startup of the Launcher.

When the MapAdapterTraceOff INI setting is commented or 0, it indicates that the default tracing should be used at the startup of the Launcher.

When the MapAdapterTraceOff INI setting is 1, it indicates that tracing is turned off for all components at the startup of the Launcher, regardless of the command line settings.

- Enable the adapter tracing feature in the Management Console.
- Configure adapter tracing dynamically through the configuration pane.
- Modify adapter tracing configurations.

Note: When you modify the adapter tracing configurations after a Launcher has started, the changes will take affect in the next map that is run for that component.

Enabling dynamic adapter tracing

The configuration pane in the Management Console is used to configure the settings for enabling dynamic adapter tracing. This procedure assumes that the Launcher is running and there is a Launcher system (.msl) file in the deployment directory.

To enable dynamic adapter tracing

1. After highlighting a Launcher system (.msl) file, from the menu bar in the Management Console, select System.
2. Select Trace & Log from the pull-down menu.
A pane opens in which you can configure the dynamic adapter tracing specifications for each map component in each system separately.

Configuration pane

When you have enabled dynamic adapter tracing through the System menu option in the Management Console, you use the configuration pane to configure settings for a watch. For each watch, you can:

- turn off adapter tracing, which overrides the adapter command line that was defined in a map through the Map Designer
- turn on adapter tracing for all adapters including child RUN maps
- use the default tracing configurations in the adapter command line that was defined in a map through the Map Designer

Selecting Trace from the Config pull-down menu in the configuration pane will present the configuration pane elements specific for dynamic adapter tracing.

- [Configuration pane elements](#)
- [Using wildcards](#)

Configuration pane elements

The Configuration pane in the Management Console used for dynamic adapter tracing consists of the elements required to enable tracing and configure the trace specifications. The following table lists the components and provides a description of their use.

Configuration Pane Elements

Description	
Config	The Config pane contains a pull-down menu where you can select either of the following two configuration options: <ul style="list-style-type: none">• Trace The Trace configuration option indicates that you want to configure dynamic adapter tracing. When you select Trace, the trace configuration for the selected Launcher will appear.• Logging The Logging configuration option indicates that you want to configure dynamic logging.
Systems	The Systems pane contains a pull-down menu where you can select the path and system file for the Launcher process.
Components	The Components pane contains all the map components in the Launcher system, which you can configure individually. It is the list of all top-level map components running in that Launcher process for the selected system. When you select a map, only the trace settings for that selected map will appear. When you select a different map, a different set of trace settings will appear. All RUN maps associated with that component will use the same settings. Only information for a running system will be available for configuration, not for a stopped system.
Settings	The Settings pane comprises the Trace and Action panes, which are used to configure trace settings for each of the components in the Components pane.
Trace	The Trace pane contains radio buttons for the following three adapter trace setting options: <ul style="list-style-type: none">• Default The Default adapter trace setting option indicates that the Launcher will use the settings on the adapter command line for a map configured in the Map Designer or the Integration Flow Designer. The Default option is the default adapter trace setting.• ON The ON adapter trace setting option enables adapter tracing for all adapters for a map. This setting overrides any setting on the adapter command line for a map configured in the Map Designer or the Integration Flow Designer.• OFF The OFF adapter trace setting option disables adapter tracing for any adapter. This setting overrides any setting on the adapter command line for a map configured in the Map Designer or the Integration Flow Designer.
Action	The Action pane contains two sub-elements related to the output adapter trace file: Adapter trace file option Allows you to configure how you want the trace content to be written.

There are two radio buttons for the adapter trace file options:

- Create
When the Create option is selected, the map will write all the trace information to a new trace file.
- Append
When the Append option is selected, the map will write all the trace information to a new trace file or append the trace information to an existing file.

File Name

Allows you to configure the trace file name and location.

See "[Using Wildcards](#)" for information about using wildcards in your trace file name.

Apply

The Apply button is used to refresh the Launcher with the settings associated with the component.

The adapter trace configuration changes for the component will be passed to the Launcher process, and will take effect when the next map is initiated for the configured component.

Note: These dynamic adapter trace configurations are not saved anywhere.

Using wildcards

Wildcards in the trace file name can be used to reference map source events. They are specified in the configuration pane of the Management Console when using the dynamic adapter tracing feature.

How wildcards are used

- Used in the trace file name based on source events that also use wildcards.
- Can not be used if the source event does not use wildcards.
- Allow for unique instances of adapter trace files.

Output files

The output of using the dynamic adapter trace feature is a file. It contains the following sections:

- Header
- Trace results for a map instance
- Trailer

The header and trailer sections are written to the trace file before and after each map instance including RUN maps. They both contain the following information:

- map name including full path
- date and time of executed action type
- map instance count, which represents the cumulative number of times the map has been executed

Note: RUN maps will be nested within top map entries to enhance your ability to see the sequence of events.

See the "[File Name:](#)" within the configuration pane element for more detailed information about naming the adapter trace output file.

Dynamic logging

Dynamic logging allows you to configure a Launcher process for logging while the Launcher is running. You can configure the following logging entries in the Management Console:

Logging Entries

- Launcher (ES)
- Resource Manager (RM)
- Connection Manager (CM)

Each logging entry is categorized by the following severity classes and debug components listed from lowest impact to highest:

Logging Categories

- Trace
- Debug
- Info
- Warn
- Error
- Fatal

Without having to stop the Launcher, you can turn each of these logging components (**Trace**, **Debug**, **Info**, **Warn**, **Error** and **Fatal**) for each of the logging entries (**Launcher**, **Resource Manager**, and **Connection Manager**) on and off individually for a Launcher process that you select to configure.

For example, you can run the Launcher with **INFO** settings to track the status of maps. If one of the maps connecting to a database starts to fail, you can dynamically enable the setting (turn on) for the **DEBUG** and **ERROR** components under the **Connection Manager** logging entry for a Launcher process that you have selected. Now you can track what the database connection is reporting without having to stop the Launcher, configure the Launcher with different debug settings, and then restart it.

- [Using the dynamic logging feature](#)
- [Enabling dynamic logging](#)
- [Configuration pane](#)
- [Output files](#)
- [Logging messages for Trace](#)
- [Logging messages for Debug](#)
- [Logging messages for Info](#)
- [Logging messages for Warning](#)
- [Logging messages for Error](#)
- [Logging messages for Fatal](#)

Using the dynamic logging feature

This feature is helpful for resolving Launcher issues *without* having to:

- bring the Launcher down to enable the logging feature
- reconfigure the Launcher with different logging settings
- restart the Launcher

The following procedure assumes you have connected to a Launcher from the Management Console and have selected the Compound System that appears in the navigation pane.

See "[Multiple Processes](#)" for additional information when using the multiple-processes feature.

To use the dynamic logging feature

1. Start the Launcher.
 - When the Launcher starts, it will use the logging category settings configured in the **INI (config.yaml)** file under each logging entry section (**Launcher**, **Resource Manager**, and **Connection Manager**) to determine which severity levels and components to debug.
 - The following logging categories in the **INI** file will be configured to 0 to turn that logging category off or 1 to turn that logging category on:
 - LogTrace
 - LogDebug
 - LogInfo
 - LogWarning
 - LogError
 - LogFatal
2. Enable the logging feature in the Management Console.

Enabling dynamic logging

The configuration pane in the Management Console is used to configure the settings for enabling dynamic logging. This procedure assumes that the Launcher is running and there is a Launcher system file (**.msl**) in the deployment directory.

To enable dynamic logging

1. After highlighting a Launcher system (**.msl** file), from the menu bar in the Management Console, select System.
 2. Select Trace & Log from the pull-down menu.
- A pane opens in which you can configure the dynamic logging specifications for each logging category component for each logging entry.

These logging category settings are not saved anywhere. Changes you make to these logging categories in the Management Console do not update the INI file.

Logging begins recording upon starting the system.

Configuration pane

When you have enabled dynamic logging through the System menu option in the Management Console, you use the configuration pane to configure logging settings for a Launcher process.

Selecting Logging from the Config pull-down menu in the configuration pane will present the configuration pane elements specific for dynamic logging.

- [Configuration pane elements](#)

Configuration pane elements

The configuration pane in the Management Console used for dynamic logging consists of the elements required for you to specify logging and configure the logging specifications.

The following table lists the components and provides a description of their use.

Configuration Pane Elements

Description
Config The Config pane contains a pull-down menu where you can select either of the following two configuration options: <ul style="list-style-type: none">• Trace The Trace configuration option indicates that you want to configure dynamic adapter tracing.• Logging The Logging configuration option indicates that you want to configure dynamic logging. When you select Logging, the Logging configuration for the selected Launcher will appear.
Launcher The Launcher pane contains the list of severity levels and components (logging categories) for the Launcher logging entry that you can enable or disable for the logging of the Launcher process you selected.
Resource Manager The Resource Manager pane contains the list of severity levels and components (logging categories) for the Resource Manager logging entry that you can enable or disable for the logging of the Launcher process you selected.
Connection Manager The Connection Manager pane contains the list of severity levels and components (logging categories) for the Connection Manager logging entry that you can enable or disable for the logging of the Launcher process you selected.
Apply The Apply button is used to: <ul style="list-style-type: none">• trigger the Management Console to pass these logging configuration changes to the Launcher process• refresh the Launcher with the settings associated with each logging category for each logging entry As a result, the Launcher will notify the Management Console to synchronize the logging settings.
Clear All The Clear All button is used to: <ul style="list-style-type: none">• disable all the check boxes under the three logging entries: Launcher, Resource Manager, and Connection Manager The Clear All button is grayed out when there are no check boxes enabled.

Output files

The output of using the dynamic logging feature is a file. It contains the log entries. A log entry is a single line entry for each event that has been logged and is comprised of components in the following format:

```
<msgid><date/time><threadid><instance><msgtext><extra>
```

Log Entry
Component
Description

msgid unique identifier of the log entry
date/time date and time of the executed action type
threadid identifier that distinguishes each map that is running concurrently
instance instance number of a map that is running
msgtext log entry
extra area for additional information (optional)

Logging messages for Trace

This documentation presents the logging messages for the Trace logging category for each logging entry (Launcher, Resource Manager, Connection Manager) listed by each message category (01xx, 03xx, 05xx).

- [Launcher trace 01xx](#)
- [Resource Manager trace 03xx](#)
- [Connection Manager trace 05xx](#)

Launcher trace 01xx

Message ID
DTXLN-T-0101

```

Launcher trace log enabled
DTXLN-T-0102
    Launcher trace log disabled
DTXLN-T-0103
    Starting Launcher <pid>
DTXLN-T-0104
    Loading MSL <msl_name>
DTXLN-T-0105
    Before Init Resource Manager
DTXLN-T-0106
    After Init Resource Manager
DTXLN-T-0107
    Before Init Connection Manager
DTXLN-T-0108
    After Init Connection Manager
DTXLN-T-0109
    Init socket <socket_type>
DTXLN-T-0110
    Map Start <map_instance_number>
DTXLN-T-0111
    Map End <map_instance_number>
DTXLN-T-0112
    Before resource pending
DTXLN-T-0113
    After resource pending
DTXLN-T-0114
    Before adapter call <adapter>
DTXLN-T-0115
    Before adapter call <adapter>
DTXLN-T-0116
    RUN map call <run_map>
DTXLN-T-0117
    Before destroy table
DTXLN-T-0118
    After destroy table
DTXLN-T-0119
    Before RMDestroy
DTXLN-T-0120
    After RMDestroy
DTXLN-T-0121
    Pending thread added
DTXLN-T-0122
    Terminating --> last message to log

```

Resource Manager trace 03xx

Message ID	Message
DTX-RM-T-0301	Resource Manager trace log enabled
DTX-RM-T-0302	Resource Manager trace log disabled
DTX-RM-T-0303	Init global resource table
DTX-RM-T-0304	Init local resource table
DTX-RM-T-0305	rm-realloc called start
DTX-RM-T-0306	rm-realloc called end
DTX-RM-T-0307	addrmentry blocking <ID>
DTX-RM-T-0308	maddrunentry suspending <ID>
DTX-RM-T-0309	maddrunentry resuming <ID>
DTX-RM-T-0310	removermentry - can't resume <ID>
DTX-RM-T-0311	Entering rmdestroy
DTX-RM-T-0312	Leaving rmdestroy

Connection Manager trace 05xx

Message ID

Message

DTX-CM-T-0501
Connection Manager trace log enabled

DTX-CM-T-0502
Connection Manager trace log disabled

DTX-CM-T-0503
Entering destroy table

DTX-CM-T-0504
Destroy table - stopping listeners

DTX-CM-T-0505
Destroy table - closing connections

DTX-CM-T-0506
Destroy table - uploading adapters

DTX-CM-T-0507
Leaving destroy table

DTX-CM-T-0508
Entering SAM entry <parameters>

DTX-CM-T-0509
Leaving SAM entry <params>

DTX-CM-T-0510
Begin rm worker thread

DTX-CM-T-0511
End rm worker thread

DTX-CM-T-0512
RMParseAdapterCommand <parameters>

DTX-CM-T-0513
Begin middleboot thread

DTX-CM-T-0514
End middleboot thread

DTX-CM-T-0515
RmiddleBoot active

DTX-CM-T-0516
FireEvent - calling launchadapter <x>

DTX-CM-T-0517
FireEvent returned <x>

DTX-CM-T-0518
Begin rmlistenerthread worker

DTX-CM-T-0519
End rmlistenerthread worker

DTX-CM-T-0520
RMInittriggers - maps = <amount>

Logging messages for Debug

This documentation presents the logging messages for the Debug logging category for each logging entry (Launcher, Resource Manager, Connection Manager) listed by each message category (11xx, 13xx, 15xx).

- [Launcher debug 11xx](#)
- [Resource Manager debug 13xx](#)
- [Connection Manager debug 15xx](#)

Launcher debug 11xx

Message ID

Message

DTXLN-D-1101
Launcher debug log enabled

DTXLN-D-1102
Launcher debug log disabled

DTXLN-D-1103
Starting Launcher <pid>

DTXLN-D-1104
Loading MSL <msl_name>

DTXLN-D-1105
Before init Resource Manager

DTXLN-D-1106
After init Resource Manager

DTXLN-D-1107
Before init Connection Manager

DTXLN-D-1108
After init Connection Manager

DTXLN-D-1109
Init socket <socket_type>

```
DTXLN-D-1110
    Map Start <map_instance_number>
DTXLN-D-1111
    Map Start <map_instance_number>
DTXLN-D-1112
    Before resource pending
DTXLN-D-1113
    After resource pending
DTXLN-D-1114
    Before adapter call <adapter>
```

Resource Manager debug 13xx

Message ID	
Message	
DTX-RM-D-1301	Resource Manager debug log enabled
DTX-RM-D-1302	Resource Manager debug log disabled
DTX-RM-D-1303	PID list exists
DTX-RM-D-1304	PID list doesn't exist
DTX-RM-D-1305	SMBASE <address + length>
DTX-RM-D-1306	PRMH <address + length>
DTX-RM-D-1307	SIG <address + length>
DTX-RM-D-1308	PRMT <address + length>
DTX-RM-D-1309	PCT <address + length>
DTX-RM-D-1310	PFTG <address + length>
DTX-RM-D-1311	PNBB <address + length>
DTX-RM-D-1312	RMAaddrunentry in shutdown
DTX-RM-D-1313	Thread enabled due to shutdown
DTX-RM-D-1314	firstRM is null when count >1
DTX-RM-D-1315	RMremoverunentry in shutdown

Connection Manager debug 15xx

Message ID	
Message	
DTX-CM-D-1501	Connection Manager debug log enabled
DTX-CM-D-1502	Connection Manager debug log disabled
DTX-CM-D-1503	Queue manager: <message>
DTX-CM-D-1504	Session manager: <message>
DTX-CM-D-1505	Rmcheckadapter object <adapter>
DTX-CM-D-1506	OpenConnection <initial_number>
DTX-CM-D-1507	OpenConnection - new conn opened
DTX-CM-D-1508	OpenConnection - existing conn refused
DTX-CM-D-1509	OpenConnection - <final_number>
DTX-CM-D-1510	OpenConnection - GTXOPEN
DTX-CM-D-1511	CloseConnection <number>
DTX-CM-D-1512	CloseConnection - GTXCLOSE

```

DTX-CM-D-1513
    FireEvent - elapsed <value>
DTX-CM-D-1514
    FindEvent stats <value>
DTX-CM-D-1515
    RMListenerThread - waiting <value>
DTX-CM-D-1516
    RMListenerThread - waited <value>
DTX-CM-D-1517
    RMListenerThread - listenmsgs <values>
DTX-CM-D-1518
    RMListenerThread - found <value>
DTX-CM-D-1519
    RMListenerThread - elapsed stats <value>
DTX-CM-D-1520
    RMListenerThread - stats2 <values>

```

Logging messages for Info

This documentation presents the logging messages for the Info logging category for each logging entry (Launcher, Resource Manager, Connection Manager) listed by each message category (21xx, 23xx, 25xx).

- [Launcher info 21xx](#)
 - [Resource Manager info 23xx](#)
 - [Connection Manager info 25xx](#)
-

Launcher info 21xx

Message ID	Message
DTXLN-I-2101	Launcher info log enabled
DTXLN-I-2102	Launcher info log disabled
DTXLN-I-2103	Starting Launcher <version>
DTXLN-I-2104	Command Line entered <value>
DTXLN-I-2105	INI settings <value>
DTXLN-I-2106	Ports <value>
DTXLN-I-2107	Gethostname <value>
DTXLN-I-2108	Resource config file <value>
DTXLN-I-2109	MSL Name - version <value>
DTXLN-I-2110	MMC Name <mapname>
DTXLN-I-2111	MMC Sources / targets <value>
DTXLN-I-2112	MMC settings <value>
DTXLN-I-2113	Watch settings <value>
DTXLN-I-2114	Adapter trigger found (init) -- <value>
DTXLN-I-2115	M4file journaling enabled
DTXLN-I-2116	Socket opened <type>
DTXLN-I-2117	Missing input for multi-source event <value>
DTXLN-I-2118	Got all triggers -- missing time event
DTXLN-I-2119	Got all triggers -- single threaded
DTXLN-I-2120	Mapping thread created
DTXLN-I-2121	Map Start <value>
DTXLN-I-2122	Map End / outcome <values>

```

DTXLN-I-2123
    Hi-Lo pause
DTXLN-I-2124
    Hi-Lo continue
DTXLN-I-2125
    Shutdown Launcher
DTXLN-I-2126
    Pause Launcher
DTXLN-I-2127
    Continue Launcher
DTXLN-I-2128
    Got STOP
DTXLN-I-2129
    Hit maximum mapping thread
DTXLN-I-2130
    Status socket accepted
DTXLN-I-2131
    Status socket closed
DTXLN-I-2132
    Monitor socket accepted
DTXLN-I-2133
    Monitor socket closed
DTXLN-I-2134
    Control socket accepted
DTXLN-I-2135
    Control socket closed
DTXLN-I-2136
    Control socket command <values>
DTXLN-I-2137
    InitPendingHigh met <value>
DTXLN-I-2138
    InitPendingLow met <value>
DTXLN-I-2139
    Terminating - last message to log

```

Resource Manager info 23xx

Message ID	Message
DTX-RM-I-2301	Resource Manager info log enabled
DTX-RM-I-2302	Resource Manager info log disabled
DTX-RM-I-2303	INI settings <value>
DTX-RM-I-2304	Table size <value>
DTX-RM-I-2305	Central logging enabled
DTX-RM-I-2306	Local logging enabled
DTX-RM-I-2307	SHM use count
DTX-RM-I-2308	Launcher use count
DTX-RM-I-2309	Pointer size <value>
DTX-RM-I-2310	Deadlock avoided - addrmentry
DTX-RM-I-2311	RMTABLE entries/size <value>
DTX-RM-I-2312	COMMANDTABLE entries/size <value>
DTX-RM-I-2313	FREETOGO entries/size <value>
DTX-RM-I-2314	NBB size <value>
DTX-RM-I-2315	Average command line size
DTX-RM-I-2316	FTG resume in rmworker thread
DTX-RM-I-2317	RMrealloc called - start <value>
DTX-RM-I-2318	RMrealloc called - end <value>
DTX-RM-I-2319	Entering RMIDestroy

Connection Manager info 25xx

Message ID	Message
DTX-CM-I-2501	Connection Manager info log enabled
DTX-CM-I-2502	Connection Manager info log disabled
DTX-CM-I-2503	INI settings
DTX-CM-I-2504	Destroy - waiting for threads
DTX-CM-I-2505	OpenConnections - shutdown time
DTX-CM-I-2506	RMIListenerThread - pause
DTX-CM-I-2507	RMIListenerThread - continue
DTX-CM-I-2508	RMIListenerThread - stop
DTX-CM-I-2509	RMIListenerThread - start

Logging messages for Warning

This documentation presents the logging messages for the Warning logging category for each logging entry (Launcher, Resource Manager, Connection Manager) listed by each message category (31xx, 33xx, 35xx).

- [Launcher warning 31xx](#)
 - [Resource Manager warning 33xx](#)
 - [Connection Manager warning 35xx](#)
-

Launcher warning 31xx

Message ID	Message
DTXLN-W-3101	Launcher warning log enabled
DTXLN-W-3102	Launcher warning log disabled
DTXLN-W-3103	Map Warning <mapname / warning>
DTXLN-W-3104	Socket disconnect <socket>
DTXLN-W-3105	Unable to load resource config file

Resource Manager warning 33xx

Message ID	Message
DTX-RM-W-3301	Resource Manager warning log enabled
DTX-RM-W-3302	Resource Manager warning log disabled
DTX-RM-W-3303	Failed to get SHM on create
DTX-RM-W-3304	Failed to get SEM on create
DTX-RM-W-3305	Create mismatch
DTX-RM-W-3306	Table size requested vs. allocated <values>
DTX-RM-W-3307	Reached ten second lock
DTX-RM-W-3308	

Connection Manager warning 35xx

Message ID	
Message	
DTX-CM-W-3501	Connection Manager warning log enabled
DTX-CM-W-3502	Connection Manager warning log disabled
DTX-CM-W-3503	DestroyTable - table already destroyed
DTX-CM-W-3504	DestroyTable - table in use
DTX-CM-W-3505	OpenConnection - smcomparerm fail
DTX-CM-W-3506	OpenConnection - all connections busy
DTX-CM-W-3507	OpenConnection - couldn't open conn
DTX-CM-W-3508	CloseConnection - invalid conn type
DTX-CM-W-3509	CloseConnection - uninitialized conn
DTX-CM-W-3510	Adapter GET returns bad conn
DTX-CM-W-3511	Failed to drop bad conn (mgetputres)

Logging messages for Error

This documentation presents the logging messages for the Error logging category for each logging entry (Launcher, Resource Manager, Connection Manager) listed by each message category (41xx, 43xx, 45xx).

- [Launcher error 41xx](#)
- [Resource Manager error 43xx](#)
- [Connection Manager error 45xx](#)

Launcher error 41xx

Message ID	
Message	
DTXLN-E-4101	Launcher error log enabled
DTXLN-E-4102	Launcher error log disabled
DTXLN-E-4103	Map error <map /error>
DTXLN-E-4104	Connection error <value>
DTXLN-E-4105	Socket error <value>
DTXLN-E-4106	Alloc thread param failure
DTXLN-E-4107	Rmadpendingentry failure
DTXLN-E-4108	Linkpendingthread error
DTXLN-E-4109	Resolvewhildcards memory failure
DTXLN-E-4110	SimulateMapRun <values>
DTXLN-E-4111	Linkpendingthread memory failure
DTXLN-E-4112	Copythreadparam memory failure
DTXLN-E-4113	Initsocketstuff memory failure
DTXLN-E-4114	Socket send failure
DTXLN-E-4115	Socket recv failure

DTXLN-E-4116
 PDH error (WIN32 only)
DTXLN-E-4117
 Can't get exit function

Resource Manager error 43xx

Message ID
Message
DTX-RM-E-4301 Resource Manager error log enabled
DTX-RM-E-4302 Resource Manager error log disabled
DTX-RM-E-4303 Lock failed
DTX-RM-E-4304 Unlock failed
DTX-RM-E-4305 Enumprocess failed (WIN32 only)
DTX-RM-E-4306 Reached max pids
DTX-RM-E-4307 Rmaccessfile failure
DTX-RM-E-4308 Rmreallocatable failed
DTX-RM-E-4309 GetAdptMaskedCmd failure
DTX-RM-E-4310 Linkrmt returns NULL
DTX-RM-E-4311 Linkct returns NULL
DTX-RM-E-4312 Linknbb returns NULL
DTX-RM-E-4313 Rmaddrunentry failure
DTX-RM-E-4314 Suspend thread failure
DTX-RM-E-4315 FTG link error
DTX-RM-E-4316 NULL CT in rmgetdetails
DTX-RM-E-4317 NULL NBB in rmgetdetails

Connection Manager error 45xx

Message ID
Message
DTX-CM-E-4501 Connection Manager error log enabled
DTX-CM-E-4502 Connection Manager error log disabled
DTX-CM-E-4503 Invalid thread - can't be suspended <value>
DTX-CM-E-4504 Suspend thread failed <value>
DTX-CM-E-4505 Invalid thread - can't be resumed <value>
DTX-CM-E-4506 Thread - not suspended <value>
DTX-CM-E-4507 SAM error <values>
DTX-CM-E-4508 InvokeAdapterMethod System exception
DTX-CM-E-4509 Invalid rmthread signature
DTX-CM-E-4510 Rmthread --> post call restore failed
DTX-CM-E-4511 Attempt to use destroyed adpt object
DTX-CM-E-4512 Can't initialize conn. Object
DTX-CM-E-4513 Illegal GTX TLS conn. cnt # <value>

```
DTX-CM-E-4514
    Attempt to destroy adpt obj in use
DTX-CM-E-4515
    Rmidleboot init failure
DTX-CM-E-4516
    Illegal cleanup call - context missing <value>
DTX-CM-E-4517
    Illegal GTX cleanup call - context missing <value>
DTX-CM-E-4518
    Failed to allocate memory - rmgetputres
DTX-CM-E-4519
    Invalid source event
DTX-CM-E-4520
    Malloc failed GTX TLS init
```

Logging messages for Fatal

This documentation presents the logging messages for the Fatal logging category for each logging entry (Launcher, Resource Manager, Connection Manager) listed by each message category (51xx, 53xx, 55xx).

- [Launcher fatal 51xx](#)
 - [Resource Manager fatal 53xx](#)
 - [Connection Manager fatal 55xx](#)
-

Launcher fatal 51xx

Message ID	
Message	
DTXLN-F-5101	Launcher error log enabled
DTXLN-F-5102	Launcher error log disabled
DTXLN-F-5103	MSL Mismatch <value>
DTXLN-F-5104	Invalid MMC <mmc_name>
DTXLN-F-5105	Cardinfo failure <mapname>
DTXLN-F-5106	Mpicreateallobjects failed <mapname>
DTXLN-F-5107	Init adapters failure <error_number>
DTXLN-F-5108	Resource manager init failure <value>
DTXLN-F-5109	Connection manager init failure <value>
DTXLN-F-5110	Socket failure <error>
DTXLN-F-5111	Thread create failure <error>
DTXLN-F-5112	Memory allocation failure <error>
DTXLN-F-5113	Readlauncherfile failed <error>
DTXLN-F-5114	Loadtriggers failed <error>
DTXLN-F-5115	No MSL files

Resource Manager fatal 53xx

Message ID	
Message	
DTX-RM-F-5301	Resource Manager error log enabled
DTX-RM-F-5302	Resource Manager error log disabled
DTX-RM-F-5303	RMworker thread create failed <error>
DTX-RM-F-5304	Semaphore create failed <error>
DTX-RM-F-5305	

Connection Manager fatal 55xx

Message ID

Message

DTX-CM-F-5501

 Connection Manager error log enabled

DTX-CM-F-5502

 Connection Manager error log disabled

DTX-CM-F-5503

 Rmloadadapter failure <error>

DTX-CM-F-5504

 Rminittriggers failure <error>

Logging option

If you select the logging option when you have configured the Launcher to run multiple systems as a compound system in a single Launcher process or as separate Launcher processes and then run the system or systems, there are two log files that are created in the logs subdirectory of the *install_dir*. Both log files have the same name as the system (.msl) file, appended with the current date and time stamp (yyyy-mm-dd-hh-mm-ss) and the host name of the computer where the Launcher is running.

The first log file has a .log filename extension.

The second log file has a .txt filename extension.

See [Trace and Log Files Naming Conventions](#) for the specific naming conventions.

- [Launcher-created log file permissions](#)

The Launcher creates its log files with the default file permissions **rw-rw-rw-**. You can change the log file permissions by changing the LogFilePermissions setting in the **config.yaml** file.

- [Example](#)

- [Log file types](#)

- [Environmental debug \(UNIX\)](#)

Launcher-created log file permissions

The Launcher creates its log files with the default file permissions **rw-rw-rw-**. You can change the log file permissions by changing the LogFilePermissions setting in the **config.yaml** file.

The LogFilePermissions setting is specified in the **[Launcher]** section of the *install_dir\config.yaml* file:

```
; this defines the permissions used to create the
; log files for the Launcher:
; default 666 (rw-rw-rw-)
; rw-rw-r-- is 664
; rw-r--r-- is 644
LogFilePermissions=666
```

To change the log file permissions, change the value of the LogFilePermissions setting:

Value	File Permissions
666	rw-rw-rw- This is the default file permission.
664	rw-rw-r--
644	rw-r--r--

Example

This is an example of file names generated from running a system with logging enabled.

Run a system on a computer with the hostname *mycomputer*, where the system file path is *install_dir\systems\small.msl* and logging is enabled. The date is May 29, 2017 and the time is 10:31:11 AM. The following two files are generated in the logs subdirectory of the *install_dir*:

- small-2017-05-17-10-31-11_mycomputer.log
- small-2017-05-17-10-31-11_mycomputer.txt

With logging enabled, these files are generated with the current date and time stamp every time the system is started.

Log file types

Using the logging option produces files that contain information falling under the following log file types:

- [Launcher Activity Log Files](#)
 - [Debug Log Files](#)
 - [Diagnostic Log Files](#)

 - [Launcher activity log files](#)
 - [Debug log files](#)
 - [Diagnostic log files](#)
-

Launcher activity log files

The text (.txt) Launcher log file produced from the Launcher contains information about Launcher activities including summary and detail information that is useful for analysis, debugging and troubleshooting. Based on default parameters, the log file contains information about:

- Launcher configuration
- Launcher startup
- Map warnings
- Map errors

The default location for the Launcher log file is in the **logs** subdirectory of the *install_dir*. As described in an example (see [Example](#)), the file name is based on the system name and current date.

For more information, see the following documentation:

- [Trace and Log Files Naming Conventions](#) for the specific naming conventions
 - [LauncherLog](#)
-

Debug log files

The text (.txt) debug log file produced from the Launcher contains information about Launcher activities that is helpful for analysis, debugging and troubleshooting. Based on the default parameters, this file contains information about:

- Watch triggers
- Time and input events
- Error messages

The text debug log file is also generated in the logs subdirectory of the *install_dir* and generates upon startup. As described in [Logging Option](#), the file name is based on the system name and current date.

For more information, see the following documentation:

- [Trace and Log Files Naming Conventions](#) for the specific naming conventions
 - [LauncherLog](#)
-

Diagnostic log files

Additional files are created when certain events occur with the Launcher:

- If an unhandled system exception occurs in a map thread that is running, troubleshooting information for technical support purposes is written to a file named **GPFThreadId-TimeStamp.txt** on Windows and **ABORT-pid.txt** on UNIX. You can find this diagnostic log file in the installation directory of WebSphere® Transformation Extender.
- The **dtxAbort.log** file is a UNIX environment-specific log file that indicates when the Launcher, Command Server, Platform API, or Java™ API is attempting to acquire memory from the operating system but does not succeed.
The **dtxAbort.log** file gets created in **\$DTX_TMP_DIR** when the respective application is started. **\$DTX_TMP_DIR** is normally empty, which indicates that there has not been a failed attempt to get additional memory.

If a failure occurs while attempting to obtain additional memory, the message appended to the **dtxAbort.log** file will be similar to one of the following messages:

- Fatal Error: malloc() - Unable to Retrieve Additional New Memory.
 - Fatal Error: realloc() - Unable to Retrieve Additional New Memory.
- The message is prefaced with text indicating the date and time that the failure occurred.
-

Environmental debug (UNIX)

To turn on the environmental debug option for UNIX Launcher, you must set the DTX_DEBUG variable to **TRUE** before you start the Launcher (by using the **launcher.sh -start** command). The environmental debug option does not work unless this action is performed.

To set the option:

```
sh: DTX_DEBUG=TRUE
      export DTX_DEBUG
```

System and watch status in JSON format

The Launcher heartbeat file summarizes the status of Launcher systems and watches in a JSON-formatted log file that an external program can read to automatically monitor Launcher status.

The Launcher logs the following information in the heartbeat file:

- Current time in ISO8601 format
- System statistics:
 - Output equivalent to the **launcher -summary** command
 - Launcher warning, error, and fatal error counts. Informational messages about log file enablement are not counted.
- System deltas:
 - Number of successful maps, map errors, and total number of maps since the last heartbeat interval
 - Number of system warnings, errors, and fatal errors since the last heartbeat interval
- Watch statistics:
 - Watch name
 - Summary:
 - Number of instances triggered
 - Total time of all instances in milliseconds
 - Minimum time for one instance in milliseconds
 - Maximum time for one instance in milliseconds
 - Average time for one instance in milliseconds
- Delta: The change to each of the Summary values since the last heartbeat interval

Configuring the Launcher heartbeat file

Options in the **/runtime/Launcher** section of the config.yaml file enable and configure the Launcher heartbeat file:

HeartbeatTimeInterval

The interval in seconds between heartbeat files. Specifying 0 disables heartbeat monitoring.

HeartbeatFileKeepNum

The number of heartbeat files to keep before the oldest file is deleted.

The most recent heartbeat file has the Launcher log file name and the .json file extension. For example:

CompoundSystem04-11-17-05-45-42-AM_myhost.json

Older heartbeat files have a numeric index appended to the file extension. For example:

CompoundSystem04-11-17-05-45-42-AM_myhost.json.1

Heartbeat file example

The log file format is similar to the following:

```
"Time": "2017-04-11T16:05:23-0400",
"System Statistics":
  "System Status": "Running",
  "Active Component Maps": "0",
  "Active Listeners Up": "20",
  "Active Listeners Down": "0",
  "Active Connections": "1",
  "Start Time": "2017-04-11T16:05:00-0400",
  "Success Time (dhms)": "0:15",
  "Failure Time (dhms)": "0:00",
  "Up Time (dhms)": "0:23",
  "Pending Initialization": "0",
  "Pending Initialization Maximum": "12",
  "Pending Resource": "0",
  "Pending Connection": "0",
  "Pending Total Maps": "0",
  "History Successes": "369",
  "History Failures": "0",
  "History Total Maps": "369",
  "History Connection Failures": "0",
  "History Deadlocks Detected": "0",
  "History Function Failures": "0",
  "System Warnings": "1",
  "System Errors": "11",
  "System Fatals": "0"

"System Deltas":
  "History Successes": "206",
  "History Failures": "0",
  "History Total Maps": "206",
  "System Warnings": "0",
  "System Errors": "0",
  "System Fatals": "0"

"Watch Statistics":
  "Name": "PreProcessEnvelope",
  "Summary":
    "Triggered": "3",
```

```

    "Tot Time (ms)": "384",
    "Min Time (ms)": "80",
    "Max Time (ms)": "208",
    "Ave Time (ms)": "128" "Deltas":
    "Triggered": "2",
    "Tot Time (ms)": "176",
    "Min Time (ms)": "80",
    "Max Time (ms)": "96",
    "Ave Time (ms)": "88"
  "Name": "ProcessInvoice",
  "Summary":
    "Triggered": "3",
    "Tot Time (ms)": "135",
    "Min Time (ms)": "31",
    "Max Time (ms)": "67",
    "Ave Time (ms)": "45"

  "Deltas":
    "Triggered": "2",
    "Tot Time (ms)": "68",
    "Min Time (ms)": "31"
    "Max Time (ms)": "37",
    "Ave Time (ms)": "34"

```

Launcher Monitor

The Launcher Monitor provides graphical, detailed information about maps that are running in the Launcher. Using the Monitor, you can create, update, delete, and start Launcher connections for monitoring. Although you can set up more than one Launcher, you can only connect to and view one at a time.

The Launcher Monitor provides a dynamic view of watches as they run. A watch is a single map that is running. Watches are sequentially numbered starting with 1. When you start the Launcher Monitor and maps begin to run, a window for each watch appears within the Launcher Monitor window.

- [How the Monitor works](#)
- [Using the Launcher Monitor](#)
- [Configuring the Monitor display](#)
- [Snapshots](#)

How the Monitor works

The Launcher Monitor works with the Launcher by means of Transmission Control Protocol/Internet Protocol (TCP/IP). Because this set of protocols allows for communications over interconnected, dissimilar networks, the Monitor can monitor the Launcher on a Windows or UNIX system.

Interprocess communication (IPC) between the server and client requires:

- An address (or computer name)
- A port number

Using the Launcher Monitor

To use the Monitor to view Launcher processes (systems) that are running in the Launcher, a series of steps must first take place:

- One or more **.msl** files are generated and placed in the deployment directory that you specified in the Launcher Administration interface.
- The Launcher service/daemon is running.
- The Launcher is configured and a process is running in the Management Console.
- From the Launcher Monitor, you must configure the server and port number where the Launcher is running.

To open the Launcher Monitor

Select Launcher Monitor from the Start > Programs menu under **WebSphere® Transformation Extender n.n** (where **n.n** represents the version number). The Launcher Monitor window opens.

To start the Launcher Monitor

1. From the Launcher Monitor window, select Options > Select Launcher.
The Select Launcher window appears.
2. Click Add.
The Add Launcher window appears.
3. Enter a name for the Launcher and click OK.
4. Under **User**, enter the user name and password that corresponds to the server you need to connect to so that you can monitor a Launcher.
5. Under **Connection**, enter the name or IP address of the server where the Launcher is running in the **Server** field.
Note: The **Server** connection is either obtained from network configuration information or can be a TCP/IP address.
6. In the **Port** field, type the port number that you want to use or accept the default port number, which is **5016**.
The port number is always the Launcher port number (configured in the Management Console) plus **1**. The default port for the Launcher is **5015**; therefore, the default port here is **5016**. See [Default Port Settings](#) for more information about default port settings.

On a UNIX server, the default port setting can be changed in the services file located on the machine where the Launcher is installed: /etc/services.

On the z/OS® operating systems, the default port setting can optionally be changed in the TCPIP.ETC.SERVICES or in /etc/services, if installed.

7. Under **Timeout**, enter a value (in seconds) in the **Response** field. This value can range from 30 to 3600 seconds, which is the time that can elapse without an initial response from the Launcher before a timeout occurs. The default setting is 30 seconds.

8. Do one of the following activities:

- To save this Launcher connection and close the Select Launcher window, click OK.
 - To connect to the Launcher and begin monitoring, click Connect.
- The Select System window opens.

Notice that both .msl files are listed under System File Name. This is because if you have more than one .msl file running, all of the data consolidates into one compound system. This is the same functionality as the Management Console.

9. Select CompoundSystem from the list (see [Multiple Processes](#) for additional information when using the multiple-processes feature) and click OK.

10. From the Select Launcher window, click OK.

Each map that is running in the process is displayed in an individual window as a sequentially numbered watch.

Configuring the Monitor display

There are many options that you can use to configure the Monitor display according to your preferences. For example, you can use the following options to configure the Monitor display:

- Arrange the watch windows.
- Change update intervals.
- Change background and grid line colors.

Each option is described here.

Note: All procedures to configure the Monitor display begin from the Launcher Monitor window.

To arrange watch windows

You can arrange watch windows to display in the Launcher Monitor window.

Select View>Tile.

- [Watches](#)
- [Adjusting colors](#)

Watches

There are two options for configuring watches:

- Changing the interval
- Choosing specific watches

To change the interval for watch updates

You can change the interval at which the watch windows in the Monitor are updated. Options include $\frac{1}{4}$ second, $\frac{1}{2}$ second, and 1 second.

1. Select Options>Update Interval.
2. Choose the number of seconds: $\frac{1}{4}$, $\frac{1}{2}$, or 1.

To select specific watches

You can select specific watches to view as well as the watch instance limit for the maximum number of watch instances.

Select Options>Snapshot Settings.
The Watch Settings window opens.

Adjusting colors

You can choose from nine different colors for the Monitor's background, grid lines, and maps. Additionally, you can select a different color for each type of map status. For example, maps in progress, successful maps, maps with warnings, and maps with errors. All options are available from the colors menu.

Vertical Grid

There are two options for configuring the vertical grid:

- Changing the spacing
- Changing the time interval

To change vertical grid spacing

You can change the vertical grid spacing by choosing the number of pixels per second. Choices include 10, 20, and 40 pixels.

1. Select Options > One Second Equals.
2. Choose the number of pixels: **10**, **20**, or **40**.

To change the vertical grid time interval

You can change the time interval between vertical grid lines. Options include $\frac{1}{2}$ second, 1 second, and 2 seconds.

1. Select Options > Vertical Grid.
2. Choose the number of seconds: **10**, **20**, or **40**.

Snapshots

From the Launcher Monitor, you can take snapshots of a watch running as well as configure how the snapshots are taken. Snapshots capture Launcher activity. The Snapshot Viewer enables you to select the type of snapshot to capture each time an error occurs, for any error or warning, or for all map instances. See ["Snapshot Viewer"](#) for more information about the Snapshot Viewer.

- [Snapshots on demand](#)
- [Snapshot settings](#)

Snapshots on demand

You can take a snapshot at any time during processing.

To capture a snapshot

1. Click the **Snapshot!** menu command.
2. Open the Snapshot Viewer (["Snapshot Viewer"](#)) to view detailed information about the snapshot.

Snapshot settings

The snapshot settings available in the Monitor enable you to configure how snapshots are taken. For example, you can select specific watches or set a maximum limit for instances. The documentation provides a description of all the options.

To access Snapshot settings from the Launcher Monitor

Select Options > Snapshot Settings.
The Snapshot Settings window opens.

Each setting is described in the following table.

Control	Description
---------	-------------

Instance Limit	Maximum number of total watch instances to include in a single snapshot. The maximum number of watch instances to be included in one snapshot is 1000. Click the arrow buttons to change the limit.
-----------------------	---

Snapshot	Select an option to specify the type of snapshot:
-----------------	---

On Demand Only	- Snapshots are only taken when you choose the Snapshot! menu command.
-----------------------	--

On Error	- A snapshot is taken automatically whenever a map error occurs.
-----------------	--

On Warning/Error	- A snapshot is taken automatically whenever a map error or warning occurs.
-------------------------	---

Continuous	- A continuous snapshot is taken, including all watch instances specified by Instance Limit .
-------------------	--

Include these watches

Select a check box next to each watch to include in a snapshot.

Snapshot Viewer

The Snapshot Viewer enables the analysis of snapshots taken of Launcher activity from the Launcher Monitor.

- [How are Snapshots created?](#)

How are Snapshots created?

A snapshot of Launcher activity is saved from the Launcher Monitor by clicking **Snapshot!** from the menu bar.

Use the Snapshot Viewer to view and analyze the snapshots taken. Each snapshot contains information that is used to display a chart of map instances over time.

When a snapshot is created, it is written to a file with this format:

MMDD<four-digit-number>.mss

where MMDD represents the two-digit month and two-digit day on which the snapshot was created. The four-digit number begins with one and is increments by one for each snapshot for the current day.

The Snapshot file (.mss) is created in the WebSphere® Transformation Extender installation directory by default.

To view a snapshot

This procedure assumes that you have already captured a snapshot from the Launcher Monitor.

1. From the Start->Programs menu, open the Snapshot Viewer under **WebSphere Transformation Extender n.n** (where **n.n** represents the version number).
2. Select **File->Open**.
3. Navigate to the .mss file, which is located in the WebSphere Transformation Extender installation directory, and open it.

The snapshot from the Launcher Monitor appears.

Map instances are color-coded, so it is easy to understand a map's status at a glance. The default colors are:

- **Green**-Maps that completed successfully.
- **Yellow**-Maps that encountered a warning condition during execution.
- **Red**-Maps that did not complete because of an error condition.

From the Colors menu, you can change colors for map instances, background, grid, and text associated with a map instance.

To see detailed information about the map instance

Right-click over a map instance and hold to see detailed information about the execution for that instance of the map.

Command line options

You can use the command line to control the Launcher and to configure all of the options that are available through the graphical Management Console and Launcher Administration tools. The command line supports automation (for example, by using a batch file or shell script) and unattended installation (for example, in a Hypervisor environment).

In addition to the Microsoft Windows **net pause** and **net continue** commands, the Launcher supports the following commands:

Table 1. Launcher commands for
Windows and UNIX systems

Windows	UNIX
launcher.bat	launcher.sh
launcheradmin.bat	launcheradmin.sh
mgmtconsole.bat	mgmtconsole.sh

On Microsoft Windows systems, the Launcher runs as a service. On UNIX and z/OS® systems, the Launcher runs as a daemon. The command line topics use the term *Launcher service* generically to refer to either environment.

- [**Syntax conventions**](#)
The command line syntax identifies keywords and variables, required and optional choices, and default values.
- [**Running the setup script**](#)
Run the **setup** script before you invoke any command. The **setup** script sets up the paths for the binary modules and libraries to load and run properly.
- [**Net pause and net continue commands**](#)
- [**Launcher command line syntax**](#)
You can use the command line to control the Launcher and get system status. You can also set trace and logging options as well as stop, start, pause, resume, and detect the status of a system.
- [**Launcher Administration command line syntax**](#)
The Launcher Administration commands configure the Launcher service and the launchers that are created by the Launcher service. You can use Launcher Administration commands to add and remove HTTP Listeners and to configure deployment directories, users and user access rights, listening ports, and Launcher service properties.
- [**Management Console command line syntax**](#)
Use the Management Console options to create the Launcher definitions and firewall and override configurations. Each launcher definition establishes a connection to one Launcher service.
- [**HTTP Listener command line syntax**](#)
You can use the **httplsnr** command to configure, start, and stop the Launcher HTTP Listener.

Syntax conventions

The command line syntax identifies keywords and variables, required and optional choices, and default values.

Syntax item

Meaning

Normal font

Keyword

italics

Variable
underline
 Default value
`[x | y]`
 Optional choice. You can choose either option or omit the parameter.
`{ x | y }`
 Required choice. You must specify one of the options.

Additional requirements:

- If a parameter contains blanks, enclose the parameter in quotation marks (" "). For example, specify:

```
"C:\IBM\WebSphere Transformation Extender
9.0.0\latest install"
```

- On UNIX and z/OS® systems, use quotation marks (" ") to enclose multiple parameters that are delineated by semicolons (;). For example, specify:

```
launcheradmin.sh -adduser "user:admin;login:admin;pwd:admin"
```

Examples

In the following example, the -action parameter is optional. If you specify the -action parameter, you must specify either the create keyword or the append keyword:

```
[-action {create | append}]
```

Running the setup script

Run the **setup** script before you invoke any command. The **setup** script sets up the paths for the binary modules and libraries to load and run properly.

From the command prompt, enter the following command:

```
. setup
```

Related reference

- [Launcher command line syntax](#)
 - [Launcher Administration command line syntax](#)
 - [Management Console command line syntax](#)
-

Net pause and net continue commands

On Windows, the Launcher is controlled by the Launcher service. *Pausing* entails pausing all Launchers. *Resuming* entails resuming or continuing any paused Launchers. The **net pause** and **net continue** commands interact with the Launcher service to pause and resume the Launcher.

The Windows **net pause** and **net continue** commands are used in the following ways:

```
net pause {ServiceName | "Service Name"}
net continue {ServiceName | "Service Name"}
```

where *ServiceName* is the case-sensitive name of the service that is being controlled, such as **WebSphere® Transformation Extender Launcher**.

To use the Net command options, the service must be started, otherwise an error message is displayed at the command prompt: **The service has not been started**. The command line will be blocked until the service pauses or resumes.

Pause and **Resume** actions can also be initiated from the Windows Computer Management console. The service must be started; otherwise the **Pause** option will not be enabled. If the service is not in a paused state, the **Continue** option is not enabled.

The following table lists the Net command-line options. Note the following rules that need to be observed when specifying service names on the command line:

- The service name must be enclosed in double quotation marks if it contains spaces.
- Service names are case-sensitive.

Command	Description
<code>net pause "WebSphere Transformation Extender Launcher"</code>	Pauses the Launcher service that is running
<code>net continue "WebSphere Transformation Extender Launcher"</code>	Resumes the paused Launcher service

The return values for the Net commands are zero for success and non-zero for failure.

Launcher command line syntax

You can use the command line to control the Launcher and get system status. You can also set trace and logging options as well as stop, start, pause, resume, and detect the status of a system.

Use the **launcher.bat** command in a Microsoft Windows environment and the **launcher.sh** command in a UNIX environment.

In a Microsoft Windows environment, the Launcher runs as a service. You use Start > Control Panel > Administrative Tools > Services to stop, start, pause, and resume the Launcher.

Supported options in a Microsoft Windows environment:

```
launcher.bat
[-status launchername {systemname | -lsnr listener_name}]
[{-summary | -history | -statusinfo | -configuration | -all} [-xml]
 launchername systemname]
[{-startsystem | -stopsystem} launchername {systemname | -all}]
[{-pausesystem | -resumesystem} launchername {systemname | -all}]
[-settrace launchername systemname systemfilename componentname
 -trace {default | on | off} [-action {create | append}] [-tracefile tracefile_name] ]
[-setlogging launchername systemname -logtype {launcher | resmgr | commgr}
 -logflags {[to | tx],[do | dx],[wo | wx],[fo | fx],[io | ix],[eo | ex]}}
 [-list launchername [{-lsnr listener_name} | {systemname [systemfilename [componentname]]}]] [-trace] [-logging]]
```

Supported options in a UNIX environment:

```
launcher.sh
[-status launchername {systemname | -lsnr listener_name}]
[{-summary | -history | -statusinfo | -configuration | -all} [-xml] launchername systemname]
[-start [adminprop.xml]]
[-stop [listenerport [hostname]]]
[-pause [listenerport [timeout [hostname]]]]
[-resume [listenerport [timeout [hostname]]]]
[{-startsystem | -stopsystem} launchername {systemname | -all}]
[{-pausesystem | -resumesystem} launchername {systemname | -all}]
[-settrace launchername systemname systemfilename componentname
 -trace {default | on | off} [-action {create | append}] [-tracefile tracefile_name] ]
[-setlogging launchername systemname -logtype {launcher | resmgr | commgr}
 -logflags {[to | tx],[do | dx],[wo | wx],[fo | fx],[io | ix],[eo | ex]}}
 [-list launchername [{-lsnr listener_name} | {systemname [systemfilename [componentname]]}]] [-trace] [-logging]]
```

- [Status command](#)

The **launcher -status** command returns the state of the specified Launcher system or Launcher HTTP Listener. The status returns as a message to the console or as a return code that you can retrieve by using the **ECHO** command.

- [Snapshot commands](#)

- [Starting, stopping, pausing, and resuming the Launcher service](#)

On UNIX systems, you can use command line options to start, stop, pause, and resume the Launcher daemon.

- [Starting, stopping, pausing, and resuming systems](#)

You can use the command line to control a specific system or all systems.

- [Trace command](#)

Use the trace command to set the trace flags for a component that is running under the specified system or compound system on the specified Launcher.

- [Logging command](#)

Use the logging command to set the logging flags for the Launcher, the Resource Manager, or the Connection Manager.

- [List system information command](#)

Use this command to list information about Launcher systems and Launcher HTTP Listeners. System information includes system files, components within each system, trace settings for each component in a system, and log settings for the system. Listener information includes ports, certificates, logging, and audit enablement.

Related reference

- [Syntax conventions](#)
- [Running the setup script](#)

Status command

The **launcher -status** command returns the state of the specified Launcher system or Launcher HTTP Listener. The status returns as a message to the console or as a return code that you can retrieve by using the **ECHO** command.

Syntax

```
{launcher.bat | launcher.sh}
[-status launchername {system_name | listener_name}]
```

system_name

Returns the status of the specified Launcher system. The *system_name* keyword is optional, but you must specify either a *system_name* or a *listener_name* for the specified Launcher.

The system status is one of the following:

Table 1. Launcher system return codes and status

Return code	Status
-------------	--------

Return code	Status
0	Stopped
1	Stopping
2	Pausing
3	Paused
4	Started
5	Starting
6	Not Responding
7	Unknown Error
101	The Launcher (with the specific settings) is not started.
102	The name of the Launcher supplied on the command line is not valid.
103	The user name, password, or both user name and password are not valid.
104	The connection between this utility and the Launcher is lost when trying to retrieve the status. For example, if another user stops the Launcher.
105	The command line supplied is invalid.
106	Launcher is not in a running state.
107	The name of the system (<i>systemname</i>) supplied on the command line is invalid, or it is blank and there is no compoundsystem present under the Launcher service.
108	The Launcher raised an exception.
	Unable to connect to the Launcher.

listener_name

Returns the status of the specified Launcher HTTP Listener, as well as the start time and running time (*up time*) of a running listener. The *listener_name* keyword is optional, but you must specify either a *system_name* or a *listener_name* for the specified Launcher.

The HTTP Listener status is one of the following:

Table 2. HTTP Listener return codes and status

Return code	Status
0	Stopped
1	Starting
2	Running
3	Not responding

Example

The following example returns the HTTP Listener status as a message to the console and retrieves the status return code.

```
C:\IBM\WebSphere Transformation Extender
9.0.0>launcher -status MyServer -lsnr LSNRA
Status of the listener "LSNRA" on the server "localhost" and port "5015" is "Listener Running"
Listener "LSNRA" Start Time is 16:24:11 and Up Time is 03:53
C:\IBM\WebSphere Transformation Extender
9.0.0>echo %ERRORLEVEL%
2
```

Snapshot commands

Snapshot options are available using the Windows or UNIX command line. You can use these commands for automation with batch files.

The Snapshot options from the command line facilitate the capturing of Launcher information by enabling you to:

- Request and view the information in XML or text format.
- Redirect and store the information in a text or XML file.
- [Using Snapshot commands](#)
- [File storage commands](#)
- [Display commands](#)
- [Return codes for system-based command options](#)

Using Snapshot commands

The same Snapshot command line options are available for both Microsoft Windows and UNIX platforms. For Windows, the associated file name is **launcher.bat**; for UNIX, it is **launcher.sh**. The following table lists the Snapshot command options.

Command

Description

-summary

This option retrieves summary information about:

- Memory and CPU Usage
- System Status
- Active Component Maps
- Active Listeners Up and Down
- Active Connections

- Start, Success, Failure, and Up Time
- Pending Initialization, Resource, Connection, and Total Maps
- History Successes, Failures, Total Maps, Connection Failures, Deadlocks Detected, and Function Failures

-history

This option retrieves history information about:

- Map Failures
- Adapter Connections
- Function Failures

-statusinfo

This option retrieves status information about:

- Active Components
- Pending Components
- Adapter Connections

-configuration

This option retrieves configuration information about:

- System
- Adapter Connections

-all

This option retrieves all Launcher information; or is the equivalent to invoking the **-summary**, **-statusinfo**, **-history**, and **-configuration** commands together.

It also can be combined with any of the **-startsystem**, **-stopsystem**, **-pausesystem**, and **-resumesystem** options so that all the systems can be acted upon at one time. This is equivalent to using the **Start All**, **Stop All**, **Pause All** and **Resume All** menu items in the Management Console.

-xml

This option enables you to retrieve and display Launcher information in XML format. The XML format follows the DTD definition of the snapshot.dtd file that is shipped with the Launcher. If you specify this option, the output is printed in XML format. If this option is not specified, the output is printed in a readable text format.

When using a Snapshot command, in addition to specifying either the **launcher.sh** (UNIX) or **launcher.bat** (Windows) file, you must specify the Launcher name that was configured in the Management Console.

Related reference

- [Return codes for system-based command options](#)
-

File storage commands

On Windows platforms, the following DOS redirection command can be used to store the summary information of the **CompoundSystem** (see ["Multiple Processes"](#) for additional information when using the multiple-processes feature) running under a Launcher named "Server1" in XML format:

```
launcher.bat -summary -xml "Server1" > MySnapshot.xml
```

On UNIX platforms, the following command line can be used to store the summary information of the **CompoundSystem** running under a Launcher named "Server1" in XML format:

```
launcher.sh -summary -xml "Server1" > "/usr/home1/
snapshot.xml"
```

Display commands

On Windows platforms, the following command can be used to display all information about the **CompoundSystem** (see ["Multiple Processes"](#) for additional information when using the multiple-processes feature) running under a Launcher named "Server1" in a readable text format:

```
launcher.bat -all "Server1"
```

On UNIX platforms, the following command line can be used to display all information about the **CompoundSystem** running under a Launcher named "Server1" in a readable text format:

```
launcher.sh -all "Server1"
```

Return codes for system-based command options

For all Snapshot command line options, including **-summary**, **-statusinfo**, **-history**, and **-configuration**, and system-controlling commands, including **-startsystem**, **-stopsystem**, **-pausesystem**, and **-resumesystem**, an error message is displayed if the operation is not successful.

If the operation is successful, a code will be returned by the batch file as follows:

- 0 for Snapshot command line options
- 1 for system-controlling commands

Code	Description
101	The specific Launcher is not started.
102	The name Launcher supplied on the command line is not valid.
103	The user name and password are not valid.
104	The connection between this utility and the Launcher is lost during the attempt to retrieve summary information. For example, this might happen if trying to find the summary information and another user stops the Launcher.
105	Command is invalid.
106	Launcher is not in a running state.
107	The name of the system (<code>systemname</code>) supplied on the command line is invalid, or it is blank and there is no compoundsystem present under the Launcher service.
108	The Launcher raised an exception.
109	Unable to connect to the Launcher.
110	The control code and the system state are not compatible. Note: This code is valid only for system-controlling commands.
111	The user does not have the proper user rights to control the system. Note: This code is valid only for system-controlling commands.

Starting, stopping, pausing, and resuming the Launcher service

On UNIX systems, you can use command line options to start, stop, pause, and resume the Launcher daemon.

```
launcher.sh
[-start [adminprofile]]
[-stop [listenerport [hostname]]]
[-pause | -resume [listenerport [timeout [hostname]]]]

-start [adminprofile.xml]
Starts the Launcher daemon with the properties that are configured in the adminprofile file. If the adminprofile file is not specified or fails to import, starts the Launcher daemon with the properties that are configured in the launcheradmin.bin properties file. If the launcheradmin.bin file does not exist, starts the Launcher daemon with the default properties.

In a Microsoft Windows environment, you specify the name of the properties file on the ADMINPROPFILE setting of the [LAUNCHER] section in the JavaServices.ini file. The properties file that is configured on the ADMINPROPFILE setting overrides the LauncherAdmin.bin file. If the LauncherAdmin.bin file does not exist, the Launcher daemon uses default settings on startup.

Start only one Launcher daemon per Launcher installation.

-stop [listenerport [hostname]]
Stops the Launcher daemon that is listening on the specified listener port and host system. If the listener port and host system are not specified, stops the Launcher daemon that is listening on the default listener port (port 5015).

The -stop command has a default timeout interval of 120 seconds. During the timeout interval, the Launcher daemon is blocked by the command line until the Launcher daemon stops.

-pause [listenerport [timeout [hostname]]]
Pauses the Launcher daemon that is running on the specified listener port and host name, if specified. Otherwise, pauses the Launcher service that is listening on the default listener port (port 5015).

-resume [listenerport [timeout [hostname]]]
Resumes the Launcher daemon that is paused on the specified listener port and host name, if specified. Otherwise, resumes the Launcher service that is listening on the default listener port (port 5015).

timeout
A value from 30 to 3600 seconds that specifies the time interval that can elapse without an initial response from the Launcher service before a failure is reported. During the time interval, the Launcher daemon is blocked by the command line unless the Launcher daemon pauses or resumes. When the timeout interval expires, the Launcher daemon returns to the command prompt.

The default timeout value is zero. A timeout value of zero indicates that the Launcher daemon is blocked by the command line for an infinite amount of time until the Launcher daemon pauses or resumes.
```

Based on the following command example, if the Launcher daemon pauses before 60 seconds, TRUE or SUCCESS is returned. Otherwise launcher.sh exits, returning FALSE.

```
. setup
launcher.sh -pause 3000 60
```

Related tasks

- [Starting and stopping the Launcher](#)
- [Controlling systems](#)

Starting, stopping, pausing, and resuming systems

You can use the command line to control a specific system or all systems.

```
{launcher.bat | launcher.sh}
[{-startsystem | -stopsystem} launchername {systemname | -all}]
[{-pausesystem | -resumesystem} launchername {systemname | -all}]
```

{-startsystem | -stopsystem} launchername {systemname | -all}

Starts or stops a specific system or all of the systems that are on the specified Launcher.

{-pausesystem | -resumesystem} launchername {systemname | -all}

Pauses or resumes a specific system or all of the systems that are on the specified Launcher.

Trace command

Use the trace command to set the trace flags for a component that is running under the specified system or compound system on the specified Launcher.

```
{launcher.bat | launcher.sh}
[-settrace launchername systemname systemfilename componentname
 -trace {default | on | off} [-action {create | append}] [-tracefile tracefile_name] ]
```

-settrace launchername

The name of the Launcher that is defined in the mgmtconsole.bin file. The *launchername* variable identifies the host name, port, user name, and password of a particular Launcher definition.

systemname

The name of the system or group of systems running under one Launcher process. The *systemname* is defined with the Launcher Administration tool.

systemfilename

The name of an individual system file (.msl) that is running under the specified system and Launcher.

componentname

The name of an individual map component in the system file that is running on the specified system and Launcher.

-trace

Specifies one the following trace options:

default

Use the settings that are configured on the adapter command line in the map. This is the default option.

on

Turn on adapter tracing for all adapters, including child RUN maps.

off

Turn off adapter tracing. This option overrides the option that is defined in the map.

-action

Specifies the action to take on the trace file:

create

Write all trace information to a new trace file.

append

Append all trace information to an existing file, or create a new trace file if none exists.

-tracefile tracefile_name

Specifies the name and path to a trace file.

Related reference

- [Syntax conventions](#)

Logging command

Use the logging command to set the logging flags for the Launcher, the Resource Manager, or the Connection Manager.

```
{launcher.bat | launcher.sh}
[-setlogging launchername systemname -logtype {launcher | resmgr | conmgr}
 -logoptions {[to | tx],[do | dx],[wo | wx],[fo | fx],[io | ix],[eo | ex]} ]
```

-setlogging launchername

The name of the Launcher that is defined in the mgmtconsole.bin file. The *launchername* variable identifies the host name, port, user name, and password of a particular Launcher definition.

systemname

The name of the system or group of systems running under one Launcher process. The *systemname* is defined with the Launcher Administration tools.

-logtype

Enables logging for one of the following resources:

launcher

The logging options apply to the Launcher.

resmgr
The logging options apply to the Resource Manager.

conmgr
The logging options apply to the Connection Manager.

-logoptions
Specifies the logging options to use. Multiple options must be delimited by commas.

to	Log trace information.
tx	Do not log trace information. This is the default.
do	Log debugging information.
dx	Do not log debugging information. This is the default.
fo	Log fatal error messages.
fx	Do not log fatal error messages. This is the default.
eo	Log error messages.
ex	Do not log error messages. This is the default.
io	Log informational messages.
ix	Do not log informational messages. This is the default.
wo	Log warning messages.
wx	Do not log warning messages. This is the default.

Related reference

- [Syntax conventions](#)

List system information command

Use this command to list information about Launcher systems and Launcher HTTP Listeners. System information includes system files, components within each system, trace settings for each component in a system, and log settings for the system. Listener information includes ports, certificates, logging, and audit enablement.

```
{launcher.bat | launcher.sh}
[-list launchername [[-lsnr listener_name] | [systemname [systemfilename [componentname]]]] [-trace] [-logging]]
-list launchername
    Lists all the systems, the system files in each system, and the components in each system file. launchername is the Launcher that is defined in the mgmtconsole.bin file. The launchername variable identifies the host name, port, user name, and password of a particular Launcher definition.
-systemname
    Lists the system files in the specified system and the components in each of the system files. The systemname is the name of the system or group of systems that is running under one Launcher process. The systemname is defined with the Launcher Administration tools. The systemname is optional, and is mutually exclusive with the -lsnr option.
-systemfilename
    Lists the components in the specified system file. The systemfilename is the name of an individual system file (.msl) that is running under the specified system and Launcher.
-lsnr listener_name
    Lists the ports, certificates, logging level, and audit enablement of the specified HTTP listener. An HTTP Listener is defined with the Launcher Administration. The -lsnr listener_name is optional, and is mutually exclusive with the systemname option.
-trace
    Lists the trace settings for the specified component.
-logging
    Lists the logging settings for the Launcher, Resource Manager, and Connection Manager for the system, if specified. Otherwise, lists the logging settings for all systems. The systemfilename and componentname are not required.
```

Related reference

- [Syntax conventions](#)

Launcher Administration command line syntax

The Launcher Administration commands configure the Launcher service and the launchers that are created by the Launcher service. You can use Launcher Administration commands to add and remove HTTP Listeners and to configure deployment directories, users and user access rights, listening ports, and Launcher service properties.

Use the Launcher **launcheradmin.bat** command in a Microsoft Windows environment and the **launcheradmin.sh** command in a UNIX environment.

On UNIX and z/OS® systems, use quotation marks (" ") to enclose multiple parameters that are delineated by semicolons (;). For example, specify **launcheradmin.sh -adduser "user:admin;login:admin;pwd:admin"**

To display the Launcher Administration graphical interface, invoke the command without any parameters. To display help for the command line options, invoke the command with the **-help** option.

```
{launcheradmin.bat | launcheradmin.sh}
[-help]
[-list [general | access | deploydirs | advanced | options | all]]
[-manual | -auto]
[-single | -separate]
[-lport listening_port]
[-prange startport:endport]
[-mrc resource_config_file]
[-ini ini_file_path]
[-status]
[-firewall {enable | disable}]
[-fsport server_port]
[-fcport client_port]
[-restart {enable | disable}]
[-retry retry_number]
[-rtimeout minutes]
[addlsnr name:listener_name[:mode:{enabled | disabled}][;httpport:port][;lnchport:port]
 [;httpcert:certificate][;lnchcert:certificate][;loglevel:{none | standard | verbose}]
 [;audit:{enabled | disabled}][
[-remlsnr name:listener_name]
[addprop property_name:property_value [:property_name:property_value; ...]]
[-delprop property_name[:property_name;... ] ]
[-deluser user:user_name;login:login_name;pwd:password[,access:{[gss|rss],[gpr|rpr],[gm|rm]}]
[-deluser user_name ]
[-addir | -raddir deployment_directory]
[-addir deployment_directory -asf system_file
 [-pd {[ini:ini_file];[rm:{local | global}];[mrc:resource_config_file]}}
 [-pf {[name:name];[desc:description];[ini:ini_file];[rm:{local | global}];[mrc:mrcfile]};[as:{yes | no}]]]
[-addir deployment_directory -dsf system_file[;system_file;...]]
[-addir deployment_directory -sys system_file {-af | -df} file1[,file2;...]
[-import properties_file]
[-export properties_file]
```

- [List the configured administration options](#)

The Launcher Administration **-list** command displays the options that are configured in the launcheradmin.bin file. You can display all configured options or a subset of the options. The subsets group the options according to the tabs where the options are configured in the graphical Launcher Administration interface.

- [General options](#)

The General options control how systems start and run, the ports and the configuration files that are used by the Launcher service, and the display of the Launcher service status. These options correspond to the options that you configure on the General tab of the graphical Launcher Administration interface.

- [Access options](#)

The Access options add, delete, or edit the permissions of Launcher users. These options correspond to the options that you configure on the Access tab of the graphical Launcher Administration interface.

- [Deployment directories options](#)

The deployment directories options add, edit, and remove directories from the list of deployment directories, add systems to the list of systems, and configure directory and file properties. These options correspond to the options that you configure on the Deployment Directories tab of the graphical Launcher Administration interface.

- [Advanced options](#)

The Advanced options specify system properties, such as java.rmi.server.hostname, that are to be added or removed from the list of properties that are initialized during Launcher service startup. These options correspond to the options that you configure on the Advanced tab of the graphical Launcher Administration interface.

- [HTTP Listener options](#)

Options control automatic restarts when the Launcher fails, and whether clients connect to the Launcher service through a firewall. You also add and remove HTTP Listeners by using Options. These options correspond to the options that you configure on the Options page of the graphical Launcher Administration interface.

- [HTTP Listener startup configuration](#)

An HTTP Listener routes HTTP requests from external clients to maps in the Launcher and returns a response to the external client. Use the Launcher Administration utility Options page and select Listeners to add, remove, and update HTTP Listeners.

- [Import and export properties options](#)

The **-import** option imports properties from an XML file and updates the launcheradmin.bin file. The **-export** option creates an XML properties file by exporting all of the launcheradmin.bin file properties.

Related reference

- [Syntax conventions](#)
- [Running the setup script](#)

List the configured administration options

The Launcher Administration **-list** command displays the options that are configured in the launcheradmin.bin file. You can display all configured options or a subset of the options. The subsets group the options according to the tabs where the options are configured in the graphical Launcher Administration interface.

```
{launcheradmin.bat | launcheradmin.sh}
[-list [general | access | deploydirs | advanced | options | all]]
```

-list

Lists the specified options that are configured in the launcheradmin.bin file. If you invoke **-list** without specifying any additional options, it is equivalent to specifying **-list all**.

```
general          Lists only the options that are configured on the General tab of the graphical Launcher Administration interface.  
access           Lists only the options that are configured on the Access tab of the graphical Launcher Administration interface.  
deploydirs       Lists only the options that are configured on the Deployment Directories tab of the graphical Launcher Administration interface.  
advanced         Lists only the options that are configured on the Advanced tab of the graphical Launcher Administration interface.  
options          Lists only the options that are configured on the Options of the graphical Launcher Administration interface.  
all              Lists all of the options that are configured on the all tabs of the graphical Launcher Administration interface.
```

Related reference

- [Syntax conventions](#)

General options

The General options control how systems start and run, the ports and the configuration files that are used by the Launcher service, and the display of the Launcher service status. These options correspond to the options that you configure on the General tab of the graphical Launcher Administration interface.

```
{launcheradmin.bat | launcheradmin.sh}  
[-manual | -auto]  
[-single | -separate]  
[-lport listening_port]  
[-prange startport:endport]  
[-mrc resource_config_file ]  
[-ini ini_file_path]  
[-status]  
  
-manual          Specifies that manual startup of systems is required.  
-auto            Specifies that systems and compound systems start automatically as part of the Launcher service startup.  
-single          Specifies that all systems and compound systems are to run under one Launcher process.  
-separate        Specifies that each system and compound system is to run in a separate process.  
-lport listening_port  
    Specifies a listening port on which the Launcher service listens for client connection requests from the Management Console and the Launcher Administration graphical and command line tools. When it listens for the Launcher Monitor, the Launcher service adds 1 to the value of the listening port that you specify. For example, the default listening port is 5015. The Launcher service adds 1 to the value and listens for the Launcher Monitor on port 5016.  
-prange startport:endport  
    Specifies the range of ports to be used by the launcher processes that are started by the Launcher service. Because the Status, Control and Monitor ports are included in this range, the minimum range is three ports.  
-mrc resource_config_file  
    Specifies the name of the resource configuration file that is used by the Launcher. The resource configuration file resolves the resource aliases that are used in maps.  
-ini ini_file_path  
    Specifies the full path to the Launcher initialization file. If the path contains spaces, enclose the path in quotation marks (""). By default, the initialization file is the config.yaml file in the installation directory.  
-status          Specifies the status of the Launcher service. The status is either Running or Stopped. When the Launcher service is stopped, any options that you set are saved in the launcheradmin.bin file and used the next time the Launcher service starts. When the Launcher service is running, the options that you set dynamically update the settings of the Launcher service. If the Launcher service is running and you attempt to set options that cannot be updated dynamically, the command line returns an error.
```

Related reference

- [Syntax conventions](#)

Access options

The Access options add, delete, or edit the permissions of Launcher users. These options correspond to the options that you configure on the Access tab of the graphical Launcher Administration interface.

```
{launcheradmin.bat | launcheradmin.sh}  
[-adduser user:user_name;login:login_name;pwd:password [:access:{[gss|rss],[gpr|rpr],[gm|rm]}]}  
[-deluser user_name]
```

On UNIX and z/OS® systems, use quotation marks (" ") to enclose multiple parameters that are delineated by semicolons (;).

-adduser user:*user_name*
 Adds the specified user. If the user already exists, updates the user configuration with the specified options.

login:*login_name*
 Specifies the login name for the user. This keyword is required to add a new user and optional to edit an existing user configuration.

pwd:*password*
 Specifies the password for the user. This keyword is required to add a new user and optional to edit an existing user configuration.

gss|rss
 Grants (gss) or revokes (rss) permission for the user to start and stop the Launcher. By default, permission is revoked.

gpr|rpr
 Grants (gpr) or revokes (rpr) permission for the user to pause and resume the Launcher. By default, permission is revoked.

gm|rm
 Grants (gm) or revokes (rm) permission for the user to monitor the Launcher. By default, permission is revoked.

-deluser *user_name*
 Deletes the specified user.

Related reference

- [Syntax conventions](#)

Deployment directories options

The deployment directories options add, edit, and remove directories from the list of deployment directories, add systems to the list of systems, and configure directory and file properties. These options correspond to the options that you configure on the Deployment Directories tab of the graphical Launcher Administration interface.

```
{launcheradmin.bat | launcheradmin.sh}
[-addir | -rddir deployment_directory]
[-eddir deployment_directory -ASF system_file
 [-pd {[ini:ini_file];[rm:{local | global}];[mrc:resource_config_file]}}
 [-pf {[name:name];[desc:description];[ini:ini_file];[rm:{local | global}];
 [mrc:mrcfile];[as:{yes | no}]}}
 [-eddir deployment_directory -DSF system_file[;system_file;...]]
 [-eddir deployment_directory -SYS system_file {-af | -df } file1[;file2;...]
```

On UNIX and z/OS® systems, use quotation marks (" ") to enclose multiple parameters that are delineated by semicolons (;).

-addir *deployment_directory*
 Adds the specified directory to the list of deployment directories.

-rddir *deployment_directory*
 Removes the specified directory from the list of deployment directories.

-eddir *deployment_directory*
 Edits the specified deployment directory.

-ASF *system_file*
 Adds the specified system file to the list of system files that should be launched in a separate launcher process. The system file is part of the directory specified by the -eddir keyword.

-pd
 Configures the properties of the directory that is specified by the -eddir keyword. Separate each keyword:value pair with a semicolon (;).

ini:*ini_file*
 Specifies the path to the initialization file. By default, it is the initialization file that is specified in the General options by the **-ini ini_file_path** option.

rm:{local | global}
 Specifies whether the resource manager is local or global. By default, the resource manager is local.

mrc:*resource_config_file*
 Specifies the path to the resource configuration file. By default, the resource configuration file path is blank.

-pf
 Configures the properties of the specified system file. Separate each keyword:value pair with a semicolon (;).

name:*name*
 Specifies the name of the system file.

desc:*description*
 A brief description of the system file.

ini:*ini_file*
 Specifies the path to the initialization file. By default, it is the initialization file that is specified in the General options by the **-ini ini_file_path** option.

rm:{local | global}
 Specifies whether the resource manager is local or global. The default value is local.

mrc:*mrcfile*
 Specifies the path to the resource configuration file. By default, it is blank.

as:{yes | no}
 Specifies whether (yes) or not (no) the system is to start automatically. The default value is no.

-DSF *system_file*[;*system_file*;...]
 Specifies one or more system files to be deleted. Separate multiple system file names with a semicolon (;).

-SYS *system_file*
 Specifies the system file to which files are added or from which files are deleted. A system file can launch multiple .msl files.

```

-af file1[file2;...]
    Specifies the full path to one or more .msl files that are to be added to system file and launched as part of the launcher process.
-df file1[file2;...]
    Specifies the full path to one or more .msl files that are to be deleted from the system file.

```

Related reference

- [Syntax conventions](#)

Advanced options

The Advanced options specify system properties, such as java.rmi.server.hostname, that are to be added or removed from the list of properties that are initialized during Launcher service startup. These options correspond to the options that you configure on the Advanced tab of the graphical Launcher Administration interface.

```
{launcheradmin.bat | launcheradmin.sh}
[-addprop property_name:property_value [property_name:property_value; ...]]
[-delprop property_name[;property_name;... ]]
```

On UNIX and z/OS® systems, use quotation marks (" ") to enclose multiple parameters that are delineated by semicolons (;).

```
-addprop property_name:property_value
    Specifies the name and the value of the property that is to be added. If the property exists, it is updated. Otherwise, the property is added.
-delprop property_name
    Specifies the name and the value of the property that is to be deleted. If the property does not exist, the command line returns an error.
```

Related reference

- [Syntax conventions](#)

HTTP Listener options

Options control automatic restarts when the Launcher fails, and whether clients connect to the Launcher service through a firewall. You also add and remove HTTP Listeners by using Options. These options correspond to the options that you configure on the Options page of the graphical Launcher Administration interface.

```
{launcheradmin.bat | launcheradmin.sh}
[-firewall {enable | disable}]
[-fsport server_port]
[-fcport client_port]
[-restart {enable | disable}]
[-rretry retry_attempts]
[-rrtimeout minutes]
[-addlsnr name:listener_name[;mode:{enabled | disabled}][;httpport:port][;lnchport:port]
[;httpcert:certificate][;lnchcert:certificate][;loglevel:{none | standard | verbose}]
[;audit:{enabled | disabled}]
[-remlsnr name:listener_name]

-firewall {enable | disable}
    Specifies whether to enable a firewall configuration. In a firewall environment, clients like the Management Console use a designated server port to connect to the Launcher service. The Launcher service opens up a listening port to establish the initial connection. All the client requests for service are exported to the server port and all services are provided through this port. By default, the firewall is disabled.
-fsport server_port
    Specifies the server-port value of the firewall-specific server port that is to be used by the Launcher service.
-fcport client_port
    Specifies the client-port value of the firewall-specific client port that is to be used by the Launcher service.
-restart {enable | disable}
    Enables or disables automatic restart when the Launcher fails. Restart is enabled by default.
    This option is valid only on distributed platforms. It is not supported on USS.
-rretry retry_attempts
    Specifies how many times the Launcher is to attempt to restart after a failure. The default value is two restart attempts. If the specified number of restart attempts has not occurred when the time specified by -rrtimeout elapses, the Launcher continues to retry until it reaches the specified number of attempts.
-rrtimeout minutes
    Specifies how long, in minutes, the Launcher is to attempt to restart after a failure. The default value is 10 minutes. There are five minutes between each restart attempt.
-addlsnr name:listener_name
    Specifies the name of the HTTP Listener to add or edit. If the specified listener does not exist, -addlsnr creates it and sets the specified properties. If the listener exists, -addlsnr edits the specified properties:
        mode:{enabled | disabled}
            Specifies whether the HTTP Listener starts when the Launcher starts. By default, the listener is disabled and does not start with the Launcher.
        httpport:port
            Specifies the port where the client posts an HTTP request. The default port number is 5017.
        lnchport:port
            Specifies the HTTP Listener port that connects to the Launcher to trigger a map. The default port number is 5018.
        httpcert:certificate
            Specifies the name in the key store of the certificate from the Certificate Signing Authority, used to for secure communication with clients.
```

```

Inchcert:certificate
    Specifies the name in the key store of the certificate that enables secure communication with maps. The key store is set in the config.yaml file in the
    installation directory.

loglevel:{none | standard | verbose}
    Specifies the trace setting for the HTTP Listener. By default, the listener is not traced (none).

audit:{enabled | disabled}
    Enables or disables auditing of HTTP Listener client requests and connections. Auditing is disabled by default.

-remlsnr name:listener_name
    Deletes the specified HTTP Listener.

```

Related tasks

- [Configuring automatic restart](#)

Related reference

- [Syntax conventions](#)

HTTP Listener startup configuration

An HTTP Listener routes HTTP requests from external clients to maps in the Launcher and returns a response to the external client. Use the Launcher Administration utility Options page to add, remove, and update HTTP Listeners.

HTTP Listener settings

Name

A unique name of up to 25 characters to represent a listener.

Mode

When enabled, the listener starts automatically when the Launcher starts. By default, the listener is disabled.

HTTP port

The port that the listener uses to communicate with external clients. The listener monitors the HTTP port for incoming HTTP requests and responds to the external client through the HTTP port. Each listener must use a unique HTTP port. If you configure multiple listeners to use the same HTTP port, only one listener starts when the Launcher starts. The default HTTP port is 5017.

Launcher port

The port that the listener uses to communicate with the Launcher. Each listener must use a unique Launcher port. If you configure multiple listeners to use the same Launcher port, only one listener starts when the Launcher starts. The default Launcher port is 5018.

HTTP certificate

Specifies the server certificate used for SSL communication with external HTTP clients on the client port. If you omit it, SSL communication between the HTTP Listener and external HTTP clients is disabled.

Launcher certificate

Specifies the server certificate used for SSL communication with HTTP adapters on the Launcher port. If you omit it, SSL communication between the HTTP Listener and maps is disabled.

Log level

The trace setting for the HTTP Listener. The Launcher creates a unique trace file name by appending the current date, timestamp, and computer host name to the HTTP Listener name (for example, lsnr_a_socket07-06-02-11-14-PM_myhostname.log). Logs are created in the installation directory on Windows systems and in the \bin directory on UNIX systems.

None

Tracing is not enabled for the HTTP Listener. This is the default setting.

Standard

Logs the HTTP Listener activity. The log file name is *listenername_timestamp_hostname.log*.

Verbose

Logs the HTTP Listener activity and socket activity. The log file name is *listenername_socket_timestamp_hostname.log*.

Audit

Enables or disables an audit log of Transformation Extender map executions and each adapter connection and external client connection to the listener. The audit log is in XML format that can be queried with XSLT transformations. The audit log schema is LHL-Audit.xsd, installed in the Transformation Extender installation directory on Windows systems or the /bin directory on UNIX systems. Auditing is disabled by default. There is one audit log per listener.

Import and export properties options

The **-import** option imports properties from an XML file and updates the launcheradmin.bin file. The **-export** option creates an XML properties file by exporting all of the launcheradmin.bin file properties.

```
{launcheradmin.bat | launcheradmin.sh}
[-import properties_file]
[-export properties_file]
```

-import properties_file

Specifies an XML file that contains properties to be added to the launcheradmin.bin file. If the launcheradmin.bin file exists, the existing properties are updated and saved. If the launcheradmin.bin file does not exist, a new launcheradmin.bin file is created with the properties from the XML file.

If the Launcher service is running, it is updated with the information from the XML file. If the Launcher service is running and you attempt to set options that cannot be updated dynamically, the command line returns an error.

The following properties cannot be updated dynamically:

- Automatic startup
- Separate launcher processes
- Listening ports
- Port Range
- Remove user
- Advanced options
- Firewall options

-export *properties_file*

Creates an XML properties file that is compliant with the LauncherAdmin.xsd schema file. You can create multiple customized properties files to use in different environments.

Management Console command line syntax

Use the Management Console options to create the Launcher definitions and firewall and override configurations. Each launcher definition establishes a connection to one Launcher service.

Use the mgmtconsole.bat command in a Microsoft Windows environment and the mgmtconsole.sh command in a UNIX® environment.

To display the Management Console graphical interface, invoke the command without any parameters. To display help for the command line options, invoke the command with the -help option.

```
{mgmtconsole.bat | mgmtconsole.sh}
[-help]
[-list [launchers | options | all]]
[-add name:launcher_name;host:host_name;port:port;login:login_name;password:pwd;interval:refresh_interval]
[-delete launcher_name]
[-override name:launcher_name;oc:{enable | disable};host:host_name;sp:status_port;
 cp:control_port]
[-firewall fc:{enable | disable};fp:firewall_port]
[-import properties_file]
[-export properties_file]

-list [launchers | options | all]
    Displays the specified options that are configured in the mgmtconsole.bin file. If you invoke -list without any other options, it is equivalent to the -list all command.

-launchers
    Displays the following Launcher information that is configured in the mgmtconsole.bin file:
        • Launcher name
        • Host name
        • Port
        • Refresh interval
        • Login and password

-options
    Displays the Override Configuration and Firewall options that are configured in the mgmtconsole.bin file.

-all
    Displays all of the Launcher information that is configured in the mgmtconsole.bin file.

-add
    Adds the specified Launcher definition to the mgmtconsole.bin file. If the Launcher definition already exists, updates the definition.

        name:launcher_name
            Specifies the name of the Launcher service that is to be added.
        host:host_name
            Specifies the Launcher service host name or IP address that is to be added to the mgmtconsole.bin file. The IP address can be in either IPv4 format or IPv6 format.
        login:login_name
            Specifies the login name for the Launcher service to be added mgmtconsole.bin file. The login name is the name that the Management Console uses to connect to the Launcher service.
        port:port
            Specifies the port on which the Launcher service is to listen. The default port is 5015.
        refresh:refresh_interval
            Specifies the time interval, in seconds, that is used to refresh the system statistics that are displayed by the Management Console. The default refresh interval is five seconds.

-delete launcher_name
    Deletes the specified Launcher from the list of Launcher definitions in mgmtconsole.bin.

-override
    Overrides the Launcher service that is defined in the Management Console and enables a direct connection to the specified Launcher.

        name:launcher_name
            Specifies the name of the Launcher service that is to be overridden.
        oc:{enable | disable}
            Enables or disables the override configuration for the launcher definition.
        host:host_name
            Specifies the IP address or host name of the computer where the Launcher is running. The default host name is localhost.
        cp:control_port
```

-firewall fc:{enable | disable};fp:*firewall_port*
 Enables or disables a firewall configuration for the Management Console and specifies a client port to be used by a Management Console that is behind the firewall.
 -import *properties_file*
 Specifies an XML file that contains properties to be imported to the mgmtconsole.bin file. If the mgmtconsole.bin file exists, the existing properties are updated and saved. If the mgmtconsole.bin file does not exist, a new mgmtconsole.bin file is created with the properties from the XML file.
 -export *properties_file*
 Creates an XML properties file that is compliant with the MgmtConsole.xsd schema file. You can create multiple properties files to use in different environments.

Related reference

- [Syntax conventions](#)
 - [Running the setup script](#)
-

HTTP Listener command line syntax

You can use the **httplsnr** command to configure, start, and stop the Launcher HTTP Listener.

```

httplsnr
[[-?]{config_file {-view | -stop} | {config_file {-edit | -start}
  [-clntport HTTP_port_number]
  [-lnchport Launcher_port_number]
  [-clntcert client_request_certificate]
  [-lnchcert HTTP_adapter_request_certificate]
  [-lpolicy {RR | FCFS | RND}]
  [-authclients {enable | disable}]
  [-adminuser username]
  [-adminpswd password]
  [-adduser username%path%password]
  [-deluser username%path]
  [-trace file_name | -traceoff]
  [-verbose file_name | -verboseoff]
  [-audit file_name | -auditoff]
  [-noprompt]})]

httplsnr
  When invoked without parameters, displays the list of httplsnr command options.

? 
  The httplsnr ? command displays the list of monitoring, auditing, and tracing commands that correspond to the HTTP Listener Status controls that are available from the Management Console or the http://Launcher_host_address:http_port/admin/status URL.

config_file
  The unique name of the configuration file used to start and stop an HTTP Listener.

-view
  Displays the settings in the specified configuration file.

-stop
  Stops the HTTP Listener specified by the configuration file.

-edit
  Sets or overrides the specified settings in the configuration file.

-start
  Starts the HTTP Listener specified by the configuration file. If the configuration file exists, the -start option saves or edits the options that are specified on the command line. If the configuration file does not exist, the -start option creates the file and saves the specified options in it.

-clntport
  Specifies the HTTP port number for communicating with the client, and saves it in the configuration file.

-lnchport
  Specifies the port number for communicating with the Launcher, and saves it in the configuration file.

-clntcert
  Specifies the server certificate used for SSL communication with HTTP clients on the client port. The -clntcert is optional. If you omit it, SSL communication with HTTP clients is disabled.

-lnchcert
  Specifies the server certificate used for SSL communication with HTTP adapters on the Launcher port. The -lnchcert is optional. If you omit it, SSL communication with HTTP clients is disabled.

-lpolicy {RR | FCFS | RND}
  Specifies how to load-balance HTTP requests among Launchers, and saves the load-balancing method in the configuration file. Load-balancing is in effect when there are multiple Launchers on one listener or multiple watches on the same Launcher.

  RR
    Round robin
  FCFS
    First come, first served
  RND
    Random

```

-authclients {enable | disable}
 Specifies whether browser requests should be authenticated, and saves it in the configuration file. By default, authentication is disabled.

-adminuser *username*
 Specifies the user ID to authenticate the connection to Launcher Administration, and saves it in the configuration file. The user ID is a maximum of 31 bytes.

-adminpswd *password*
 Specifies the password for the adminuser, and saves it in the configuration file. The password is a maximum of 31 bytes.

-adduser *username%path%password*
 Specifies the name, path, and password of a new Launcher user to add to the configuration file. The values are delimited by the percentage symbol (%).

username
 The ID that the Launcher uses to connect to the HTTP listener. The user name is a maximum of 31 alphabetic or numeric characters. A user name can have more than one path, and each *username%path* combination represents a unique Launcher user.

path
 The URI where the Launcher user account is to be created. The path is a maximum of 127 bytes.

password
 The password for the Launcher user ID. The password is a maximum of 31 alphabetic or numeric characters.

-deluser *username%path*
 Specifies the name and path of a Launcher user to remove from the configuration file. The values are delimited by the percentage symbol (%). The name and password are a maximum of 31 bytes.

-traceoff
 Specifies the HTTP Listener trace file name and enables tracing, or disables HTTP Listener tracing.

-verbose *file_name*
-verboseoff
 Specifies the HTTP Listener log file name and enables logging of HTTP Listener socket activity, or disables logging of HTTP Listener socket activity.

-audit *file_name*
-auditoff
 Specifies the HTTP Listener audit file name and enables auditing of HTTP client requests and connections, or disables auditing of HTTP client requests and connections.

-noprompt
 Runs the **httpsnr** command without returning any output to the console.

Multiple processes

The WebSphere® Transformation Extender Launcher has a multiple-processes feature.

This feature enables the Launcher to run multiple Launcher processes, each containing one or more systems (.msl files), within a single installation. It will run each system or group of systems in a separate Launcher process allowing you to add, remove and modify a Launcher process without affecting other running Launcher processes. Running the Launcher in separate Launcher processes also offers a more scalable and robust run-time environment. The WebSphere Transformation Extender Launcher can be configured to run using the **CompoundSystem** approach (all systems run in a single Launcher process), or to run each system (.msl file) or group of systems in multiple Launcher processes.

The documentation describes the different types of configurations available for running multiple processes within a single installation and how each configuration type affects system processing.

The WebSphere Transformation Extender Launcher multiple-processes feature is also available to use with the z/OS® operating system on IBM® z/Architecture® platforms.

Note: If you will be running systems specifically through the z/OS native administration interface (NAI), the WebSphere Transformation Extender Launcher multiple-processes feature allows you to run multiple Launcher processes, each containing one or more systems (.msl files), within a single installation of the Launcher as described in the "[z/OS Launcher](#)" documentation.

- [Activities summary](#)
- [Installation](#)
- [Using resource management](#)
- [Using Launcher Administration to run systems as separate processes](#)

The multiple-processes feature of the Launcher runs an .msl system file or group of .msl system files as separate Launcher processes. Use the Launcher Administration application to configure separate Launcher processes.

- [Managing the Launcher process](#)

Activities summary

To run multiple processes within a single installation:

- change resource management settings through the Resource Manager section of the initialization file depending on the way in which the Launcher will be running the Launcher processes
- change settings through the various windows within the Launcher Administration to configure the Launcher to use this feature
- manage the Launcher process by:
 - starting the Java™ Launcher
 - controlling the Launcher and viewing the status of the system either through the Management Console or by using command line utilities
 - viewing map execution information and taking snapshots of Launcher activity through the Launcher Monitor
 - analyzing the contents of debug trace and log files

Installation

In Windows environments, the Launcher installation setup program installs WebSphere® Transformation Extender Launcher as a service with **Manual** as the default value for **Startup Type** and **LocalSystem** as the default value for **Log On As**. An example of the service name that you might see is **TX n.n Launcher**.

See the [release notes](#) for more information about the installation.

Note: The Launcher run on UNIX and z/OS® does not run under a service.

Using resource management

The Resource Manager is a subsystem of the Launcher that controls those resources that do not have inherent resource management such as databases. File adapters are the most common adapters that require resource management. You can change resource management settings, which will affect how the Launcher will run.

There are two types of resource management:

- local - used within a single Launcher process.
- global - used across all of those Launcher processes that have been configured to use this type of resource management.
- [Global resource management](#)
- [Launcher systems run options](#)

Global resource management

Use global resource management, also referred to as central resource management, when running a system or group of systems in separate Launcher processes and the systems can share resources. The extra overhead of central resource management is minimal. The Resource Manager table requires an operating system-level lock rather than a process-level lock while accessing the Resource Manager. You should use central resource management if a system or group of systems running in separate Launcher processes start to have map errors that produce the error messages such as: source not available or failed to put data on output.

When running systems in multiple Launcher processes, central resource management can be turned on or off for a select Launcher process. Central resource management enables these different processes from the same installation to manage shared resources.

Launcher systems run options

The following list presents four data transformation scenarios and the various ways in which the Launcher can run a system or group of systems

- in a single Launcher process, as a Compound System
- in separate Launcher processes, each with localized resource management
- in separate Launcher processes, all using centralized resource management
- in separate Launcher processes, with a combination of centralized and local resource management

Using Launcher Administration to run systems as separate processes

The multiple-processes feature of the Launcher runs an **.msl** system file or group of **.msl** system files as separate Launcher processes. Use the Launcher Administration application to configure separate Launcher processes.

- [Benefits of using separate Launcher processes](#)
It is simpler to maintain and configure a system or group of systems that run with separate Launcher processes.
- [General tab](#)
- [Access tab - User Properties dialog](#)
- [Adding and removing processes and system files with the Deployment Directories tab](#)
Use the Launcher Adminstration application to add or remove a Launcher process or .msl system file when a system or group of systems is configured to run as separate processes. In the Deployment Directories tab, click Edit to work with the processes and .msl system files.
- [Configuring Launcher processes in the File List dialog](#)
- [Configuring Launcher processes to start automatically](#)
- [Modifying systems while the Launcher is running](#)
- [Configuring the initialization file](#)

Related tasks

- [Adding and removing processes and system files with the Deployment Directories tab](#)

Related reference

- [Restrictions when modifying deployment directories and processes](#)
- [Benefits of using separate Launcher processes](#)
- [Selecting separate Launcher processes](#)

Benefits of using separate Launcher processes

It is simpler to maintain and configure a system or group of systems that run with separate Launcher processes.

- The run-time environment is more robust. Problems that occur in maps or adapter connections in one Launcher process do not cause problems in other Launcher processes.
- You can add and reconfigure Launcher processes without stopping other Launcher processes.
- You can use systems within systems for a single .msl Launcher system file for logical processes definitions.
- You can group systems to run in a separate Launcher process and start and stop them without having to shut down the Java™ Launcher.

Related tasks

- [Adding and removing processes and system files with the Deployment Directories tab](#)
- [Modifying systems while the Launcher is running](#)

Related reference

- [Selecting separate Launcher processes](#)
- [Restrictions when modifying deployment directories and processes](#)

General tab

The General tab of the Launcher Administration window has controls that you can set to enable the Launcher to use the multiple-processes feature.

The following list presents the controls, used with the multiple-process feature, along with their location in the Launcher Administration window:

- **Automatic startup**
Enabling **Automatic startup** gives you the ability to start up a Launcher process automatically. For information about using this option, see "[Configuring Launcher Processes to Start Automatically](#)".
- **Separate Launcher system processes**
Indicates that the Launcher will be run in different Launcher processes and that one or more system files can be configured for each process.
- **Initialization File:**
Identifies the initialization file, which contains configuration settings for WebSphere® Transformation Extender.
Default is *install_dir\config.yaml*.
- **Status:**
Displays the status of the Java™ Launcher.
- [Selecting separate Launcher processes](#)
- [Connection ports](#)

Selecting separate Launcher processes

When selecting Separate Launcher processes, configuration for different initialization files can be set for each Launcher process. Also, checking both **Automatic startup** and **Separate Launcher processes** adds the **Automatic Start** property in the Properties window for each process. The **Automatic Start** property has two options:

- **Yes**
 - Enables the specific Launcher process on which the **Yes** option is set to start up automatically.
- **No**
 - Prevents the specific Launcher process on which the **No** option is set from starting up automatically.

When the **Separate Launcher processes** checkbox is not checked, the Launcher runs all systems as a Compound System.

Related tasks

- [Adding and removing processes and system files with the Deployment Directories tab](#)

Related reference

- [Restrictions when modifying deployment directories and processes](#)
- [Using Launcher Administration to run systems as separate processes](#)
- [Benefits of using separate Launcher processes](#)

Connection ports

Three consecutive ports for each separate Launcher process are required. The Launcher automatically assigns them based on the first port entered in the port range.

For information about the listening ports, see "[Connection Ports](#)".

Access tab - User Properties dialog

Through the Access tab of the Launcher Administration window, you can invoke the User Properties window to add new or edit existing users and access permissions for the Management Console and the Launcher Monitor by clicking **Add** or **Edit**.

After clicking **Add** or **Edit**, the User Properties window will appear with default access permission set to **ALL Systems** under the **Systems** column heading. You can change the permissions for **Start/Stop**, **Pause/Resume** and **Monitor** to **Grant**.

To change the permissions in the User Properties window

1. Click on the permissions under one of the permission category columns (**Systems**, **Start/Stop**, **Pause/Resume**, and **Monitor**) to expand the drop-down menu.
2. Select the permission you want from the drop-down list.
3. To apply the changes, click **OK**.
4. You can repeat these steps for any of the permission categories you want to change.

Adding and removing processes and system files with the Deployment Directories tab

Use the Launcher Adminstration application to add or remove a Launcher process or .msl system file when a system or group of systems is configured to run as separate processes. In the Deployment Directories tab, click **Edit** to work with the processes and .msl system files.

In the Deployment Directories tab in the Launcher Administration application, select **Separate Launcher processes** to enable the **Edit** button. Click **Edit** to display the File List window.

Use the File List window to:

- add or remove individual Launcher processes.
 - [Adding a Launcher process](#)
 - [Removing a Launcher process](#)
- add or delete individual .msl system files for a specific Launcher process.
 - [Adding .msl system files to a Launcher process](#)
 - [Deleting .msl system files from a Launcher process](#)

Note: The Launcher process that is listed in the File List window under the Properties tab is treated as if it were the *main* .msl system file; it must exist before others can be added and it cannot be deleted until all additional systems are deleted.

Related reference

- [Selecting separate Launcher processes](#)
- [Restrictions when modifying deployment directories and processes](#)
- [Benefits of using separate Launcher processes](#)

Adding a Launcher process

1. Click **Add** in the view under the Properties tab.
The Select window appears.
2. Click on the .msl system files you want to deploy.
To select multiple files, click Ctrl and click on the files you want. Your selected files will be highlighted.
3. Click **OK**.
The Select window closes and the File List window reappears with the selected files listed.
4. Click **OK**.
The File List window closes and the window under the Deployment Directories tab in the Launcher Administration application reappears.

Removing a Launcher process

1. In the File List window, click the Launcher process that you want to remove.
2. Click **Remove** in the view under the Properties tab.
The Launcher process is removed from the list.

Before you can remove a Launcher process, you must first stop its execution.

Adding .msl system files to a Launcher process

1. In the File List window, click the Launcher process to which you want to add .msl system files.
2. Click **Add Files...** in the **Select additional files...** view under the **Files...** tab.
The Select window opens.
3. Click on the .msl system files you want to deploy in the specified Launcher process.
To select multiple files, click Ctrl and click on the files you want. Your selected files will be highlighted.

4. Click Select.

The Select window closes and the File List window reappears with the selected files listed in the main pane of the **Select additional files...** view.

Deleting .msl system files from a Launcher process

1. In the File List window, click the Launcher process from which you want to delete .msl system files.
2. Click in the main pane of the **Select additional files...** view on the specific .msl system file you want to delete.
3. Click Delete files... in the **Select additional files...** view under the Files... tab.

The .msl system file is removed from the list.

The .msl system file that appears in the File Name property, acting as a control file, defines the main Launcher process. As such, you cannot delete it as described in this procedure, which is only for the additional Launcher system files.

To delete the main Launcher .msl system file, you would have to remove the Launcher process entirely.

Configuring Launcher processes in the File List dialog

When using separate Launcher processes, configure each of them in the File List window. There are two sections of the File List window: the Property (Directory) section and the Property (File) section.

- [Property \(Directory\)](#)
 - [Property \(File\)](#)
-

Property (Directory)

The Property (Directory) enables you to override the default values and configure settings for those specific Launcher processes (.msl files). It allows you to propagate the default settings to those Launcher processes (.msl files) within that directory.

To propagate the Property (Directory) default settings

1. Select the property in the Property (Directory).
2. Right-click on the property.
A menu appears.
3. Select Propagate.
The default settings are copied to those Launcher processes (.msl files) within that directory.

Property (File)

The Property (File) enables you to configure the specific Launcher process to override the default settings.

The following table lists the Launcher process properties along with a description, and default and valid values.

Property (File)	Description
Name	Unique name that will be used for the Management Console and command line utilities to control the Launcher process individually.
Description	Description for the Launcher process.
File Name	Launcher system path and name.
Initialization File	Valid WebSphere® Transformation Extender initialization file. Default value is the global entry on the General tab of the Launcher Administration.
Resource Manager	Configuration value indicating whether to use global or local resource management. Valid values are Global and Local selected from pull-down menu. Default value is Local .
	Global indicates that resource management will allow multiple Launcher processes to share common resources properly.
Resource Configuration	Resource configuration file for this system. Default value is the entry on the General tab of the Launcher Administration.
Automatic Start	This property row appears when Automatic startup is enabled on the General tab of the Launcher Administration window. It allows you to start up this Launcher process automatically. Valid values are Yes and No , which are selected from a pull-down menu. Default value is No .

To apply your system configuration changes

Click OK.

The File List window closes and the window under the Deployment Directories tab in the Launcher Administration application reappears.

Configuring Launcher processes to start automatically

After you have configured the Launcher to run as multiple Launcher processes within a single installation, they are ready to be configured to start automatically. When **Separate Launcher processes** and **Automatic startup** are both selected on the General tab of the Launcher Administration, the **Automatic Start** property on the Properties window appears for each Launcher process under the Properties tab in the File List window.

To configure the Launcher processes to start automatically

1. Navigate to the File List window from the Deployment Directories tab in the Launcher Administration application.
2. In the File List window, click the configured Launcher process that you want to start up automatically.
3. Select Yes from the pull-down menu of the **Automatic Start** property.
Note: You can select Automatic startup if **Separate Launcher processes** is not selected. In this case, the **CompoundSystem** will automatically start when the Java™ Launcher starts.
4. Click OK.

The Launcher processes have been configured to start automatically. After they are started, while the Java Launcher is running, you can modify them through the Launcher Administration application by following the instructions.

Modifying systems while the Launcher is running

While the Java™ Launcher is running, you can make changes to Launcher system files (**.msl**) that are in separate Launcher processes in the Launcher Administration application. You can also add and remove Launcher processes.

Note: Before you can remove a Launcher process, you must first stop its execution.

After the Launcher processes are started, while the Java Launcher is running, the Launcher Administration application allows you to make the modifications but also has some functionality restrictions; some of the options are disabled and grayed out. They are presented in the following topics:

- [Changing file configurations](#)
- [Restrictions when modifying deployment directories and processes](#)
You can add deployment directories or Launcher processes while the Launcher service (or daemon) is running. You cannot remove a deployment directory while the Launcher service is running, and you must stop a Launcher process before you can remove the process.
- [Modifying administration user access](#)

Changing file configurations

While the Launcher service is running, you can change file configurations. The detailed procedures are described in the ["General Tab"](#) documentation.

- Cannot Change Startup, Execution and Connection Settings
 - Under the General tab, the Startup, Execution and Connection sections are disabled.
- Changing Initialization File Configuration Settings
 - You can make changes to the initialization file in the Configuration section.
 - The Apply button is enabled. When you click Apply, changes will take effect after the Java™ Launcher is restarted.
 - When the Java Launcher is started, you will see the **Running** status of the Launcher in the Launcher section of the window under the General tab.

Restrictions when modifying deployment directories and processes

You can add deployment directories or Launcher processes while the Launcher service (or daemon) is running. You cannot remove a deployment directory while the Launcher service is running, and you must stop a Launcher process before you can remove the process.

The detailed procedures are described in [Deployment Directories Tab](#).

- Cannot Remove Deployment Directories
 - Under the Deployment Directories tab, the Remove button is disabled and grayed out so you are unable to remove deployment directories while the Launcher service is running.
- Removing a Launcher Process
 - Under the Deployment Directories tab in the File List window that is invoked by clicking the Edit button, the Remove button is disabled and grayed out if the specific Launcher process is running, and enabled if the specific Launcher process has been stopped. (To remove a Launcher process, it must first be stopped.)
- Adding Deployment Directories
 - However you can still add new deployment directories under the Deployment Directories tab by clicking the Add button.
- Adding Launcher Processes
 - And you can still add new Launcher processes in the File List window by clicking the Add button. (The File List window is invoked by clicking the Edit button.)
 - Modification of a Launcher process that is currently running takes effect after the Launcher is restarted.

Related tasks

- [Adding and removing processes and system files with the Deployment Directories tab](#)

Related reference

- [Using Launcher Administration to run systems as separate processes](#)
 - [Selecting separate Launcher processes](#)
 - [Benefits of using separate Launcher processes](#)
-

Modifying administration user access

While the Java™ Launcher is running, you can modify administration user access. The detailed procedures are described in [Access Tab - User Properties window](#).

- Adding or Editing Administration User Access
 - Under the Access tab, you can add new or edit existing Administration users.
 - The **Edit** button settings will take effect after the Java Launcher is restarted.
 - Cannot Remove Administration User Access
 - Under the Access tab, the **Remove** button is disabled.
-

Configuring the initialization file

Whether using the single process **CompoundSystem** or separate Launcher processes comprising one or more **.msl** system files, you can configure the Launcher initialization file. You can rename it (the default name is **config.yaml**) as well as make modifications to its contents.

Some of the changes you can make to the initialization file include modifications to the following resource manager settings in the **[Resource Manager]** section:

Resource Management Settings

Description

CentralMgmt

Valid values are:

0, local resource management

1, global (centralized) resource management

Default is 0.

TableSize

Size in megabytes for the Resource Manager table.

Default is 10.

If the Launcher Administration is set to run systems in separate Launcher processes and the resource manager setting for a Launcher process is set to **Local**, the **CentralMgmt** entry in the initialization file has precedence over those other settings. Setting **CentralMgmt** to 1 will enable global (centralized) resource management. The default is 0 or local resource management.

The **TableSize** entry is used for both local and global resource management. It determines the initial size of the Resource Manager table before having to re-size it if required. Re-sizing requires all Launchers using the table to re-calculate the entries and might fail if memory cannot be obtained. The value is in megabytes, and the minimum value is 1. The default table size is 10 (10 megabytes).

The following table correlates the size of the Resource Manager table to the number of entries that the table can hold.

Resource Manager Table Size (in megabytes)	Total Entries	Unique Entries (138 bytes average size)
1	5461	4369
5	27306	21845
10	54613	43690
25	136533	109226

Note: There is, at most, one entry for each initialize pending map and one entry for each card for each resource pending map during Launcher processing, depending on the type of resource (file, database, ftp, and so on) and whether it needs WebSphere® Transformation Extender resource management.

When using global resource management, the initial Launcher process that creates the Resource Manager table determines the size for subsequent processes that use global resource management.

Managing the Launcher process

There are several activities you can do to manage the Launcher process. You can manage it by:

- starting the Java™ Launcher
- controlling the Launcher process and viewing the status either through the Management Console or by using command line utilities
- viewing map execution information and taking snapshots of Launcher process activity through the Launcher Monitor
- analyzing the contents of debug trace and log files
- [Starting the Launcher service or Launcher daemon](#)
- [Monitoring systems through the Management Console](#)
- [Shell script and batch file \(launcher.sh and launcher.bat\)](#)
- [Using the Launcher Monitor](#)
- [Debug trace and log files](#)

Starting the Launcher service or Launcher daemon

Start the Launcher daemon or Launcher service before you start the Launcher processes that run within the Management Console or use the **launcher.bat** (Windows systems) or **launcher.sh** (UNIX systems) commands.

Start only one Launcher daemon or Launcher service per Launcher installation.

Monitoring systems through the Management Console

After WebSphere® Transformation Extender has been installed and configured, and the Java™ Launcher started, the Management Console can be used to control and view the statistics about the Launcher processes (.msl system files) that are running.

You must log in to the Management Console and connect to the Java Launcher.

If **Separate Launcher processes** in the Launcher Administration was not checked, and the connection was successful, you will see the **CompoundSystem**.

The Management Console is connected to the Java Launcher, which is running all Launcher systems (.msl files) in a single Launcher process, or CompoundSystem, or in one or more separate Launcher process.

If **Separate Launcher processes** in the Launcher Administration was checked, and the connection was successful, you will see the unique names given to the individual Launcher processes (.msl files) from the File List window in the Launcher Administration. The Management Console is connected to the Java Launcher, which is running separate system Launcher processes for each Launcher system (.msl file) or group of systems.

In this configuration, you can control and monitor each of the different Launcher processes individually without affecting another.

Expand one of the Launcher system files to see the categories of information that you can view in the Management Console. They are **Status**, **History** and **Configuration**.

The **Summary** tab of the Management Console is an example of the statistical information for one of multiple Launcher processes.

Shell script and batch file (launcher.sh and launcher.bat)

The command line utilities that control the Launcher (**launcher.sh** on UNIX operating systems and **launcher.bat** on Windows operating systems) can be configured specifically for running systems in multiple Launcher processes.

When running systems in separate Launcher processes, you control each process with the **systemname** parameter, which is the unique name of the system file (.msl) as defined in the File List window when **Edit** is selected from the Deployment Directories tab in the Launcher Administration. Select -all for all configured Launcher processes.

- [UNIX shell script](#)
- [Windows batch file](#)

UNIX shell script

The following code shows the UNIX **launcher.sh** command usage:

```
Usage: launcher.sh [-start] [-stop [listenerport [hostname]]]
                  [-pause [listenerport [timeout [hostname]]]]
                  [-resume [listenerport [timeout [hostname]]]]
                  [-status launchername systemname]
                  [<[-summary] [-history] [-statusinfo] [-configuration]
                   [-all] [-xml]> Launchername systemname]
                  [-startsystem Launchername <systemname [-all]>]
                  [-stopsystem Launchername <systemname [-all]>]
                  [-pausesystem Launchername <systemname [-all]>]
                  [-resumesystem Launchername <systemname [-all]>]
```

Windows batch file

The following code shows the Windows **launcher.bat** command usage:

```
Usage: launcher.bat [-status Launchername systemname]
                  [<[-summary] [-history] [-statusinfo] [-configuration] [-all]
                   [-xml]> Launchername systemname]
                  [-startsystem Launchername <systemname [-all]>]
                  [-stopsystem Launchername <systemname [-all]>]
                  [-pausesystem Launchername <systemname [-all]>]
                  [-resumesystem Launchername <systemname [-all]>]
```

On Windows platforms, use the operating system's **NET** command to start, stop, pause and continue the Java™ Event Service through the command line.

On UNIX platforms, use the first four *Usage* commands in the **launcher.sh** shell script, start, stop, pause and resume.

Using the Launcher Monitor

When running the Launcher with the **Separate Launcher processes** checkbox in the General tab of the Launcher Administration enabled for multiple Launcher processes, each Launcher process (.msl file) that is configured will be displayed in the Select System window.

Because the Launcher Monitor can only observe one Launcher process at a time, multiple Launcher Monitor instances are required to monitor multiple processes.

Debug trace and log files

The Launcher debug trace and log files are located in the logs directory under the *install_dir*, which refers to the directory where your product is installed.

See [Trace and Log Files Naming Conventions](#) for the specific naming conventions.

z/OS Launcher

This documentation includes detailed information about how to enable the Launcher execution on z/OS® environments.

- [Overview](#)

With WebSphere® Transformation Extender for z/OS, you can run multiple maps or systems on a z/OS operating system on the IBM® z/Architecture® platforms.

- [Referenced terms](#)

Throughout the documentation, the following references are made:

- [Architecture](#)

- [Utilities and tools](#)

- [Native file system](#)

- [Local z/OS administration interfaces](#)

WebSphere Transformation Extender for z/OS includes a set of local administration interfaces that you will use to run the Launcher on the z/OS operating system running on z/Architecture platforms.

- [Multiple processes in the native administration interfaces](#)

- [Troubleshooting tip](#)

Overview

With WebSphere® Transformation Extender for z/OS®, you can run multiple maps or systems on a z/OS operating system on the IBM® z/Architecture® platforms.

Note: To use the z/OS Launcher, you will need to deploy systems. Although the focus of this documentation is to present the details of the z/OS Launcher, you will still need to reference other documentation for the details pertaining to deploying systems.

There are minimal differences between the z/OS implementation, and the Windows and UNIX implementation.

Among the differences are the methods available to administer the Launcher operations. They are:

- Native administration interfaces (ISPF Panel and Operator's Console)

◦ For native administration interfaces, reference the specific documentation.

- Existing user interfaces based on UNIX operating systems (Launcher Administration and Management Console)

◦ For administration user interfaces based on UNIX operating systems, you can reference processes presented in ["Getting Started"](#).

Note: The **DISPLAY** environment variable must be exported and an X Server must be available to host the user interface displays.

Note: Reference ["Starting the Launcher Monitor"](#) for details on how to start the Window Launcher Monitor:

Another difference between the implementations is the capability of the z/OS implementation to access the native file system.

Referenced terms

Throughout the documentation, the following references are made:

Term	References
WebSphere® Transformation Extender for z/OS®	
Includes Launcher	
Command Server	
Batch	
CICS® Execution Option	
CICS	

See the WebSphere Transformation Extender glossary for additional information about terminology used in this documentation.

Architecture

WebSphere® Transformation Extender for z/OS® runs in and uses both the z/OS UNIX System Services (z/OS UNIX) layer and the Hierarchical File System (HFS). It runs as a single process, multi-threaded server from within the z/OS UNIX layer of z/OS. It relies on TCP/IP sockets as the communications method under the TCP/IP communications protocol.

Utilities and tools

WebSphere® Transformation Extender for z/OS® includes JCL utilities and tools to transfer log, trace, and debug data from the HFS into the native file system. A z/OS UNIX System Services Command Server (z/OS UNIX Command Server) is provided as a tool to verify expected map operation prior to deployment in the Launcher.

Note: The z/OS UNIX Command Server is used for testing map behavior only. It is not a tool to be used for running maps in a production environment.

Note: When you are testing maps using the z/OS UNIX Command Server, if those maps were set up in the IFD with overrides, you need to use those override names directly in the maps.

- [JCL tools](#)

The following tools are available in the DTX.SDTXSAMP PDS to assist in transferring data from the HFS into the native file system.

- [z/OS UNIX System Services Command Server](#)

The z/OS UNIX System Services (z/OS UNIX) Command Server allows you to verify a map's behavior prior to deploying it in a system in the Integration Flow Designer.

JCL tools

The following tools are available in the DTX.SDTXSAMP PDS to assist in transferring data from the HFS into the native file system.

Member Name

Description

DTXDHFS

Defines, allocates, and mounts an HFS filesystem.

- Installation process defines a user ID, an HFS mount point, and 'HOME' directory.

DTXH2PDS

Copies the file from the HFS to a PDS or PDS/E member

DTXH2SF

Copies the file from the HFS to a sequential file

By default, all diagnostic information is stored in the HFS in the following locations:

Default Location

File Types in the HFS

\$DTX_HOME_DIR/config/config.yaml

Contains configuration information, including diagnostic definitions

\$DTX_HOME_DIR/dump

Contains diagnostic information when failures are generated by a Launcher execution

\$DTX_HOME_DIR/logs

The logs from each of the user interfaces

\$DTX_TMP_DIR

Contains data generated during execution of a map or system

z/OS UNIX System Services Command Server

The z/OS® UNIX System Services (z/OS UNIX) Command Server allows you to verify a map's behavior prior to deploying it in a system in the Integration Flow Designer.

It is the same as the UNIX Command Server with the additional functionality:

- [Access to native files \(PDS and PDS/E members and Sequential files\)](#).
- Ability to specify output file-creation options for native files using allocation parameters ([Allocation Parameters](#)).
- Ability to specify record terminators using the Record Terminators (/V) command option ([Record Terminators \(/V\)](#)).

Note: For more information and a comparison of the UNIX and Batch/CICS Command Servers, refer to the appropriate chapters in the *Command Server* documentation.

Note: Avoid running the z/OS UNIX Command Server and the Launcher at the same time if both servers are attempting to access the same resources.

Native file system

The native file system is used in processing the Launcher for z/OS®. The Launcher supports the various native file system file types ("[File Types](#)") in various contexts.

The subsequent documentation describes the native file system:

- "[File Types](#)"
- "[File Access](#)"
- "[File Triggering](#)"
- "[Map Names](#)"

- [File types](#)
 - [File access](#)
 - [Map names](#)
 - [Related data file locations](#)
 - [Map sources and targets](#)
 - [File triggering](#)
-

File types

The Launcher for z/OS® supports the following native file system file types:

- Sequential files
- PDS and PDS/E members
- Virtual Storage Method (VSAM) files
- Generation Dataset Group (GDG) Files

Note: For information on these file types, see the specific version of the IBM® z/OS C/C++ Programming Guide for your operating system release.

File access

There are different ways to access files in the native file system through specifications in map input/output.

- IFD specifications
 - ALLOC syntax
 - Map server location
 - Describe location of 'related' data files
 - [Allocation](#)
-

Allocation

Native file system files can be specified in the following locations:

- as input or outputs on maps in your systems.
 - as maps in your map server location setting.
-

Map names

The Launcher for z/OS® supports native file system conventions for specifying the location of compiled maps on the execution server.

- [Compiled map name specifications](#)
-

Compiled map name specifications

When specifying a native file system location for the compiled map, the Launcher for z/OS® supports native file system formats for the following file types:

- [Sequential files](#)
- [PDS and PDS/E members](#)
- [Virtual Storage Method \(VSAM\) files](#)
- [Generation Dataset Group \(GDG\) Files](#)

Note: For formatting information about these file types, see the specific version of the IBM® z/OS C/C++ Programming Guide for your operating system release.

Related data file locations

The location of data files for compiled maps is dependent on the option chosen at WebSphere® Transformation Extender for z/OS® map or system definition time.

The location of related data files (for example, log and trace files) for maps that are stored in the native file system is dependent on the options configured for those map in the Integration Flow Designer prior to deployment. The options are:

- DEFAULT ("DEFAULT")
 - CUSTOM ("CUSTOM")
 - UNIQUE ("UNIQUE")
-
- [DEFAULT](#)
 - [CUSTOM](#)
 - [UNIQUE](#)

DEFAULT

The following information presents the location of data files related to the compiled maps in the native file system when the **DEFAULT** option was specified in the map settings at map or system definition time:

Note: For files in the HFS, the default location is **\$DTX_TMP_DIR**.

Data File in Native File System

Default Location

Audit log
 \$DTX_TMP_DIR/MYUSERNAME.ANY/MAP.log
Input work file
 \$DTX_TMP_DIR/MYUSERNAME.ANY/MAP.Inn
Output work file
 \$DTX_TMP_DIR/MYUSERNAME.ANY/MAP.Onn
Trace file
 \$DTX_TMP_DIR/MYUSERNAME.ANY/MAP.mtr

CUSTOM

The following information presents the location of data files related to the compiled maps in the native file system when the **CUSTOM** option was specified in the map settings at map or system definition time:

Note: For files in the HFS, the location that was specified at map or system definition time is used.

Data File in Native File System

Default Location

Any file
 If the file is in the native file system, the map will fail.

Note: Do not use the **CUSTOM** setting for maps in the native file system.

UNIQUE

The following information presents the location of data files related to the compiled maps in the native file system when the **UNIQUE** option was specified in the map settings at map or system definition time.

Note: For files in the HFS, the default location is **\$DTX_TMP_DIR**.

Data File in Native File System

Default Location

Audit log
 \$DTX_TMP_DIR/MYUSERNAME.ANY/Mer_MAP_process-key_map-counter_hostname.log
Input work file
 \$DTX_TMP_DIR/MYUSERNAME.ANY/Mer_MAP_process-key_map-counter_hostname.Inn
Output work file
 \$DTX_TMP_DIR/MYUSERNAME.ANY/Mer_MAP_process-key_map-counter_hostname.Onn

Map sources and targets

For map source (["Map Sources"](#)) and targets (["Map Targets"](#)), the Launcher for z/OS® supports native file system formats for the following file types:

- [Sequential files](#)
- [PDS and PDS/E members](#)
- [Virtual Storage Method \(VSAM\) files](#)
- [Generation Dataset Group \(GDG\) Files](#)

Note: Many type definitions use [terminators](#) such as <CR><LF>. If data is created locally on the z/OS environment or transferred to the z/OS environment using a file transfer protocol (FTP), the terminators, which would indicate that there is no more data in the record, are not used to terminate the data. When the map processes that data, it would not know when to complete because it would not understand where the end of the data in the record occurs. Therefore, an override must be used to allow the map to complete as if the terminators were contained in the data stream. This override would indicate to the map where the end of each record occurs.

If the target of the FTP is z/OS UNIX System Services (z/OS UNIX), the FTP converts the <CR><LF> terminators (**ASCII** mode) in Windows files to <NL> (**EBCDIC**) on the z/OS environment.

If the target of the FTP is a file in the native file system, the <CR><LF> characters are removed; there is no record terminator at all.

Type trees are designed to indicate the record terminators. Make sure that the appropriate overrides are specified when running the maps on z/OS so that type validation works as expected. The override is used to place an end of record indicator in an otherwise unmarked record.

- [Record Terminators \(/V\)](#)

- [Allocation parameters](#)
- [Map sources](#)
- [Map targets](#)

Record Terminators (/V)

Use the Record Terminators execution command (/V) in source and target map settings to enable special handling of data with embedded record terminators in files with variable-length records, specifically with a record format of variable (V), variable block (VB), or variable block spanned (VBS).

`[/V[terminator_list...]]`

Option

Description

terminator_list

A comma-delimited string of bytes that can be expressed in hex format (for example: /VX5B,X6C,X7C) or character format (for example: /VS,,@). The default record terminator, if none, is indicated on the /V command as a single byte, hex15.

This string of terminator bytes should match the record terminator defined in the type tree that represents this data.

This command functions differently for data sources and targets. If specified for input data, the indicated (or default) record terminator string is appended to each record after it has been read. If specified for an output data set, the record terminator string is interpreted as marking the end of a record in the output.

For inputs, this command causes a string of one or more bytes to be appended to each record to ensure that the data maps properly when using the Launcher for z/OS®. For example, suppose you are mapping an input file on the PC that consists of lines of text terminated by carriage-return/line-feed (0D 0A) pairs. Even though you cannot see them in a text editor, these terminators are embedded in the data, they are defined in the type tree that is created to represent this data, and the map expects them to be there.

Now you need to run this same map using the Launcher for z/OS. However, when you upload the input file to a z/OS environment, the record terminators embedded in the data are removed during the transfer. Consequently, when the file is read in for mapping using the Launcher for z/OS, the record terminators must be reinstated or the data will not map correctly. The solution is to use the command [/VX0D,X0A] to append a hex 0D 0A string to the end of each record.

For outputs, this command specifies that the indicated (or default) byte or bytes are used to mark the ends of each record in the output. When this command is used, the map must embed record terminators in the output data. These terminators indicate the places where record boundaries are to occur in the final output. The record terminators are removed when the records are created in the output target or targets and do not appear in the resulting output.

Allocation parameters

Use allocation parameters to specify output file-creation options when native files are specified as targets and the targets do not exist. It is comparable to specifying output data set definitions on the DD statement in the JCL used if you were executing the Command Server for z/OS® Batch.

Allocation parameters are specified as an ALLOC statement in the **FilePath** setting in the Settings window as part of the file name.

When you use allocation parameters, the **BPXWDYN** service handles file allocation. It is used to define the new output and enables file allocation to be available to programs running outside of a TSO environment. Allocation parameters must be specified in the output card settings for the output file to be created successfully.

Note: See the *z/OS Using REXX and z/OS UNIX System Services* manual for more information about file allocation and the **BPXWDYN** service.

Example

The following example demonstrates how to define the same output data set using two different methods. One method is to use allocation parameters specified in an ALLOC statement and the other method is to use data set definitions specified in the DD statement in the JCL.

Allocation (ALLOC) Parameters

```
ALLOC DD(LABELS) DSN('MYUSERNAME.LABELS') RECFM(F,B) LRECL(80)
UNIT(SYSDA) TRACKS SPACE(5,2) NEW CATALOG DELETE
```

JCL DD Statement

```
//LABELS DD DSN=MYUSERNAME.LABELS,
//          DCB=(RECFM=FB,LRECL=80),
//          UNIT=SYSDA,
//          SPACE=(TRK,(5,2)),
//          DISP=(NEW,CATLG,DELETE)
```

Map sources

This is a description of the specifications you might need to use when native files are specified as sources in the Integration Flow Designer.

Input or inputs

- Record Terminators
 - Place the record terminators in the **FilePath** setting when specifying the map input in the Launcher Settings window.

- A set of brackets ([]) indicates to the z/OS® environment that the data contained inside the brackets are the file characteristics.
- Place the record terminators inside a set of [] in the map input **FilePath**.
- Input(s).#1 FileIn.>.GET.>.Source **FilePath**
Example: // 'MYUSERNAME.IN1' [/VX0D,X0A]

Map targets

This is a description of the specifications you might need to use when native files are specified as targets in the Integration Flow Designer.

Output or outputs

- Allocation Parameters and Record Terminators (if required)
 - Place allocation parameters and record terminators (if required) in the **FilePath** setting in the Settings window as part of the file name when specifying native files as targets.
 - A set of brackets ([]) is used to define allocation parameters and potential record terminators.
 - Place the allocation parameters and record terminators (if required) inside a set of [] in the map output **FilePath**.
 - Output(s).#1 FileOut.>.PUT.>.Target.>.FilePath

Example: [ALLOC DD(OUT1) DSN('MYUSERNAME.OUT1') TRACKS SPACE(5,1) NEW CATALOG RECFM(F,B) LRECL(80) /VX0D,X0A]

Note: Systems (.msl files) will not reside in the native file system; they will reside in the HFS.

File triggering

The Launcher can be set to trigger when files are updated or created on UNIX and Windows systems. Triggering on files in the native z/OS® file system is more complex than in the HFS. This is most evident when trying to do file-update triggering because the z/OS operating system, in general, does not maintain update information for files.

Triggering on native file creation or update supports the following native file system file types, depending on the triggering type:

- [Sequential files](#)
Example: // 'MYUSERID.DATA.INPUT1'
- [PDS and PDS/E members](#)
Example: // 'MYUSERID.DATA(INPUT1)'
- [File-creation triggering](#)
- [File-update triggering](#)
- [Wildcards in native file system names](#)
- [File pre-allocation](#)
- [Incomplete file creation or update](#)

File-creation triggering

File-creation triggering is allowed for any native file system or file system member. In file-creation triggering, when a specified file is created and there is a successful open-file attempt, the event is triggered.

File-update triggering

File-update triggering is more complex because, as mentioned previously, in general, the z/OS® operating system does not maintain information for files that could be used as a way of recognizing that the file changed. The z/OS Launcher will infer a change of a PDS or PDS/E member by recognizing the change in its location, TTR, or pointer.

File-update triggering is allowed for PDS or PDS/E members, but not sequential files. There is no straightforward way to check for sequential file updates.

The following process repeats until the specified event is triggered. The process awaits a specified file to be updated and when it recognizes that it has been updated, by a successful directory read attempt, it triggers execution of an event.

To trigger on an updated file:

1. Retrieve the current TTR information for the PDS or PDS/E member or members by reading the directory.
2. Save the retrieved information.
3. Repeat the loop until the process recognizes that the file has been updated.
 - Get the TTR information.
 - If the TTR information has changed, the event is triggered and the TTR is updated.
 - If the TTR information has not changed, the process repeats the loop to check for file updates.

Wildcards in native file system names

Wildcards are characters that are used to substitute unknown characters. Using wildcards for file creation or update triggers can be costly on z/OS® operating systems.

Note: When using wildcards, place them in a location of native file system names where you are certain that the search will be performed for a well-defined portion of the catalog. The search that is invoked acts like a TSO Data Set List Utility search. It is very powerful and searches through the entire portion of the catalog defined by the location of the wildcard. The larger the portion of the catalog that is searched, the longer the search takes to complete.

Examples of wildcard-triggering uses

The following examples present different ways you can specify and use wildcard-triggering. The bullet points highlight their characteristics, similarities and differences.

```
//'MYUSERID.DATA.INPUT(*)'
```

- The partitioned data set (PDS) being searched is a single data set.
- Any of its members can be a potential trigger.
- The system catalog does not have to be searched.
- There will be repeated reads on the data set to check for new or changed files.

```
 //'MYUSERID.DATA.INPUT*' or  
 //'MYUSERID.DATA.INPUT.*'
```

- The search is on all data sets that have the same specified pattern of nodes or qualifiers.
- Any data set that meets the criteria can be a potential trigger.
- The system catalog is searched.
- There will be repeated searches of the system catalog for new or changed files that match the specified pattern.

Note: In the z/OS operating system, the system catalog is a single resource used by all tasks running on the system. CPU and resource consumption could be very high.

File pre-allocation

Some jobs are designed to create or pre-allocate a file in advance of being populated with data. If the Launcher is set up to trigger on file creation, file pre-allocation might cause the map execution not to complete as expected. Considerations about whether or not to design your jobs with pre-allocated files might be necessary when setting up your Launcher processes.

Incomplete file creation or update

It is possible to trigger on file creation or update if only a portion of the file was created or updated. If the Launcher is set up to trigger on file creation or update, incomplete file creation or update might cause the map execution not to complete as expected.

A method of triggering on file creation or update when it has not completely been created or updated is to create a file with the following disposition parameter setting so that the file will not appear until it is cataloged at the end of the job:

```
DISP=(NEW,CATLG)
```

Local z/OS administration interfaces

WebSphere® Transformation Extender for z/OS® includes a set of local administration interfaces that you will use to run the Launcher on the z/OS operating system running on z/Architecture® platforms.

The two local administration interfaces are:

- [ISPF Panel Interface](#)
- [z/OS Operator Console Interface](#)
- [Overview](#)
- [ISPF panel interface](#)
- [z/OS operator console interface](#)

Overview

A benefit to using the local administration interfaces is that they provide the ability to connect to the user interfaces.

The only limitation of using the local administration interfaces is that you cannot start systems from the Management Console.

- [Activities summary](#)
- [Using the Launcher Management Tools](#)
- [Using the Management Console](#)

After the Management Console is started, you will perform the following activities to establish a connection to the Launcher process running on a z/OS environment.

- [Starting the Launcher Monitor](#)

The Windows Launcher Monitor can be used to monitor the Launcher.

- [Default port settings](#)
- [Environment settings](#)

To run WebSphere Transformation Extender for z/OS, a set of environment variables must be set.

Activities summary

Local administration interfaces perform the following activities:

- Read the system deployment directory.
- Build the command to start the Launcher, including a fully qualified path to the Launcher, list of systems, and command ports.
- Start the Launcher.
- The operator console interface invokes Write-To-Operator (WTO) calls, which communicate status for the operator console interface. WTO messages display the results from the Launcher process for each operation.
- The ISPF interface processes return codes and messages for display in the ISPF panel.

Note: WebSphere® Transformation Extender for z/OS® can be managed through the Management Console and monitored through the Windows Launcher Monitor, even though the administration is performed locally.

After WebSphere Transformation Extender for z/OS is started using one of the local administration interfaces, the Windows/UNIX Management Console can be used to manage the Launcher. The Management Console can be used to view statistical data and also control the compound system that is running from the process running. See [Multiple Processes](#) for additional information when using the multiple-processes feature.

For more details, see the following documentation:

- [Launcher Options](#)
- [Using the Launcher Management Tools](#)
 - [Using the Management Console](#)
 - [Starting the Launcher Monitor](#)

Using the Launcher Management Tools

When using the local administration interfaces to perform Launcher administration tasks, there are Management Console and Launcher Monitor configuration considerations.

Management and monitoring of the Launcher process is dependent on the service having been started. Since the Launcher service is not started when using the local administration interfaces, a different method of accessing the Management Console and Launcher Monitor must be performed.

Using the Management Console

After the Management Console is started, you will perform the following activities to establish a connection to the Launcher process running on a z/OS® environment.

The connection will enable you to use the Management Console to manage your systems running on the z/OS environment.

Note: Because the local administration interfaces are used to start systems, you cannot start systems from the Management Console. The Management Console performs all the other capabilities required to manage the Launcher process running your systems on a z/OS environment.

To configure the Launcher in the Windows and UNIX Management Console

1. Define a new Launcher.
2. In the Management Console, select Tools->Options.
The Options window opens.
3. From the Options list, select Launchers to connect to the remote Launcher.
The override configuration options appear.

Starting the Launcher Monitor

The Windows Launcher Monitor can be used to monitor the Launcher.

The following method can be used to start the Window Launcher Monitor:

- Issue the following command from the Windows command prompt:
 - `dtxmon -c:<computer_name> -p:<monitor_port_number>`
 - *computer_name* is the TCP/IP hostname of the system running the Launcher.
 - *monitor_port_number* is the second port in the 'port range'.
 - Default port is 7001.
- Note: The Launcher Monitor application saves the configuration information you entered in the user interface when you initially started the Launcher Monitor through the command line. It makes the configuration information available for subsequent starts that you can do through the Windows Start menu as follows: Select Start->Programs->Transformation Extender->Launcher->Launcher Monitor.

Default port settings

The local administration interface applications used to run the Launcher on the z/OS® operating system running on z/Architecture® platforms work collectively and must communicate by means of ports. Each application has default port settings. However, you can change them as needed.

The following list displays the default port settings for each application, and the interrelated structure of the applications.

- The Launcher relays monitoring information over port 7001 to the Monitor.
- The Launcher relays status information over port 7000 to the Management Console.
- The local administration interface sends control information (start, stop, pause, resume) to the Launcher process that is running over port 7002.

Environment settings

To run WebSphere® Transformation Extender for z/OS®, a set of environment variables must be set.

They are usually established by running a shell script from the UNIX command line. Since there is no UNIX command line being used in this case, an environment-variable file is used to configure the settings for the local administration interfaces.

File location

The sample environment variable file location resides in DTX.SDTXCNFG(DTXENVAR). It contains pre-specified default values that are included in the WebSphere Transformation Extender for z/OS installation.

Note: The location of the PDS and member name containing the environment variable file must be specified in the DTXENV DD statement of the JCL for starting the z/OS Operator Console command. A sample PROC is included in the installation-defined data set, DTX.SDTXSAMP(DTXPROC).

The location of the DTXENVAR must also be specified in a CLIST or REXX EXEC to start the ISPF interface. A sample REXX EXEC is included in the installation-defined data set, DTX.SDTXCLST(DTXCLIST).

DTXENVAR environment variable file

The contents of **DTXENVAR** include settings for the following variables:

Environment Variable

Description

_CEE_RUNOPTS

Language Environment® runtime options for the Launcher process.

This is an optional field that you can override. See the IBM® z/OS Language Environment Programming Reference, SA22-7562 for a description of the valid runtime options.

CENTRAL_MGMT

Configuration value indicating whether to use centralized (global) or local resource management.

Valid values are YES for centralized (global) resource management and NO for local.

Centralized (global) indicates that resource management will allow multiple processes to share common resources properly.

Default value is NO.

CONNECTION_RETRY_COUNT

The maximum number of times the z/OS native interface is allowed to attempt the TCP/IP connection to the Launcher control port before an error condition, if attempts are not successful, is raised.

Default value is 10.

Note: If the **CONNECTION_RETRY_COUNT** value is reached before the z/OS native interface can start or detect the Launcher, the "TX is not active..." message will appear.

DEBUG

Enable debug option. Valid values are YES to enable the debug option and NO to disable it.

Default value is NO.

INI_FILE

config.yaml file

Override the default location and name of the **config.yaml** file.

Uncomment the line by removing the # character in front of **INI_FILE** and replace <configuration directory>/<ini file> with a valid file location of the **.ini** file to be used.

LAUNCHER_NAME

Symbolic name assigned to the current Launcher. It uniquely identifies the Launcher instance and is also used to prefix the Launcher and debug logs when running the multiple-process Launcher.

The name is displayed in the ISPF DTXEVNT window.

Default value is Launcher-Name.

Note: Maximum length is 19 characters and the name should not contain any spaces.

DTX_CONTROL_PORT

TCP/IP port number used for control information when communicating with the Launcher process.

Default value is 7002.

DTX_HOME_DIR

Installation location for the Launcher)

Default value is /u/dtx/Launcher.

DTX_MONITOR_PORT

TCP/IP port number used for monitoring information when communicating with the Launcher process.
Default value is 7001.

DTX_STATUS_PORT

TCP/IP port number used for status information when communicating with the Launcher process.
Default value is 7000.

SYSTEM_FILE

Location of a single system (.MSL) to be used in a multiple-process Launcher configuration.

Uncomment the line by removing the # character in front of **SYSTEM_FILE**. Replace <deployment directory>/<system name>.msl with the valid file location of the system to be executed.

TIME_ZONE

Time zone used.

This is required only if the locale defined to z/OS UNIX System Services (z/OS UNIX) does not contain a valid **LC_TOD** category.

Default value is EST5EDT.

VIPA_HOSTNAME

The host name for the Virtual IP Address (VIPA) where the Launcher is running. This environment variable is required only when the ports used by the Launcher are defined as VIPA ports.

ISPF panel interface

The ISPF panel interface allows you to manage the Launcher as well as view Launcher information on the z/OS® operating system.

- [**Commands**](#)
The ISPF commands are available to control the execution of the ISPF administration interface.
- [**Menu options**](#)
To manage the Launcher through the ISPF panel interface, select the function by entering the associated menu option number at the **Option ==>** prompt in the **Launcher Administration Main Menu** panel as follows:
- [**Server information**](#)
The server information is contained in the **Launcher Information** area of the panel.
- [**Managing multiple Launchers**](#)

Commands

The ISPF commands are available to control the execution of the ISPF administration interface.

The following procedure uses the ISPF Command Shell screen to initiate the execution of the interface.

To start the ISPF administration interface

1. From ISPF, select Primary Option Menu > option 6 - Command.
 2. At the command prompt in the ISPF Command Shell screen, enter the EXEC 'DTX.SDTXCLST(DTXCLIST)'.
- The ISPF administration interface starts. The Administration Main Menu appears, listing the command options available to control the execution.

See the menu options ([Menu options](#)) for the available ISPF administration commands presented in the Launcher Administration Main Menu.

Menu options

To manage the Launcher through the ISPF panel interface, select the function by entering the associated menu option number at the **Option ==>** prompt in the **Launcher Administration Main Menu** panel as follows:

Menu Function Use	Menu Option	Option Description
Start the Launcher	1	Start the Launcher identified in the Launcher Information area of the panel. Status field must be Unknown.
Stop the Launcher	2	Stop the Launcher identified in the Launcher Information area of the panel. Status field must be Running or Paused.
Pause the Launcher	3	Pause the Launcher identified in the Launcher Information area of the panel. Status field must be Running.
Resume the Launcher	4	Resume the Launcher identified in the Launcher Information area of the panel. Status field must be Paused.
Refresh Launcher Status	5	Get the status of the Launcher identified in the Launcher Information area of the panel. This is allowed for any current state; Status field can contain any valid value.

Server information

The server information is contained in the **Launcher Information** area of the panel.

The following list describes the various fields that are used in the LauncherAdministration Main Menu panel to display information about the Launcher, the status of the Launcher, as well as the ports used.

Field	Description
Server	Name of the current Launcher (user-supplied name)
Status	Current status of the Launcher. Values are: <ul style="list-style-type: none">• Not Responding - not started.• Starting - in the process of starting• Started - has started• Stopping - in the process of stopping• Stopped - has stopped• Pausing - in the process of being paused• Paused - currently paused and not running deployed systems• Unknown - in an unknown state• Error - has detected an error and might or might not continue
Status Port	Port used to communicate the status of the Launcher. Also used by the Management Console interface.
Monitor Port	Port used to communicate monitoring information. Also used by the Windows Launcher Monitor interface.
Control Port	Port used to send commands (start, stop, pause, resume) to the Launcher. Also used by the Management Console interface.

The ISPF panel resides in DTX.SDTXPENU(DTXYEVNT). It must be invoked from a REXX EXEC or CLIST.

Managing multiple Launchers

Multiple Launchers are administered by starting multiple ISPF panels. They must be invoked through multiple CLIST or REXX EXECs and use different configuration files.

Note: You must create a separate environment variable file for each Launcher instance to be able to manage your systems.

z/OS operator console interface

The z/OS® Operator Console interface provides a standard interface that allows the z/OS system operator to control a server by using a cataloged procedure (PROC).

- [Commands](#)
The operator console interface commands are available to control the execution of the Operator Console administration interface.
- [Managing multiple Launchers](#)

Commands

The operator console interface commands are available to control the execution of the Operator Console administration interface.

The following procedure uses the start command to initiate the execution of the interface.

To start the Operator Console interface:

From the Operator Console, enter S DTXPROC.

The Operator Console administration interface has started.

The following table lists the available Operator Console administration commands.

Operator Console Command Use

Commands
Start S DTXPROC
Stop P DTXPROC
Pause F DTXPROC, APPL=PAUSE
Resume F DTXPROC, APPL=RESUME
Status F DTXPROC, APPL=Status

DTXPROC is a PROC that resides in DTX.SDTXSAMP.

Note: The DTXPROC PROC must first be copied to a system procedure library, such as SYS1.PROCLIB, before the procedure can be started from the Operator Console.

Managing multiple Launchers

The z/OS® Operator Console interface manages multiple Launchers in a similar manner as the ISPF panel interface. The only difference is that they are invoked through multiple PROCs, instead of multiple REXX EXECs or CLISTS.

Note: You must create a separate environment variable file for each Launcher instance so that you can manage your systems.

Multiple processes in the native administration interfaces

The WebSphere® Transformation Extender Launcher multiple-processes feature, as described in [Multiple Processes](#), is also available to use with the z/OS® operating system on IBM® z/Architecture® platforms. If you will be running systems specifically through the z/OS native administration interfaces (NAI), the WebSphere Transformation Extender Launcher multiple-processes feature allows you to run multiple Launcher processes (systems or .msl files) within a single installation of the Launcher as described in this documentation.

To enable the multiple-processes feature, global management is selected through resource management settings in the Resource Manager section of the NAI **DTXENVAR** environment variables file. This file is included in the WebSphere Transformation Extender Launcher installation. See [Using Resource Management](#) for background information about global resource management.

The NAI will optionally launch a single system for each Launcher installation instance or recognize a request for global resource management and launch multiple systems in a single instance depending on the way in which you instruct the Launcher, through the settings in the **DTXENVAR** environment variables file, to run the system or systems. Also, there are additional settings in the file that you can change to enable the Launcher to create debug and Launcher log files.

See [DTXENVAR Environment Variable File](#) for the settings that you can configure to enable the WebSphere Transformation Extender Launcher multiple-processes feature.

Troubleshooting tip

If you encounter a problem when running the Launcher for your z/OS® environment, run the maps through the z/OS UNIX System Services (z/OS UNIX) Command Server as a debugging test. The results from running maps through the z/OS UNIX Command Server should be the same as running maps through the Launcher for the z/OS environment. If the source, targets, and settings have changed, you can override them by using the appropriate command line options.

SNMP Agent

Hardware and software network resources that are to be managed and controlled by SNMP-enabled network management software must provide an interface that supports the Simple Network Management Protocol (SNMP). That interface is known as an SNMP Agent. The SNMP Agent allows you to manage the Launcher using SNMP and SNMP-enabled network management software such as HP OpenView, Tivoli® NetView®, and BMC Patrol.

For the purpose of this documentation, the generic term **SNMP manager** is used when referencing SNMP-enabled network management software.

With the SNMP Agent, you can monitor and control the status of the Launcher and associated map events. The following functionality is provided using the SNMP Agent:

- Configure thresholds
- Detect errors
- Monitor Launcher performance and help identify failure situations
- Start, stop, pause, or resume Launcher
- Notify SNMP manager of changes in the state of:
 - Connections
 - Listeners
 - Launcher
- [SNMP overview](#)
- [How an SNMP Agent works](#)
- [How It all fits together](#)
- [Configuration](#)
- [SNMP operations and thresholds](#)
- [DTX MIB](#)

SNMP overview

SNMP is composed of the following components:

Management Information Base (MIB)

The MIB is a database of object definitions. The definition specifies if an SNMP manager can monitor the object.

Simple Network Management Protocol (SNMP)

A network protocol used to manage TCP/IP networks. This protocol provides functions that allow you to access the data object whose definitions are located in the MIB.

- [What is MIB?](#)

What is MIB?

Hardware and software network resources that are controllable using an SNMP manager must provide a definition of the data objects that can be queried or updated. These definitions are located in a database known as the Management Information Base (MIB).

MIB data object definitions that are supported by the SNMP Agent are specified in the **dtx_sm1.mib** file. This file is located in the directory in which the SNMP Agent is installed.

- [MIB data objects](#)
- [MIB variables](#)

MIB data objects

The MIB has an inverted tree structure and each MIB data object is uniquely identified by its location in the MIB tree. Each object is given a unique numeric object identifier (OID) and a descriptive name. A MIB browser allows you to navigate through the MIB tree and to GET and SET MIB variables. All vendor-specific MIB data objects are located under the **iso.org.dod.internet.private.enterprise (1.3.6.1.4.1)** OID. For example, the MIB data object for IBM is located under the **iso.org.dod.internet.private.enterprise.ibm (1.3.6.1.4.1.4769)** OID.

See ["DTX MIB"](#) for the DTX MIB data object definitions.

MIB variables

MIB variables are located at the termination point of each branch in the MIB tree. These variables contain actual configuration, status, and statistical information for the Launcher and the map system it is running. MIB variables can be a single value or string, or even an array or table. Access permission for MIB variables is defined as read-only, read-write, write-only, or none.

How an SNMP Agent works

The SNMP Agent handles the SNMP network communication. When the agent receives a request from an SNMP manager, it checks the OID of the requested data object and if the OID is valid, the agent sends a response back to the SNMP manager. If the OID is not valid, the agent returns an error.

The SNMP Agent is the software component responsible for the Launcher object and responds to queries, carries out requests, and issues traps. A trap is a message sent by an SNMP Agent to the SNMP manager indicating that an event has occurred on the host running the network resource.

How It all fits together

The basic communication process among the key components, SNMP, SNMP manager, SNMP Agent, MIB, and the Launcher, is as follows:

- The SNMP manager issues a request to the SNMP Agent.
- The SNMP Agent SETS and GETS information from the MIB based on the request.
- SNMP Agent sends a reply to the SNMP manager and issues a trap.

When SNMP messages are passed between the SNMP Agent and the SNMP manager, they include the MIB OID, MIB value, what the variable is set to, and the community name of the agent.

Configuration

Before the SNMP Agent can be used in conjunction with third-party SNMP management tools to monitor the Launcher, the following steps must be performed:

To prepare the SNMP Agent to monitor and manage the Launcher status

1. Install and configure the SNMP Agent.
 2. Define user-based security or community permissions for the Launcher's MIB space.
 3. Import the MIB into the SNMP management software. For information about how to import a MIB, consult your vendor-specific documentation.
The Launcher does not require special command lines or other options to enable SNMP management support.
- [Installation](#)
 - [To change the SNMP Agent locale](#)
 - [Configuring the SNMP Agent](#)
 - [SNMP security](#)
 - [Starting the SNMP Agent](#)

Installation

See the release notes for the latest information.

To change the SNMP Agent locale

The SNMP Agent application must be closed before performing this task.

1. Open *install_dir/snmpadmin.bat* (*snmpadmin.sh*) in a text editor.
2. At the **REM set USRLANGUAGE=** line, remove the **REM** comment indicator. At the end of the line, enter a locale value.
Example for Japanese: **set USRLANGUAGE=ja**
3. At the next line, **REM set USRCOUNTRY=**, remove the **REM** comment indicator. At the end of the line, enter the corresponding country indicator if needed.
Example for Traditional Chinese:

```
set USRLANGUAGE=zh
set USRCOUNTRY=TW
```
4. At the line that begins with **REM call %JRE% -DINSTALLDIR="%DTXHOME%"**
-Duser.language=%USRLANGUAGE% ..., remove the **REM** comment indicator from the beginning of the line.
5. At the next line (**call %JRE% -DINSTALLDIR="%DTXHOME%"**
-Dsun.java2d.nodraw ...), add an **REM** comment indicator to the beginning of the line.
For Traditional Chinese, the modified text should look similar to the following:

```
set USRLANGUAGE=zh
set USRCOUNTRY=TW

call %JRE% -DINSTALLDIR="%DTXHOME%" -Duser.language=%USRLANGUAGE% ...
REM call %JRE% -DINSTALLDIR="%DTXHOME%" -Dsun.java2d.nodraw ...
```

6. Save and close the file.

See the *IBM Transformation Extender Introduction* documentation for additional information about locale settings.

Configuring the SNMP Agent

The SNMP Administration utility is used to configure the SNMP Agent. To launch the SNMP Administration utility, click Start > Programs > Transformation Extender > SNMP Agent Administration.

For UNIX platforms, the SNMP Agent Administration utility can be accessed from *install_dir/bin/snmpadmin.sh*. For Windows platforms, the SNMP Agent Administration utility can be accessed from *install_dir\SNMPAdmin.exe*.

- [SNMP administration utility](#)

SNMP administration utility

The SNMP Administration utility is composed of two tabs: **General** and **Startup Settings**.

- [General tab](#)
- [Startup Settings tab](#)

General tab

The following tasks can be performed from the General tab:

- Login and connect to the Launcher
- Configure SNMP Agent IP address and port number information
- Specify interval at which to check Launcher status

Launcher

The following information is used when connecting to the Launcher:

Login Name

This is the name of the Launch user. Launcher user profiles are set up in the Launcher Administration

Password

This is the password for the Launcher user. Launcher user profiles are set up in the Launcher Administration.

Port

This is the port number of the Launcher's listening port. The listening port for the Launcher is set up in the Launcher Administration.

See the Launcher documentation for more information about the Launcher.

Host

This name of the machine on which the Launcher is physically located.

SNMP Agent

The following information is used if multiple instances of the SNMP Agent are installed on the same machine.

On the Windows platform, the agent can be binded to one IP address on multiple ports. On UNIX platforms, the agent can be binded to multiple IP addresses on the same port or on multiple ports using virtual network interfaces.

SNMP IP Address

This is the Internet Protocol (IP) address of the SNMP Agent.

SNMP Port

This is the port number of the SNMP Agent.

SNMP Trap Port

This is the port number to receive trap notifications. The SNMP manager installed on the machine that is connected to that port will receive trap notifications. The default port number is **162**.

A trap community must be defined in the **dtx.acl** file that specifies which IP addresses or domains are to receive trap notifications. The port number that the specified IP address is binded to is specified in the **SNMP Trap Port** field.

System heartbeat

The following information is used to specify an interval for determining the system status of the Launcher:

System Status Polling Period (sec)

This is the interval, in seconds, at which the SNMP Agent will check the Launcher status.

Startup Settings tab

The following tasks can be performed from the Startup Settings tab:

- Specify thresholds that trigger map and connection pending state traps.
- Identify the Launcher and contact information for the human network manager.

Thresholds

The following information is used to set thresholds for map and connection pending states. These thresholds settings are implemented when the SNMP Agent first establishes a connection with the Launcher and are used to trigger a trap.

See "[Thresholds](#)" for more information on thresholds. See "[Traps](#)" for more information on traps.

Init Pending Count

This is the number of maps in initialization pending state that will trigger a trap.

IO Pending Count

This is the number of maps in input/output pending state will trigger a trap.

Connection Pending Count

This is the number of connections in pending state will trigger a trap.

Resource Pending Count

This is the number of maps in resource pending state will trigger a trap.

See the Launcher documentation for more information on map pending states.

MIB II

The following information is used to help you physically locate the machine hosting the Launcher and contact the person responsible:

sysName

This is the contact name for the person that is responsible for the Launcher in case of problems.

sysContact

This is the phone number for the person that is responsible for the Launcher in case of problems

sysLocation

This is the physical location and description of the Launcher.

Authentication Traps

This option enables the SNMP Agent to send a notification to the valid SNMP manager when the community name or context name and access rights of the SNMP manager sending the request does not match the community name and access rights defined in the **dtx.acl** file or the **dtx.uacl** file.

SNMP Agent tracing

The SNMP Agent Trace Level option appends tracing information to the log file that is generated when the SNMP Agent is started. The following options are available:

Option

Description

Disable

No tracing information is added to the log file.

Level 1

Exception error information is added to the log file.

Level 2

Trap and exception error information is added to the log file.

When you run the SNMP Agent on UNIX operating systems with tracing enabled, log information about the Launcher and associated map events is not only output to a file, but also output to the standard output console. To see how to disable the logging of the messages to the output console, see [Disabling the logging of messages to the output console](#).

For more information about the SNMP Agent log file, see ["SNMP Security"](#).

- [Disabling the logging of messages to the output console](#)

This procedure describes the steps to disable the logging of the messages about the Launcher and associated map events to the standard output console.

Disabling the logging of messages to the output console

This procedure describes the steps to disable the logging of the messages about the Launcher and associated map events to the standard output console.

Use this procedure when you are running the SNMP Agent on UNIX operating systems with tracing enabled, and you do not want the SNMP Agent to output the log information to the output console.

To disable the logging of messages to the output console, modify the `logging.properties` file in the following way:

1. Open the `logging.properties` file, which is located under the `java/lib` subdirectory of the path where IBM Transformation Extender is installed.
2. Set the following entry to `OFF` and then save the file:

```
java.util.logging.ConsoleHandler.level  
= OFF
```

When you run the SNMP Agent with the `java.util.logging.ConsoleHandler.level` set to `OFF`, the log information is no longer output to the console screen; it is only output to the log file.

SNMP security

With SNMP version 1 and version 2, SNMP security is controlled through communities that you configure in the `dtx.acl` file. With SNMP version 3, the SNMP user-based security model (USM) is controlled through users and contexts that you configure in the `dtx.uacl` file and the `dtx.security` file.

- [SNMP communities](#)
- [SNMP user-based security model](#)

SNMP version 3 (SNMPv3) supports a user-based security model (USM) that provides message-level security through authentication, privacy, and access control.

SNMP communities

In SNMP version 1 and version 2 (SNMPv1 and SNMPv2), administration groups are known as *communities*. SNMP communities consist of one agent and one or more SNMP managers. You can assign groups of hosts to SNMP communities for limited security checking of agents and management systems or for administrative purposes. Defining communities provides security by allowing only management systems and agents within the same community to communicate.

Communities are identified by community names that you assign. A host can belong to multiple communities at the same time, but an agent does not accept a request from a management system outside its list of acceptable community names. All `GET` and `SET` requests from an SNMP manager to an agent need to specify the community name of which the SNMP manager is a member so that the agent can perform access permission checking.

The `dtx.acl` file, located in the IBM Transformation Extender installation directory, identifies the SNMP communities to the SNMP Agent.

There is no relationship between community names and domain or workgroup names. Community names represent a shared password for groups of network hosts, and should be selected and changed as you would change any password. Deciding which hosts belong to the same community is generally determined by physical proximity.

- [Defining SNMP communities](#)

Defining SNMP communities

The `dtx.acl` file is used to define and maintain community definitions that specify the SNMP managers who can `GET` and `SET` MIB variables. A community definition contains the following:

Component	Description
Community Name	The name used to group SNMP hosts.
Permissions	MIB variable access rights which can be none, read-only, read-write, or write-only.
Host	The domain name or IP address of the physical machine. You can specify an IP address that is in IPv4 (for example, 1.2.3.4) or IPv6 (for example, a:b:c:d:0:1:2:3) format.

The `dtx.acl` file contains three default community definitions: Public, Private, and Trap. Public communities usually enable view-only access to MIB variable information. Typically, all hosts belong to Public, which is the standard name for a common community of all SNMP hosts. Private communities usually grant more permissions such as read-write or write-only. Trap communities identify the SNMP managers that are to receive trap notifications.

1. In a text editor, open the `install_dir\dtx.acl` file.

2. Add the following within the **acl** section and replace variables with actual values:

```
{  
    communities = community_name  
    access = MIB_variable_permissions  
    managers = localhost  
}
```

3. Save the file.

SNMP user-based security model

SNMP version 3 (SNMPv3) supports a user-based security model (USM) that provides message-level security through authentication, privacy, and access control.

- [Defining SNMPv3 users](#)
 - [Adding users to the security file](#)
-

Defining SNMPv3 users

The *install_dir\dtx.uacl* file defines the users, contexts, and security information that control access to the SNMP agent. Do not modify the file name. If you do, the SNMP Agent cannot validate the information when receiving SET and GET requests from the SNMP manager.

context-names

The comma-delimited list of context names. The *context* defines a scope within which an instance of a managed object type can be uniquely identified. For example, a context can be a physical or logical device. An MIB can include multiple contexts.

access

The access-level for retrieving or setting objects in the MIB:

```
read  
    The user can retrieve objects by using the SNMP Get operation  
read-write  
    The user can retrieve or set objects by using the SNMP Get and Set operations
```

security-level

The SNMP security level of requests to the SNMP Agent:

```
noAuthNoPriv  
    No authentication and no privacy  
authNoPriv  
    Authentication and no privacy  
authPriv  
    Authentication and privacy
```

users

The comma-delimited list of SNMPv3 users that are authorized to the SNMP Agent.

For more information, see the *Java™ Dynamic Management Kit* product documentation.

1. In a text editor, open the *install_dir\dtx.uacl* file.

2. Add the following within the **acl** section and replace variables with actual values:

```
{  
    context-names = context_name,context_name...  
    access = MIB_variable_permissions  
    security-level = security_level  
    users = user_name,user_name...  
}
```

3. Save the file.

Adding users to the security file

The *install_dir\dtx.security* file defines the local SNMP engine and the users that are authorized to the SNMP Agent.

localEngineID

A hexadecimal string identifying the local SNMP engine.

localEngineBoots

The number of times the local engine will boot. The SNMP Agent uses this value to evaluate the timeliness of SNMP requests.

userEntry

The SNMPv3 local engine ID, user name, and optional security information. Include a *userEntry* keyword for each user that is authorized to the SNMP Agent. The *userEntry* keyword can include the following variables:

localEngineID
A hexadecimal string identifying the local SNMP engine. This parameter is required.

user_name
An SNMPv3 user that is authorized to the SNMP Agent. SNMPv3 users are defined in the **dtx.uacl** file. This parameter is required.
security_name

A text string used to authenticate communication between the SNMP Agent and the SNMP manager. This parameter is optional.
authentication_algorithm

One of the following algorithms. This parameter is optional.

- usmHMACMD5AuthProtocol
- usmHMACSHAAuthProtocol
- usmNoAuthProtocol

authentication_key

A text password or hexadecimal key that conforms to the requirements of the authentication algorithm. This parameter is optional.

privacy_algorithm

One of the following. This parameter is optional. If omitted, the default value is usmNoPrivProtocol.

- usmDESPrivProtocol
- usmNoPrivProtocol

privacy_key

A text password or hexadecimal key. If you provide a hexadecimal key, it must be a localized key.

For more information about generating the engine ID and the optional userEntry parameters, see the *Java™ Dynamic Management Kit* product documentation.

1. In a text editor, open the **install_dir\dtx.security** file.
2. Add the following and replace variables with actual values:

```
{  
localEngineID = engine_ID  
localEngineBoots = number_of_boots  
userEntry = localEngineID,user_name,security_name,authentication_algorithm,  
authentication_key,privacy_algorithm,privacy_key  
}
```

3. Save the file.

Starting the SNMP Agent

The SNMP Agent is started as a service on Windows or as a daemon on UNIX.

SNMP operations and thresholds

SNMP operations include retrieving information (GET), modifying information (SET) and reporting an event (TRAP). These operations provide management access to MIB variables and agent status:

- **GET:** fetch value of one or more variables
- **SET:** set value of one or more variables
- **TRAP:** unsolicited event notification generic to all or vendor-specific

Any implementation supporting the SNMP operations must adhere to requirements as defined in RFC1157.

- [GET](#)
- [SET](#)
- [Traps](#)

GET

GET is a retrieval operation. The SNMP manager uses the MIB to issue requests across the network to the SNMP Agent running on the remote site. The agent then looks in its list of configured objects to find the requested object. The result is passed back to the SNMP manager from the SNMP Agent.

SET

SET is a modification operation used to set the value of one or more variables. When setting a value, the SNMP manager issues a SET request and forwards it to the SNMP Agent. The SET operation is used to configure and control the managed system.

When a SET request is issued to the SNMP Agent to change the state of a variable, the agent makes the change in the MIB and sends a GET response back to the SNMP manager with the updated state of the variable. Another indicator, error status, informs the SNMP manager if the change was successful.

For example, to start, stop, pause, or resume the Launcher, a SET request is issued by the SNMP Manager, the SNMP Agent modifies the MIB variable and then sends a GET response back to the SNMP Manager with the updated state of the variable. The request to change the state of the Launcher is not executed until this GET response is received.

The error status codes included in response to a SET request are listed below.

Error Code	Description
0	

```

SET was successful
2      SET specified nonexistent variable
3      SET specified invalid value or syntax
4      SNMP manager tried to modify read-only variable

```

Traps

Traps are event-reporting operations issued by the SNMP Agent. A trap is an unsolicited, device-initiated message or notification. The notification containing the OID is always sent from the Launcher.

Using the SNMP adapter, you can create and send custom-defined traps to an SNMP manager. See the SNMP Adapter documentation for more information.

The SNMP manager cannot set traps. It can only receive traps from the SNMP Agent. Traps are stored in the DTX MIB and users define which traps they want by means of the SNMP manager.

Traps can send notifications of warnings and errors. The following table lists the five types of traps that can be issued:

Trap	Description
Map failure	Signals when a map fails
State and Listener change	Signals a change in state of the Launcher or Listeners
Launcher/resource connection failure	Notifies the SNMP manager that a connection between the Launcher and the resource could not be established
Thresholds	Signals when a map or connection pending state threshold has been met
Custom	User-defined trap created and sent using the SNMP adapter

- [Defining SNMP traps](#)
- [Supported traps](#)
- [Thresholds](#)
- [Troubleshooting](#)

Defining SNMP traps

The **dtx.acl** file contains three default community definitions: Public, Private, and Trap. Trap communities identify the SNMP managers that are to receive trap notifications. You can define only one trap community in the **dtx.acl** file.

SNMPv3 does not use the community string to identify destinations. When creating an SNMPv3 trap group, use only the SNMP manager's IP address. For more information, see the *Java™ Dynamic Management Kit product* documentation.

1. In a text editor, open the **install_dir\dtx.acl** file.
2. Add the following within the **trap** section and replace variables with actual values:

```
{
trap-community = trap_community_name
hosts = localhost
}
```

3. Save the file.

Do not modify the **dtx.acl** file name. If you do, the SNMP Agent cannot validate community information when receiving **SET** and **GET** requests from the SNMP manager.

Supported traps

The following table lists all traps, and the variables for which information is included on in the notification message, that are supported by the SNMP Agent. These traps are defined in the **dtx_smi1.mib** file. This file is located in the directory in which the SNMP Agent is installed.

See ["DTX MIB"](#) for descriptions of the MIB object variables.

Trap	#	Trap Description	Variables Returned
dtxInitPendingThreshold	1	Notification of an initialization pending threshold that has been met	dtxMslFileName dtxSystemName dtxInitPendingCount
dtxResourcePendingThreshold	2	Notification of a resource pending threshold that has been met	dtxMslFileName dtxSystemName dtxResourcePendingCount

Trap	#	Trap Description	Variables Returned
dtxIoPendingThreshold	3	Notification of an I/O pending threshold that has been met	dtxMslFileName dtxSystemName dtxIoPendingCount
dtxConnectionPending Threshold	4	Notification of a connection pending threshold that has been met	dtxMslFileName dtxSystemName dtxConnectionPendingCount
dtxSystemStopped	5	Notification that the System has stopped	dtxMslFileName dtxSystemName dtxSystemStatus
dtxSystemStopping	6	Notification that the System is preparing to stop	dtxMslFileName dtxSystemName dtxSystemStatus
dtxSystemPausing	7	Notification that the System is preparing to pause	dtxMslFileName dtxSystemName dtxSystemStatus
dtxSystemPaused	8	Notification the system has entered a pause state	dtxMslFileName dtxSystemName dtxSystemStatus
dtxSystemStarted	9	Notification the system has started	dtxMslFileName dtxSystemName dtxSystemStatus
dtxSystemStarting	10	Notification the system is preparing to start	dtxMslFileName dtxSystemName dtxSystemStatus
dtxSystemNotResponding	11	Notification the system is not responding	dtxMslFileName dtxSystemName dtxSystemStatus
dtxSystemError	12	An unknown error has occurred within the system process	dtxMslFileName dtxSystemName dtxSystemStatus
dtxSystemModified	13	The system definition has been modified	dtxMslFileName dtxSystemName
dtxSystemNew	14	A new system has been added	dtxMslFileName dtxSystemName
dtxSystemDeleted	15	System has been deleted	dtxMslFileName dtxSystemName
dtxSystemResumed	16	Notification that the system has resumed	dtxMslFileName dtxSystemName
dtxEventMapFailed	17	Notification of a map failure	dtxMslFileName dtxSystemName dtxComponentName dtxMapFile dtxMapReturnCode dtxMapErrorMessage dtxFailingCardNumber dtxFailingCardDirection dtxFailingCardReturnCode
dtxEventConnectionFailed	18	Notification of a connection failure	dtxMslFileName dtxSystemName dtxAdapterAliasName dtxAdapterErrorCode dtxConnectionFailureCount
dtxEventListenersDown	19	Listener has lost a connection to the resource	dtxMslFileName dtxSystemName dtxComponentName dtxListenerCardNumber

Trap	#	Trap Description	Variables Returned
dtxEventListenersUp	20	Listener has reestablished the connection to the resource	dtxMslFileName dtxSystemName dtxComponentName dtxListenerCardNumber
dtxUserData	21	User-defined trap created and sent using the SNMP adapter	dtxMapFile dtxUserDataInt dtxUserDataTxt dtxUserDataBin

Thresholds

Thresholds are used to monitor the performance and to help identify situations that require attention. When a threshold limit has been met or exceeded, a notification message is sent to the SNMP manager. Notifications of warnings and errors are used by human managers to actively log and respond to situations when necessary.

The Launcher monitors counts for the following statistical information:

Threshold	Description
Initialization pending count	When the number of maps in init pending state have reached or exceeded the specified threshold
IO pending state	When the number of maps in IO pending state have reached or exceeded the specified threshold
Connect pending	When the number of connections in pending state have reached or exceeded the specified threshold
Resource pending state	When the number of maps in resource pending state have reached or exceeded the specified threshold

- [Setting threshold limits](#)
- [Threshold notification](#)

Setting threshold limits

Users can set limits for each count. The startup threshold values are configured in the SNMP Administration utility. When the SNMP Agent first establishes a connection with the Launcher, the agent sets the threshold limits. Limits can be changed dynamically using the SNMP manager. However, the modification is only a temporary override as the next time the SNMP Agent is restarted, the threshold values originally configured using the SNMP Administration utility will once again be used. To permanently change the values, use the SNMP Administration utility to make the modifications.

See "[SNMP Administration Utility](#)" for more information on setting threshold values.

Threshold notification

Once a threshold meets or reaches a limit, the Launcher sends a notification message to the SNMP Agent, who then issues a trap to the SNMP manager.

While the threshold count is still equal or above the limits, no more messages will be sent for that threshold until the count falls below the limit. A flag is reset at this time, and notifications are sent the next time the limit is reached.

Threshold notification method

The type of notification method used by the SNMP Agent is called single edge. Single edge notification occurs when a value exceeds the threshold limit and the *LowerBound* and *UpperBound* values, the upper or lower limit in a permitted range of values, are the same.

Troubleshooting

Every time the SNMP Agent is started, a log file is created containing information on major events such as Launcher connection status and map failure. This log file can be used as a troubleshooting aide for the Launcher and the map system it is running.

The log file, **SNMPagentlogdate-time.log**, is created in the **install_dir\logs** directory. For example, if the SNMP Agent is started at 4:35 pm on November 16, 2005, the **SNMPagentlog11-16-05-4-35-22-pm.log** file is created.

DTX MIB

This documentation provides information on the variables for the DTX MIB object. The variables in the Listener, Map, and Adapter properties are used in trap notifications. The Table objects contain variables that pertain to a map system component. The associated map system component is located in a System object.

The tables in this documentation also note which variables allow a GET or SET to be performed and which variables are included in trap notifications.

See "[MIB Data Objects](#)" for more information on the structure the DTX MIB.

- [Listener properties](#)
- [Map properties](#)
- [Adapter properties](#)
- [System objects](#)
- [Table objects](#)

Listener properties

The following table lists information on the variables for Listener properties:

Name and OID	Description	GET	SET	TRAP
dtxListenerCardNumber 1.3.6.1.4.1.4769.1.1.3.1	The number of the input card for which a listener is associated	✓		✓

Map properties

The following table lists information on the variables for Map properties:

Name and OID	Description	GET	SET	TRAP
dtxConnectionFailureCount 1.3.6.1.4.1.4769.1.1.3.1.9	Number of times that a connection to a resource has failed	✓		✓
dtxFailingCardReturnCode 1.3.6.1.4.1.4769.1.1.3.1.8	Return code from the failing card	✓		✓
dtxFailingCardDirection 1.3.6.1.4.1.4769.1.1.3.1.7	Whether the failing card is input or output	✓		✓
dtxFailingCardNumber 1.3.6.1.4.1.4769.1.1.3.1.6	The number of the failing card	✓		✓
dtxMapErrorMessage 1.3.6.1.4.1.4769.1.1.3.1.5	Error message associated with the map return code	✓		✓
dtxMapReturnCode 1.3.6.1.4.1.4769.1.1.3.1.4	Return code from the map execution. See the Map Designer documentation for a list of map return codes.	✓		✓
dtxMapFile 1.3.6.1.4.1.4769.1.1.3.1.3	The fully qualified path name of a map file	✓		✓
dtxComponentName 1.3.6.1.4.1.4769.1.1.3.1.2	The name of the map component	✓		✓
dtxWatchID 1.3.6.1.4.1.4769.1.1.3.1.1	Unique identifier assigned by the Launcher	✓		✓
dtxSysTable 1.3.6.1.4.1.4769.1.1.2.1.2	A list of system entries. The number of entries is specified as the value of dtxSysCount .	✓		✓

Adapter properties

The following table lists information on the variables for Adapter properties:

Name and OID	Description	GET	SET	TRAP
dtxAdapterErrorCode 1.3.6.1.4.1.4769.1.1.3.2.2	Return code from the adapter indicating the nature of the failure	✓		✓
dtxAdapterAliasName 1.3.6.1.4.1.4769.1.1.3.2.1	Adapter alias name identifying which resource has failed	✓		✓

System objects

The following table lists information on the variables for System objects:

Name and OID	Description	GET	SET	TRAP
dtxSysCount 1.3.6.1.4.1.4769.1.1.2.1.1	The number of systems in the table, regardless of the current state of the systems	✓		
dtxAdminAllSystemsStatus 1.3.6.1.4.1.4769.1.1.2.1.3	Describes the <i>intended</i> status of the system: 1 = Stop 2 = Start 3 = Pause 4 = Resume	✓	✓	

Table objects

The following table lists information on the variables for Table objects:

Name and OID	Description	GET	SET	TRAP
dtxSysTableEntry 1.3.6.1.4.1.4769.1.1.2.1.2.1	A system entry containing objects for a particular system	✓		✓
dtxMaxWatchCount 1.3.6.1.4.1.4769.1.1.2.1.2. 1.19	The maximum number of instances of any specific map	✓		
dtxMaxThreadCount 1.3.6.1.4.1.4769.1.1.2.1.2. 1.18	The maximum number of threads used to execute maps	✓		
dtxThreadUsage 1.3.6.1.4.1.4769.1.1.2.1.2. 1.17	Number of threads being used by the system	✓		
dtxCPUUsage 1.3.6.1.4.1.4769.1.1.2.1.2. 1.16	Percentage of CPU being used by the system	✓		
dtxMemoryUsage 1.3.6.1.4.1.4769.1.1.2.1.2. 1.15	The number of Mbytes being used by the system	✓		
dtxWatchCount 1.3.6.1.4.1.4769.1.1.2.1.2. 1.14	The number of events being monitored	✓		
dtxFailureTime 1.3.6.1.4.1.4769.1.1.2.1.2. 1.13	Sum of time, in seconds, used by failed map	✓		
dtxSuccessTime 1.3.6.1.4.1.4769.1.1.2.1.2. 1.12	Sum of time, in seconds, used by successful maps	✓		
dtxFailureCount 1.3.6.1.4.1.4769.1.1.2.1.2. 1.11	The number of maps that have failed as long as the Launcher has been running	✓		
dtxConnectionPendingMaps 1.3.6.1.4.1.4769.1.1.2.1.2. 1.10	The number of maps in a 'connection pending' state	✓		✓
dtxIOPendingMaps 1.3.6.1.4.1.4769.1.1.2.1.2. 1.9	The number of maps in an 'IO pending' state	✓		✓
dtxResourcePendingMaps 1.3.6.1.4.1.4769.1.1.2.1.2. 1.8	The number of maps in a 'resource pending' state	✓		✓
dtxInitPendingMaps 1.3.6.1.4.1.4769.1.1.2.1.2. 1.7	The number of maps in an 'initialization pending' state	✓		✓
dtxCurrentMapCount 1.3.6.1.4.1.4769.1.1.2.1.2. 1.6	The number of maps currently being executed	✓		
dtxTotalMaps 1.3.6.1.4.1.4769.1.1.2.1.2. 1.5	The number of maps that have been executed	✓		
dtxUpTime 1.3.6.1.4.1.4769.1.1.2.1.2. 1.4	The amount of time, in seconds, that the system has been running	✓		
dtxMslFileName 1.3.6.1.4.1.4769.1.1.2.1.2. 1.3	The name of the .msl file in which the system is defined	✓		✓

Name and OID	Description	GET	SET	TRAP
dtxConnectionPendingCount 1.3.6.1.4.1.4769.1.1.2.1.2. 1.24	Upper bound limit for connection pending maps	✓	✓	
dtxSystemName 1.3.6.1.4.1.4769.1.1.2.1.2. 1.2	The name of the system defined in the system definition	✓		✓
dtxIoPendingCount 1.3.6.1.4.1.4769.1.1.2.1.2. 1.23	Upper bound limit for I/O pending maps	✓	✓	
dtxSystemIndex 1.3.6.1.4.1.4769.1.1.2.1.2. 1.1	A unique value for each system. The value ranges between 1 and the value of the dtxSysCount .	✓		
dtxResourcePendingCount 1.3.6.1.4.1.4769.1.1.2.1.2. 1.22	Upper bound limit for resource pending maps	✓	✓	
dtxInitPendingCount 1.3.6.1.4.1.4769.1.1.2.1.2. 1.21	Upper bound limit for initialization pending maps	✓	✓	
dtxAdminSystemStatus 1.3.6.1.4.1.4769.1.1.2.1.2. 1.25	Describes the <i>intended</i> status of the Launcher: 1 = Stop 2 = Start 3 = Pause 4 = Resume	✓	✓	
dtxActSystemStatus 1.3.6.1.4.1.4769.1.1.2.1.2. 1.26	Describes the <i>actual</i> status of the Launcher: 1 = Stopped 2 = Started 3 = Paused 4 = Resumed 5 = Stopping 6 = Starting 7 = Pausing 8 = Not Responding 9 = Error	✓		
dtxAdminDebugSystemStat us 1.3.6.1.4.1.4769.1.1.2.1.2. 1.27	Describes the <i>intended</i> trace status: 1 = Off 2 = On	✓	✓	
dtxActDebugSystemStatus 1.3.6.1.4.1.4769.1.1.2.1.2. 1.28	Describes the <i>actual</i> trace status: 1 = Off 2 = On	✓		

Resource Adapters

A comprehensive collection of communication, database, messaging, and utility adapters are available for IBM Transformation Extender, as well as adapters for third-party applications and web services. Use Custom installation to install the adapters.

[Resource Adapters overview](#)

Resource adapters overview

IBM Transformation Extender resource adapters are used to retrieve and route data. They provide access to databases, files, messaging systems, enterprise applications and other data sources and targets. You can use adapters with the Command Server, Launcher, Software Development Kit, or with a map in a map rule.

Each adapter includes a set of adapter commands that can be used to customize its operation. Use the adapter commands to specify different queues and queue managers, specific messages by message ID, specific sets of messages with the same message ID, message descriptors in your data, and more.

Resource adapters answer the questions "Where should the data come from?" and "Where should the data go?"

Source and target data may be in a file, from a specific application, in a database, from a message queue, from an archive file, or from other sources. The client components of the Design Server, Integration Flow Designer, map designer, schema designer, and Database Interface Designer are integral to using the resource adapters.

Each input of data and each output of data requires content definition settings. These card settings are specific to input and output cards, although some settings are common to both cards.

- [Importers](#)
- [System requirements](#)
- [General rules for adapter commands](#)
- [Command syntax summaries](#)
- [Adapter command aliases](#)
- [Configuring Java-based adapters](#)
- [Encoding and decoding data](#)
- [Encode/decode scenarios](#)

- [Transport encode/decode examples](#)
- [Using resource adapters](#)
- [Using an adapter in the map designer](#)
- [Using an adapter in a map or component rule](#)
- [Using an adapter in the Integration Flow Designer](#)
- [On the command line](#)
- [Functions](#)
- [Using wildcards](#)
- [Scope settings and FetchUnit interpretation](#)
- [Using global transaction management](#)
- [File Adapter](#)
- [Text File Importer](#)
- [Sink adapter](#)
- [Application adapter](#)
- [Source and target card settings](#)
- [Communication adapters overview](#)
- [VAN access through the FTP adapter](#)
- [Messaging adapters overview](#)
- [Database adapters overview](#)
- [Return codes and error messages](#)

Importers

The Importer Wizard uses a series of maps to convert metadata into a schema script file (.mts). The Type Tree Maker then processes this schema script and generates a schema that contains all the supported types that are defined in the imported metadata.

Many of the schemas that are generated by the Importer Wizard can be immediately used for map development. However, depending on the contents of the interface-specific metadata file, it might be necessary to modify the generated schema using the schema designer.

For information about importers corresponding to adapters that you have installed, see the specific resource adapter documentation.

System requirements

See the system requirements for operating system and hardware requirements. See the release notes and the adapter documentation for adapter-specific configuration requirements.

It is assumed that a Command Server has already been installed on the computer where the utility adapters are to be installed for run-time purposes.

General rules for adapter commands

Following is a list of the general rules that apply when specifying adapter commands:

- Each command must begin with a hyphen.
- Command characters can be uppercase, lowercase, or mixed-case.
- Command strings are parsed in left-to-right order.
- The order of commands is not important.
- Command files can contain multiple lines. Line breaks can be used in place of the required space that must exist between commands. The line breaks are converted to spaces before processing.
- Space characters:
 - Spaces preceding the first command are ignored.
 - At least one space between commands is required
(for example: **-KEEP -TRACE**).
 - A space is required preceding a variable value
(for example: **-TRACE+S full_pathname**).
- The value of the **Command** setting (adapter-specific commands) in input and output cards and in a **GET** or **PUT** function within a map rule may not exceed 2000 bytes. The expansion of the resource value is included in this limitation.

For platform-specific information about using specific adapter commands with execution commands, refer to the following:

- [For Windows Sources and Targets on the Command Line](#)
- [For UNIX Sources and Targets on the Command Line](#)

Command syntax summaries

A syntax summary is a comprehensive list of the adapter-specific commands organized for use with data sources and data targets. Syntax notation, as applied to all command documentation, indicates required and optional commands, as well as options available for each command.

Required commands appear at the beginning of the list. Optional commands appear in brackets []. Command dependencies are also noted when commands appear on the same line.

The following is a portion of the syntax summary for the IBM® MQSeries® messaging adapter command available for use with data sources:

```
-QMN queue_manager_name  
-QN queue_name
```

This example illustrates that both the Queue Manager Name adapter command (**-QM**N****) and the Queue Name adapter command (**-QN**) are required for data sources. In this example, you may use the Queue Manager Name adapter command (**-QM**N****) to specify the name of the queue manager on which the queue (specified by the **-QN** command) exists.

The following is an additional example of the syntax notation used by the IBM MQSeries messaging adapter:

```
-QMN queue_manager_name  
-QN queue_name  
[-MID message_ID| -HMID hex_message_ID]
```

This example illustrates the use of brackets to indicate optional message ID adapter commands. The brackets [] indicate that both of the message ID commands are optional. The pipe | indicates that you may choose one of these commands, but not both.

In this example, you may use the Message ID adapter command (**-MID**) to specify a particular message ID for a data source. Or you may choose to use the Hex Message ID adapter command (**-HMID**) to specify message identifiers using hexadecimal pair notation.

* The File and Sink adapters do not require any adapter commands for their operation. The File adapter uses only the full pathname and filename for a parameter.

Adapter command aliases

Specify the adapter commands using an execution command string on the command line. You can also create a command file that contains adapter commands dictating the desired execution settings.

Use the appropriate Input Source Override and Output Target Override execution commands with the adapter alias.

Refer to the specific adapter documentation for detailed information for each alias command.

Configuring Java-based adapters

Jar files included with the software and installed in the **install_home** directory are normally added to the CLASSPATH environment variable automatically as in the case of the Command Server and the Launcher. You must modify the CLASSPATH environment variable for all Java-based adapters that rely on external .jar files in order to execute properly.

You can do this directly, or by adding the .jar files to the **config.yaml** file. You must use a full path in the **config.yaml** file. To do so, add entries to the /runtime/External Jar Files dictionary section. Refer to the *Launcher documentation* for more information on the **config.yaml** file.

For example:

```
[External Jar Files]  
jar1=c:\J2EE\lib\j2ee.jar  
jar2=c:\mypath\myjar.jar
```

Note: For Windows, use either a forward slash "/" or two back-slashes "\\\" for the directory delimiter in config.yaml.

- [UNIX only](#)
- [API usage](#)

UNIX only

For all Java™ components the directories containing the Java Virtual Machine (JVM) libraries need to be in the library path environment variable. Typically, this is already defined in the environment, however if it is not, then it must be manually added to the library path environment variable. The following environment variables exist for:

UNIX Platform

Environment Variable

AIX®

LIBPATH

Solaris

LD_LIBRARY_PATH

HP-UX

SHLIB_PATH

Linux®

LD_LIBRARY_PATH

API usage

When a map is started using the Java™ API, where the main process starts with the java command, all necessary .jar files from the **install_home** directory and external .jar files related to the adapter must be added to the CLASSPATH environment variable manually.

Encoding and decoding data

There are a number of adapters which only manipulate data (for example, SOAP), while others provide transport (for example, HTTP). Chaining these adapters together provides greatly enhanced functionality.

Using the **-ENCODE** and **-DECODE** commands in combination with the **-TRANSPORT** command allows you to effectively combine the encoding/decoding capabilities of one adapter with the getting or sending of data of another adapter.

- [Supported adapters](#)
- [Limitations](#)
- [Encoding](#)
- [Decoding](#)
- [Transporting](#)

Supported adapters

Almost any adapter can be used as a transport adapter, although it must be used in its supported contexts. The only adapter that cannot be a transport adapter is the File adapter because it is the only adapter that is not started from the Resource Manager.

The following adapter can be used as an encoding or decoding adapter:

- SOAP

Limitations

The following limitations exist:

- There can be only ONE transport adapter in a card. Since the transport generally represents the end point or start point in a process, it does not make sense to permit more than one transport adapter.
- Encode/decode adapters can be nested. However, the total length of the command line cannot exceed 2000 bytes.

Encoding

Data can be encoded using a specified adapter. The encoding adapter is called before the transport adapter in an output card, **PUT** or **GET** where request data is passed as the third parameter.

The following characteristics exist for the **-ENCODE** command:

- The **-ENCODE** command is used within the command line of a transport adapter, or another encoding adapter.
- The **-ENCODE** command cannot be used in an input card.

For example:

```
-ENCODE 'encoding_adapter (command_line)'
```

The **-ENCODE** command is not required if the **-TRANSPORT** command is used. Encoding is implied by using **-TRANSPORT** in an output card, **PUT** or **GET** where request data is passed as the third parameter.

See "[Encode/Decode Scenarios](#)" for more information.

Decoding

You can also decode the data using a specified adapter. The decoding adapter is called after the transport adapter in an input card or **GET**.

The following characteristics exist for the **-DECODE** command:

- The **-DECODE** command is used within the command line of a transport adapter, or another decoding adapter.
- The **-DECODE** command cannot be used in an output card or a **PUT** function.

For example:

```
-DECODE 'decoding_adapter (command_line)'
```

The **-DECODE** command is not required if the **-TRANSPORT** command is used. Decoding is implied by using **-TRANSPORT** in an input card, or **GET** function.

See "[Encode/Decode Scenarios](#)" for more information.

Transporting

After specifying either an encode or decode command, data can be transported using a specified adapter. The transport adapter is called:

- Before the decoding adapter in an input card or **GET**
- After the encoding adapter in an output card or **PUT**
- Between the encoding and decoding adapters in a **GET** where request data is passed as the third parameter.

The **-TRANSPORT** command is used in the command line of an encoding or decoding adapter. If used from a **GET**, the *transport_adapter* must support request-reply and the adapter that is used with the first parameter of the **GET** must support both encode and decode functions.

For example:

```
-TRANSPORT 'transport_adapter (command_line)'
```

When reversing the chained adapters and using the **-ENCODE** or **-DECODE** commands rather than the **-TRANSPORT** command the **-TRANSPORT** command the behavior is unaltered.

For example, the following **PUT** function calls are functionally equivalent.

```
=PUT(" email ", "-TO fred@hotrods.com -ENCODE ' SOAP (-T)'", Message)
=PUT(" SOAP ", "-T -TRANSPORT ' email (-TO fred@hotrods.com)'",
Message)
```

See [Transport Encode/Decode Examples](#) for more information.

Encode/decode scenarios

The following scenarios apply when using the encode/decode commands in an input card, output card, or in a **GET** function.

- [Input card or GET](#)
- [Output card or PUT](#)
- [GET function](#)

Input card or GET

In an input card or **GET**, where the **-DECODE** command is specified, data is:

- received by the transport adapter,
- decoded by the decoding adapter.

Output card or PUT

In an output card or **PUT** function, where the **-ENCODE** command is specified, data is:

- sent by the transport adapter,
- encoded by the encoding adapter.

GET function

In a **GET** function performing a request/reply using the optional third parameter, the following activities occur:

- Encoding by the encoding adapter
- Sending by the transport adapter
- Receiving by the transport adapter
- Decoding by the decoding adapter

Transport encode/decode examples

The following examples show how the transport adapter encodes or decodes the data when used in an input card, an output card, or in a **GET** function.

- [SOAP Input card](#)
- [SOAP Output card](#)
- [SOAP request/reply in a GET function](#)

SOAP Input card

SOAP
Command
-ENV -TRANSPORT 'HTTP (-URL <http://www.mysite.com>)'
Meaning
The HTTP adapter is called for the transport, then the data is passed to the SOAP adapter to decode.

SOAP Output card

Adapter
SOAP
Command
-ENV -TRANSPORT 'HTTP (-URL <http://www.mysite.com>)'
Meaning
The SOAP adapter is called to encode the data, then the data is passed to the HTTP adapter to transport.

SOAP request/reply in a GET function

Function Call
GET ("SOAP", "-ENV -TRANSPORT 'HTTP (-URL <http://www.mysite.com>, Request)'
Meaning
The SOAP adapter is called to encode the data. The data is then passed to the HTTP adapter to send the request and pick up the response. The SOAP adapter is called again to decode the response.

Using resource adapters

Resource adapters can be specified:

- As a resource in the map designer
 - In map rules and component rules when using the functions **RUN**, **GET**, or **PUT**
 - From the execution settings in the Integration Flow Designer
 - On the command line for a Command Server or Integration Platform Programming Interface
- A wide range of execution commands are available with the Command Server or Software Development Kit. For a complete list of the available options, refer to the Execution Commands documentation.

See the map designer documentation for information on source and target settings in input and output cards.

Using an adapter in the map designer

Use the **Source** setting on an input card or the **Target** setting on an output card to specify an adapter, for example, HTTP, or database, representing any of the database adapters. The adapter-specific commands are entered in the **Command** setting on the card. The adapter-specific commands define how to connect to the data source or target and how to retrieve or route the data.

Using the map designer, select an adapter for a source to retrieve files or data, or a target to transfer files or data.

Maps define how to generate data objects of a certain type. A map contains input and output cards that transform data content from source formats to target formats.

- The **Command** setting on the input card of a map identifies the source of the input data and specifies the use of a resource adapter for input data.
- The **Command** setting on the output card specifies the use of a resource adapter for output data.

On an input card, a **Source** setting of **File** defines the input source as a file and the Source FilePath setting is used to define the **Name** and **Path** of the input file.

If the input data is coming from a database, the **GET Source** Command setting might include commands specifying the database name, user ID, and password.

See the following example to learn how to specify the FTP adapter as a data source in the map designer.

- [Specifying the FTP adapter in the map designer](#)

Specifying the FTP adapter in the map designer

The FTP value for the **Source** setting in this input card identifies the FTP adapter as the source of the input data.

To specify the FTP adapter:

1. In the input card for a map, set **Source** to **FTP**.
2. For the **Command** setting, enter the required FTP URL adapter command (-**URL**), the name of the file to transfer, and other FTP adapter commands as needed.
3. Click OK.

The Source Command value for the FTP input card above is:

```
-TS -URL FTP://sales@host/c:/forms/myfile.txt
```

When this map is run, a file named **myfile.txt** is retrieved from a remote host named **sales**. The Summary Trace adapter command **-TS** specifies that the adapter will create a summary trace file to report adapter activity information during the FTP process.

Using an adapter in a map or component rule

Specify adapter commands using the **GET**, **PUT**, **RUN**, **DBLOOKUP** or **DBQUERY** functions when defining map rules in the map designer or in component rules in the schema designer. The **DBLOOKUP** and **DBQUERY** functions are used for databases only. You must specify the resource adapter with an execution command and a resource adapter alias. For information about adapter aliases, see [Adapter Command Aliases](#).

Use adapter commands in functions and in component rules in the schema designer and map rules in the map designer.

- [Examples of database adapter commands](#)

Examples of database adapter commands

- The following example shows a map rule using a **DBQUERY** function with database adapter commands to specify particular adapter settings.

```
DBQUERY ("SELECT PART_NAME from PARTS where PART_NUMBER =1",
        "-DBTYPE ORACLE -CONNECT MyDatabase -USER janes")
```

This example uses the **-DBTYPE ORACLE** adapter command to specify the database adapter type as Oracle and **-CONNECT MyDatabase** to designate MyDatabase as the Oracle connect string. The **-USER janes** adapter command identifies janes as the user ID.

For information about using the **DBLOOKUP** and **DBQUERY** functions, refer to the Database Interface Designer documentation.

- Use adapter commands with either the Input Source Override - Database execution command (**-ID**) or the Output Source Override - Database execution command (**-OD**) in a **RUN** function to override particular settings. The following example shows using a **RUN** function to specify adapter commands without using an **.mdq** file:

```
RUN ("somemap.mmc" ,
      "-od1 '-dbtype oracle -connect remotedb -user " +
      "userid:profile -password " + password:profile +
      " -table purchordr'")
```

In this example, a map named **somemap.mmc** is being run. The output of this map will be inserted in the PurchOrdr table in an Oracle database defined by the RemoteDB connect string. The logon, user ID, and password settings are retrieved from an input card named Profile.

For information about using the **RUN**, **GET** and **PUT** functions, refer to the Functions and Expressions documentation.

Using an adapter in the Integration Flow Designer

In the Integration Flow Designer, while accessing the Launcher or Command Server execution settings for a map, you can use the adapter as a source to retrieve files or as a target to transfer files.

The **Execution Mode** of the system component determines the execution settings modified to use a resource adapter.

If the **Execution Mode** setting is Launcher, edit the **Launcher Settings** for that system component.

If the **Execution Mode** setting is Command Server, edit the **Command Server Settings**.

For documentation purposes, these settings are referred to as execution settings regardless of the execution mode selected.

You can use the adapter as a source to retrieve files or as a target to transfer files.

For example, to override database-specific adapter settings in an input card or to use the adapter for a source, access the execution settings for a map. From the **GET > Source** list in the execution settings of the map, select Database. In the **GET > Source > Command** setting, specify the adapter commands.

An example of a database-specific adapter command is as follows:

```
-M c:\MDQfiles\Orders.mdq -DN Parts -TE+
```

where **-M** and **-DN** are used to specify usage of a database/query file named **Orders.mdq** and a database named **Parts**. The **-TE+** command causes database trace information containing only errors occurring during map execution to be produced and appended to the existing trace file.

- [Specifying the TIBCO RV adapter in the Integration Flow Designer](#)

Specifying the TIBCO RV adapter in the Integration Flow Designer

The following example illustrates the use of the **TIBCO RV** adapter in the Integration Flow Designer.

To specify the TIBCO RV adapter:

1. Open a system in the Integration Flow Designer.

2. Open the Launcher settings for a map component.
3. Go to the Input(s) #1 in1 > GET-> Source setting.
4. For the **Source** setting, select TIBCO RV from the drop-down list.
5. In the **Command** field, enter a TIBCO RV adapter command. For example, enter:

```
-SBN rvcm.test.in -CONFIRM -T -CMN input_CM -LFN rvcmcfmin.lgr
-LSN 100
```

This example input card retrieves incoming messages with the subject name defined as **rvcm.test.in** using the Subject Name adapter command (-SBN rvcm.test.in). The -CONFIRM adapter command requests that the messaging adapter send an explicit confirmation to the sender that the message was received after successful completion of the map. The Trace adapter command (-T) dictates that a trace file be created to report adapter activity information, recording the events that occur while the adapter is retrieving this data.

The adapter returns messages with the certified session named **input_CM** using the required Certified Session Name adapter command (-CMN input_CM). The ledger file named **rvcmcfmin.lgr** is specified as the ledger file to log the TIBCO Rendezvous PRO certified delivery for this particular certified session using the Ledger File Name adapter command (-LFN rvcmcfmin.lgr). The adapter waits 100 seconds for a message receipt, as specified using the Listen adapter command (-LSN 100).

On the command line

In addition to using a command string on the command line, you can also use adapter commands in command files, batch files, shell scripts, or with the z/OS Platform API.

Use adapter-specific execution commands to:

- Override particular settings for an existing source, such as a user ID and password for database adapters.
- Override an existing source or target, using file, application, message or another entity.

Specify the preferred adapter commands using an execution command string on the command line, or, create a command file that contains adapter commands dictating the execution settings.

For information on adapter-specific input and output execution command overrides, see the Execution Commands documentation.

- [Database-specific adapter commands on the command line](#)
- [For Windows sources and targets on the command line](#)
- [For UNIX sources and targets on the command line](#)

Database-specific adapter commands on the command line

For a database-specific adapter, an existing data source which is a file, can be overridden and specified to be a database using the Input Source Override - Database execution command (-ID). Or, you can override an existing data target that is a table in one database with a table in a different database using the Output Source Override - Database execution command (-OD). For information about all of the options to use within these database execution commands, refer to the Execution Commands documentation.

The values specified using the Input Source Override - Database execution command (-ID) and the Output Source Override - Database execution command (-OD) can be used to designate a query or table to be used as the data source or target, or to override one or more of the adapter commands of an existing database definition as follows:

- using an **.mdq** file
Specify the name of an .mdq file and the name of a particular database that will override the appropriate adapter settings. In this usage, the .mdq file must be available at map execution time. The values in the .mdq file will override all values defined in the map.
- not using an **.mdq** file
Use a command string that includes all of the database adapter commands necessary to specify the desired settings for a map. An **.mdq** file is not used.
- [Command line examples](#)

Command line examples

The following examples show various ways to use the database-specific adapter commands on the command line.

The following examples for data sources and targets are for Oracle database adapters using UNIX-specific syntax. For the Windows platform, adapter commands are enclosed in single quotation marks. Specify adapter commands using an **.mdq** file as follows:

```
dtx parts -ID1 "'-MDQ crib1.mdq -DBNAME Inventory
-QUERY 003A'"
```

This example shows a data source override. The data for input card 1 for the parts map will come from the Inventory database using the query named 003A, all of which are defined in the **crib1.mdq** database/query file.

- Override individual adapter commands as follows:

```
dtx XlatLgcy -ID3 "'-STMT SELECT CustID, Name,
Addr FROM Cust_Table'"
```

This example shows how a SELECT statement may be specified on the command line. Note that the SQL statement consists of all of the input following **-STMT** until the next adapter command or until the single quotation mark ending the adapter command is reached. In this example, the remaining database information from the values compiled into the map file XlatLgcy is used (for example, the **.mdq** file, database, user ID, and other non-overridden values).

- Specify and override the settings in an **.mdq** file:

```
dtx CvtApps -OD6 "'-MDQ /test/new.mdq -DBNAME Purchases -USER  
John -PASSWORD sds34u -TABLE NewerTable -UPDATE'"
```

This example shows that the target for output card 6 in a map named CvtApps. This map is first overridden by the database definition Purchases in **/test/new.mdq** by using the Database/Query File adapter command (**-MDQ**) and the Database Name adapter command (**-DBNAME**). The **new.mdq** database/query file is first read to override the data target that is compiled into the CvtApps map. Subsequently, the User ID adapter command (**-USER**), the Password adapter command (**-PASSWORD**), the Table Name adapter command (**-TABLE**), and the Update adapter command (**-UPDATE**) override those settings obtained from the **.mdq** file.

- Override both a data source and a target adapter command:

```
dtx XferMsgs -ID1 "'-MDQ /share/prod.mdq -DBNAME LgcyProd  
-QUERY xToday'" -OD1 "'-DBTYPE ORACLE -CONNECT ProdDB  
-USER shaz -PASSWORD shaz01 -TABLE ActiveMsgs -UPDATE'"
```

This example shows both an input and an output being overridden. For the data source (**-ID1**), the **.mdq** file (**/share/prod.mdq**), database name (**LgcyProd**), and query (**xToday**) are specified. For the data target (**-OD1**), an Oracle database (**ProdDB**) is designated by the connect string. Subsequent adapter commands specify the user ID (**shaz**), password (**shaz01**), and table (**ActiveMsgs**) to be updated.

For Windows sources and targets on the command line

When using execution commands on Windows platforms, the specified adapter commands must begin with a single quote and end with a single quote. Separate each command with a space within the single quotes.

For example:

```
-IMMQS1 'command1 command2'
```

For UNIX sources and targets on the command line

When using execution commands, while running in a shell environment on UNIX platforms, the specified adapter commands must begin with a double quote, followed by a single quote and end with a single quote, followed by a double quote. Separate each command with a space within the single quotes.

For example:

```
-IAMQS1 "'command1 command2'"
```

Functions

For detailed information about using the **GET**, **PUT** and all other functions, refer to the Functions and Expressions documentation.

- [Using the GET function](#)
- [Using the PUT function](#)
- [Using the RUN function](#)

Using the GET function

The **GET** function returns messages from the adapter as a single text item.

The following example shows a map rule using a **GET** function with messaging adapter commands to specify particular adapter settings:

```
GET ("MQS", "-QM globalque -QN top1")
```

This **GET** example retrieves messages from the MQSeries® server messaging adapter as a single text item. The Queue Manager Name adapter command (**-QM** **globalque**) specifies a queue manager named **globalque**. The Queue Name adapter command (**-QN** **top1**) specifies the **top1** queue.

For more adapter-specific usage information, refer to adapter-specific documentation.

Using the PUT function

The **PUT** function sends messages to the adapter as output and returns a single text item that is always "none".

For all adapters, the scope of the **PUT** function is burst. The executable map controls the transaction behavior, commit or roll back, regardless of nested **RUN** functions.

The following example shows a map rule using a **PUT** function with messaging adapter commands to specify particular adapter settings:

```
PUT ("MQSC", "-QMN myque -QN best", TEXT(messageobject))
```

This PUT example places **TEXT (messageobject)** on the MQSeries® client messaging adapter and returns a single text item of NONE. The Queue Manager Name adapter command (**-QMN myque**) specifies the myque queue manager. The Queue Name adapter command (**-QN best**) specifies the best queue.

For more adapter-specific usage information, refer to each adapter-specific documentation.

Using the RUN function

In a **RUN** function, use adapter commands with the appropriate Input Source Override execution command or the Output Source Override execution command to override particular settings.

The following example shows using a **RUN** function to specify adapter commands without using a database/query file (.mdq):

```
RUN ("somemap.mmc",
    "--od1 '-dbtype oracle -connect remotedb -user " +
    "userid:profile + -password" + password:profile +
    "-table purchordr'")
```

In this example, a map named **somemap.mmc** is being executed. The output of this map is to be inserted in the PurchOrdr table in an Oracle database defined by the **RemoteDB** connect string. The logon, user ID, and password settings are retrieved from an input card named **Profile**.

For more information about using the **RUN** function, refer to the Functions and Expressions documentation.

For more adapter-specific usage information, refer to adapter-specific documentation.

Using wildcards

In most cases, the use of wildcards is supported when using resource adapter commands. Refer to the specific adapter documentation to see if wildcards are supported.

If resource adapter commands with wildcards are used with event triggers on input, events are coordinated to ensure that wildcards are matched. A wildcard value returned from an event trigger is used in other non-input event input cards or output cards, substituting any wildcard value that may exist in any of their properties.

For example:

Card	Adapter	Command
Input card 1 (event):	MQSeries®	MID = *
Input card 2 (not-event):	MQSeries	MID = 123*456
Output card 1:	File	Filename = OUT*.TXT

If a message is received from input card 1 with a message ID of **XYZ**, then:

- input card 2 is called with the MID property set to **123XYZ456**.
- the file created from output card 1 is named **OUTXYZ.TXT**.

Scope settings and FetchUnit interpretation

The following table shows the Scope settings and FetchUnit interpretation available by adapter type. The Scope settings are:

M	Map
B	Burst
C	Card

Table 1. Scope settings and FetchUnit interpretation by adapter type

Source Adapter	Source > Scope Settings	Target > Scope Settings	Aggregates	Divides	FetchUnit Interpretation
Application	M, B, C	M, B, C	No	No	Logical
Archive (Tar, Zip)	M, B, C	M, B, C	Yes	Yes	File in archive
Batch File	M, B, C	M, B, C	No	No	Logical
COM Automation	M, B, C	M, B, C	No	Yes	Logical
Connect:Direct®	M, C	M, C	No	No	Logical
Database	M, C	M, B, C	No	Yes	Row
E-mail	M, B, C	M, B, C	Yes	Yes	Message
File	M	M, C	No	No	Logical
FTP	M, B, C	M, B, C	Yes	Yes	File
GZIP	M, B, C	M, B, C	Yes	Yes	File in archive

Source Adapter	Source > Scope Settings	Target > Scope Settings	Aggregates	Divides	FetchUnit Interpretation
HTTP	M, C	M, C	No	No	Logical
MLLP	M, B, C	M, B, C	Yes	Yes	Message
MQSeries® (client & server)	M, B, C	M, B, C	Yes	Yes	Message
MSMQ	M, B, C	M, B, C	Yes	Yes	Message
Oracle AQ	M, B, C	M, B, C	Yes	Yes	Message
R/3 (ALE, BAPI, BDC)	M, C	M, B, C	No	No	Logical
Shell Script	M, B, C	M, B, C	No	No	Logical
Socket	M, B, C	M, B, C	Yes	Yes	Message
TIBCO Rendezvous	M, B, C	M, B, C	Yes	Yes	Message

Using global transaction management

Certain database and messaging adapters support triggering and global transaction management. Triggering and global transaction management allows monitoring of transactions that can include operations on two or more different data sources. For more information, refer to the Global Transaction Management documentation.

File Adapter

The File adapter is a listener for the Launcher that supports file triggering. The `GET` and `PUT` commands are internal for this adapter.

The only parameter available for the file adapter is the filename specified in the **FilePath** card setting. The settings for this adapter are:

- `GET Source.>.FilePath` for the input card
 - `PUT Target.>.FilePath` for the output card
- The File adapter does not require any adapter commands for its operation. It uses only the full path and filename for a parameter.

The File adapter is internal to the Command Server, Launcher, and z/OS Platform API. However, listener functionality for the Launcher is an external adapter.

Text File Importer

The Text Importer Wizard allows users to select text fragments and create composition of schema groups and items and edit some of the properties. The schemas that are generated by the Text Importer Wizard can be used immediately for map development. However, depending on the contents, it may be necessary to modify the generated schema by using the schema designer. You cannot import a schema while a schema with the same name is open in the schema designer.

- [Supported character sets](#)

Supported character sets

The Text File Importer supports all of the character sets that are supported by the schema editor.

- [Western character sets, international and non-international versions](#)
- [Japanese character sets \(international version\)](#)

Western character sets, international and non-international versions

- ASCII
- EBCDIC
- UTF - 8
- UNICODE Little Endian
- UNICODE Big Endian

Japanese character sets (international version)

- SJIS
- EUC
- UNICODE Big Endian
- UNICODE Little Endian
- UTF - 8

Sink adapter

The Sink adapter is used as a temporary data destination for an output map card which then discards the mapped data.

This capability is useful when a temporary destination is needed to accept output data as part of the map execution without writing the output to a stationary destination.

- [Using the Sink adapter](#)

Using the Sink adapter

The Sink adapter does not require any adapter commands for its operation. In addition, the Sink adapter does not access a persistent source or destination, therefore, a rollback and retry capability is not provided.

For example, use a Sink adapter instead of a file as the destination for temporary output. The advantages of using the Sink adapter include:

- faster map performance due to less file I/O
- reduced file resource conflicts when running multiple instances of the map
- reduced disk space usage

This adapter can be used with the Command Server, Launcher, Software Development Kit, or in a map rule.

For example, using the map designer, select Sink for the **Target** setting in an output card dialog box. The Sink adapter does not require any additional card settings or adapter commands.

You can use the Output Target Override execution command (-OA) with the Sink adapter alias. For example, to override the output card with the Sink adapter so that the data is discarded, the command string would be:

```
-OASINK1 .
```

The Sink adapter command line, for example, -OASINK1 ., must be followed by any dummy argument, for example, a period (.) or an asterisk (*) in order for the adapter to parse the information correctly.

Application adapter

The application adapter and the APP alias is provided only for backwards compatibility for pre-6.0 application adapters.

Source and target card settings

Each input of data and each output of data require content definition settings. These settings are specific to input and output cards, although some settings are common to both cards. For detailed information about source and target settings, see the map designer documentation.

- [Source settings](#)
- [Target settings](#)

Source settings

The Source settings identify the source of the input data. The data may be in a file, from a specific application, in a database, from a message queue, or from other possible sources. Defining the adapter source specifies where to go to get the data and specifics for the data to get.

The significance of the various source settings is specific to the adapter being used to get the input data.

Target settings

The Target settings identify the target for the data produced by the map. The data may be written to a file, inserted into or updated in a table in a database or copied into an e-mail message. The adapter target is the destination of the output data after it is built by the map.

The significance of the various target settings is specific to the adapter being used to send the output data.

Communication adapters overview

Communication adapters provide connectivity to other data sources or targets using industry standards or by using protocols for moving data, for example, the FTP or HTTP adapters.

This group of adapters also provides access to customer applications, either data objects, for example, the Java™ Class adapter, or transactions such as the CICS® adapter. Each adapter includes a set of commands to customize its operation.

VAN access through the FTP adapter

VANs provide services beyond normal transmission, such as protocol conversion or message storing and forwarding, and are often used to transport Electronic Data Interchange (EDI) data.

Many VANs support access through FTPs.

For example, you can access the GXS VAN with the FTP adapter and send and receive secure data using a customized map. This is convenient for the following reasons:

- No special hardware is required
- Internet usability
- Secure transfer of data using FTPs

The *install_dir\examples\adapters\ftp\gxs* directory contains an example of how to gain VAN access through the FTP adapter.

- [Specifying the FTP adapter for GXS VAN access](#)
- [GXS VAN command syntax](#)

Specifying the FTP adapter for GXS VAN access

The following steps explain how to specify the **FTP** adapter while using the GXS VAN and sending example data to an EDI mailbox.

Note that this is only an example and the maps and schemas trees that are provided in the examples folder are example data developed for testing purposes. You must substitute your actual data when using this procedure.

In this example scenario, we will be sending a test file from a map to a test mailbox and then retrieving the file for viewing.

1. Using the map designer, open the **gxscert.mms** file.
2. Verify that you have the **gxs1to1.edi** file located in the *install_dir\examples\adapters\ftp\gxs* directory.
3. Open the **gxs_transmit #1 Output Card**.
4. Edit the **PUT > Target > Command** by changing the **<LUSERID:LPASSWORD>** to the **<LUSERID:LPASSWORD>** needed to access your EDI mailbox.
For example, change: **<LUSERID:LPASSWORD>** to **AAA22406:A3KPP3HB** as the respective User ID and Password.

You must enter a colon (:) between the User ID and Password.

5. Build and run the **gxs_transmit map**.
6. Open the **gxs_retrieve #1 Input Card**.
7. Edit the **GET > Source > Command** by changing the **<LUSERID:LPASSWORD>** to the **<LUSERID:LPASSWORD>** needed to access your EDI mailbox.
For example, change: **<LUSERID:LPASSWORD>** to **AAA22406:A3KPP3HB** as the respective User ID and Password. As previously noted, there must be a colon (:) between the User ID and Password.

We use the same User ID and Password that we used in the transmit (Output card) because the file has been sent to the same location.

8. Build and run the **gxs_retrieve map**.
9. View the results. Your EDI data should be in the **gxs_receive.out** file, which you can view using any text editor.

GXS VAN command syntax

The following command is used in the **gxs_transmit #1 Output card** example for accessing the GXS VAN through the FTP adapter. Although this is only an example, the command structure is the same for when you use the adapter in your work environment.

```
-TV -implicit -URL FTPS://  
AAA22406:A3KPP3HB@sftp.beta.am.gxsics.com:6366/send/  
sendfile;type=A
```

Messaging adapters overview

Messaging-specific adapters are a collection of middleware drivers used to provide a way to specify data sources and targets for maps. Each adapter includes a set of commands used to customize its operation.

The use and operation of a specific messaging adapter is documented separately under its own name.

- [Message flow](#)
- [Adapter transactions](#)
- [Messages as a data source](#)
- [Messages as a data target](#)

Message flow

Maps control the flow of messages between messaging applications in the following ways:

- One-to-one
Messages can flow from a single data source message to a single data target message using a map.
 - One-to-many
Messages can flow from a single data source message to many data target messages using a map.
 - Many-to-one
Messages can flow from many data source messages to a single data target message using a map.
 - Many-to-many
Messages can flow from many data source messages to many data target messages using a map.
- Additionally, message flow can be sequenced between applications and used to integrate non-messaging environments.

Adapter transactions

Map definitions provide settings to control the scope of messaging adapter transactions. It is important to understand how connections to messaging systems are controlled and how this relates to transactional scope.

- [Transactional control](#)
- [Connection management](#)

Transactional control

Use the **Scope** setting for a source or target in a map to determine transaction commitment. The **Scope** setting can be one of the following:

- Map
- Burst
- Card

Connection management

Connection management results in fewer connections being made to the messaging system which may improve performance. Existing connections are reused whenever possible. Two situations exist where connection sharing occurs:

- between cards and maps within a map
- between multiple maps running serially within the Launcher

In the situation in which connection sharing occurs between cards and maps within a map, the connection is described as "active" as long as at least one card or rule is accessing a database and has yet to be committed or rolled back (that is, there is an active transaction). When all cards or rules complete the database access, connection is "inactive", but still open.

Messages as a data source

The two data source processing modes are: integral mode and burst mode.

For each input card in the map, the mode is set to either **Burst** or **Integral** in the map designer Input Card dialog box using the **SourceRule > FetchAs** setting.

Within a map, some input cards may be in burst mode while others are in integral mode, indicating that these cards are executed only once.

- [Example of data processing modes](#)

Example of data processing modes

In addition to the burst mode on a per card basis, the map itself may be set to operate iteratively, thereby invoking multiple source card bursts in a single map invocation. For example, if the Quantity adapter command (**-QTY**) is specified with a setting of 50 on the input card **GET > Source > Command** and the **FetchUnit** value is 5, the map processes the input and output cards until 50 data source objects (messages) are retrieved. The map executes 10 (50÷5) times, with each input card returning five messages concatenated together.

Messages as a data target

When any messaging adapter is specified as the data target, the messaging adapter formulates and sends a single message.

Database adapters overview

This section discusses database-specific adapter information. The use and operation of these adapters is covered in each database-specific adapter's documentation.

Using the Database Interface Designer with the database adapters, data stored in a database under the control of a Relational Database Management System (RDBMS) can be easily identified, defined, and accessed as the source or target for a map.

The Database Interface Designer is used to:

- Specify the databases to use for a source or target.
- Define query statements.
- Automatically generate schemas for queries or tables.

After defining the specifications for accessing each database in the Database Interface Designer, define your map in the map designer where, in map cards, you can specify a query or stored procedure as an input source or a table or a stored procedure as an output target.

The Database Interface Designer is installed as part of the Design Server, along with all of the Windows-based database adapters. These Windows-based adapters provide connectivity to databases residing on your local PC and on other platforms so that you can automatically create schemas for those queries and tables. The adapters installed as part of your Design Server provide a test environment for maps that access the database.

Non-Windows database adapters do not require the Database Interface Designer if you plan to only use **mtsmaker** without a database or query (.mdq) file to generate each schema. For information about using **mtsmaker**, refer to [Generating Type Trees Using mtsmaker](#).

- [Supported database adapters](#)
- [Restrictions and limitations](#)
- [Using database-specific adapters](#)
- [List of commands](#)
- [Generating schemas using mtsmaker](#)

Supported database adapters

- DB2 (z/OS)
- DB2 (z/OS ODBC)
- DB2 (Windows/UNIX)
- Informix
- Microsoft SQL Server
- ODBC
- OLE DB
- Oracle
- Sybase SQL Server

Restrictions and limitations

The Database Interface Designer and database adapters offer options and functions for accessing and manipulating data contained within a database. However, there are some restrictions and limitations for the following functions when using certain adapters:

- **Database triggers:** Using a data source as an input event trigger for the Launcher.
- **Bind facility:** Using bind values in database functions.
- **Stored procedures:** Using stored procedures to access adapter commands and return values from stored functions.

The following table indicates the functions that are not supported by a specific adapter. The Oracle adapters support all functions.

Database-specific Adapter	Triggers	Bind Facility	Stored Procedures
DB2® (z/OS®)	NO	NO	NO
DB2 (z/OS ODBC)	NO		
DB2 (Windows/UNIX)	NO		
Informix®	NO		
Microsoft SQL Server		NO	
ODBC	NO		
OLE DB	NO	NO	
Sybase SQL Server	NO	NO	

Using database-specific adapters

Database-specific adapter commands can be used in any of the following ways:

- When specifying adapter settings for sources and targets.
- With functions in map and component rules.
- On the command line using execution commands to override source or target settings.

See [General Rules for Adapter Commands](#) and [Using Resource Adapters](#) for more information.

- [Adapter usage in functions](#)
- [Adapters in Input Data and Output Data Settings](#)

Adapter usage in functions

The set of adapter commands specified in functions must begin with one double quotation mark and end with one double quotation mark. Each adapter command is separated by one space. For example:

```
DBLOOKUP ("SELECT * FROM PARTS", "command1 command2...")
```

Adapters in Input Data and Output Data Settings

Use the map designer or Integration Flow Designer to specify a database adapter either as a source or as a target by selecting Database as the value for the GET...Source or PUT...Target setting, respectively. In addition to the common adapter settings such as **Scope**, **Retry**, and so forth, you can enter any of the database adapter commands for the **GET > Source > Command** or **PUT > Target > Command** settings.

List of commands

The following table lists each database-specific adapter command and whether the command can be used for a source or target, or in a **DBLOOKUP**, **DBQUERY**, **GET**, or **PUT** function. Usage is indicated with ✓. For a **RUN** function, any adapter command can be used as indicated for a source or target.

Name	Adapter Command Syntax	Source	Target	DBLOOKUP DBQUERY	GET	PUT
Audit (-A or -AUDIT)	-AUDIT[G][+] [full_path]	✓	✓	✓	✓	✓
Bad Data (-BD or -BADDATA)	-BADDATA[+] full_path		✓			✓
Connect String (-C or -CONNECT) (Oracle only)	-CONNECT connect_string	✓	✓	✓	✓	✓
Commit by Card (-CC or -CCARD)*	-CCARD	✓	✓	✓	✓	✓
Commit by Statement (-CS or -CSTMT)	-CSTMT [number]	✓	✓	✓	✓	✓
Delete (-D or -DELETE)	-DELETE		✓			✓
Database Name (-DN or -DBNAME)*	-DBNAME database_name	✓	✓	✓	✓	✓
Database Adapter Type (-DT or -DBTYPE)	-DBTYPE database_adapter	✓	✓	✓	✓	✓
Data Source (-DS or -SOURCE)	<ul style="list-style-type: none">• -SOURCE datasource• -SOURCE server\\databas e	✓ ✓	✓ ✓	✓ ✓	✓ ✓	✓ ✓
File (-F or -FILE)	-FILE [directory]	✓			✓	
Global Transaction Management (-GTX) (Oracle, Oracle AQ, Microsoft SQL Serveronly)	-GTX	✓	✓	✓	✓	✓
Database/Query File (-M or -MDQ)*	-MDQ mdq_file_name	✓	✓	✓	✓	✓
Password (-PW or -PASSWORD)	-PASSWORD password	✓	✓	✓	✓	✓
Stored Procedure Name (-PR or -PROC)	-PROC procedure_name		✓			✓
Query (-Q or -QUERY)	-QUERY query_name	✓			✓	
Row Count (-RC or -ROWCNT) (ODBC, Oracle, DB2)	-ROWCNT row_count	✓		✓	✓	
SQL Statement (-S or -STMT)	-STMT SQL_statement	✓			✓	
Table Name (-TB or -TABLE)*	-TABLE table_name		✓			✓
Trace (-T or -TRACE)	-TRACE[+] full_path	✓	✓	✓	✓	✓
Trace Error (-TE or -TRACEERR)	-TRACEERR[+] [full_path]	✓	✓	✓	✓	✓
Trigger (-TR or -TRIG) (Oracle, Microsoft SQL Server only)	-TRIG trigger_string	✓				
Update (-UP or -UPDATE)	-UPDATE [OFF ONLY]		✓			✓
User ID (-US or -USER)	-USER user_ID	✓	✓	✓	✓	✓
Variable (-V or -VAR)	-VAR name=value...	✓		✓	✓	

*A card setting exists. Using the adapter command will override any value set in the corresponding card setting. For examples, refer to [Command Line Examples](#).

- [Audit \(-A or -AUDIT\)](#)
- [Bad Data \(-BD or -BADDATA\)](#)
- [Connect String \(-C or -CONNECT\)](#)
- [Commit by Card \(-CC or -CCARD\)](#)
- [Commit by Statement \(-CS or -CSTMT\)](#)
- [Delete \(-D or -DELETE\)](#)
- [Database Name \(-DN or -DBNAME\)](#)
- [Database Adapter Type \(-DT or -DBTYPE\)](#)
- [Data Source \(-DS or -SOURCE\)](#)

- [File \(-F or -FILE\)](#)
 - [Global Transaction Management \(-GTX\)](#)
 - [Database/Query File \(-M or -MDQ\)](#)
 - [Password \(-PW or -PASSWORD\)](#)
 - [Stored Procedure Name \(-PR or -PROC\)](#)
 - [Query \(-Q or -QUERY\)](#)
 - [Row Count \(-RC or -ROWCNT\)](#)
 - [SQL statement \(-S or -STMT\)](#)
 - [Table Name \(-TB or -TABLE\)](#)
 - [Trace \(-T or -TRACE\)](#)
 - [Trace Error \(-TE or -TRACEERR\)](#)
 - [Trigger \(-TR or -TRIG\)](#)
 - [Update \(-UP or -UPDATE\)](#)
 - [User ID \(-US or -USER\)](#)
 - [Variable \(-V or -VAR\)](#)
-

Audit (-A or -AUDIT)

Use the Audit adapter command (-AUDIT) to create a file that records the adapter activity for each specified input and output card. This adapter command can be specified for individual input and output cards on a card-by-card basis, or, optionally, as a global audit that encompasses all database activity for the entire map. This command can be used for a source or target, or in a **DBLOOKUP**, **DBQUERY**, **GET**, or **PUT** function. The default usage is to produce a file named **audit.dbl** in the directory in which the map is located. Optionally, you can append the audit information to an existing file or specify a name or full pathname for the file. For more information, refer to the Database Interface Designer documentation.

-AUDIT [G] [+] [*full_pathname*]

Value	Description
G	Activate a global audit for the entire map by specifying this command for the first database card that will have database activity. This records all database activity for all sources, targets, and functions for all defined cards.
+	Append audit information to an existing file.
<i>full_path</i>	Specify the name of the audit file that can include the directory pathname.

Bad Data (-BD or -BADDATA)

Use the Bad Data adapter command (-BADDATA) for a target or in a **PUT** function so that if any inserts, updates, or procedure calls fail to execute, the data that could not be processed is written to a file you specify, and processing continues. The map completes successfully and the following adapter warning code and message are recorded in the execution section of the audit log:

1 One or more records could not be processed
-BADDATA[+] [*full_path*]

Value	Description
+	Append the information to the specified bad data file. This allows multiple cards or multiple iterations of the same map to be able to write to the same file.
<i>full_path</i>	Specify the name for the bad data file or the full pathname to specify the directory. By default, the directory is where the map is located. However, you must specify the file name.

If the **PUT > Target > Transaction > Warnings** setting is set to Fail in an output card, the data that could not be processed is still written to the specified file. However, the map fails.

Connect String (-C or -CONNECT)

Use the Connect String adapter command (-CONNECT) to specify the Oracle host connect string. This command can be used for a source or target, or in a **DBLOOKUP**, **DBQUERY**, **GET**, or **PUT** function. If an SQL*Net host connection string is specified, the connection is established through SQL*Net. If none is specified, a direct connection is established to the database identified by the **ORACLE_SID** environment setting.

-CONNECT *connect_string*

Commit by Card (-CC or -CCARD)

Use the Commit by Card adapter command (-CCARD) to commit or roll back the transaction when the card has been processed. This command can be used for a source or target, or in a **DBLOOKUP**, **DBQUERY**, **GET**, or **PUT** function. When used for a source or within **DBLOOKUP**, **DBQUERY**, or **GET** functions, the behavior is identical to the

Commit by Statement adapter command (-CSTMT) because there is only a single statement or action to be executed.

Note that even if there are multiple **DBLOOKUP**, **DBQUERY**, **GET** and **PUT** map functions in a single output card, each of them will be committed immediately upon execution and not after the card completes, even if they had the -CCARD adapter command specified.

If specified for a database output card, a commit or rollback, based upon the **OnFailure** setting of a card, is made after the completion of the execution of the output card in which multiple **INSERT**, **UPDATE**, or stored procedure statements may have been made.

For a source, this adapter command can be used when the input is a query that executes a stored procedure to perform inserts, updates, or deletes.

-CCARD

Using this command is equivalent to selecting Card as the **PUT > Target > Transaction > Scope** value. If you specify the Commit by Card adapter command (-CCARD), it overrides the value in the **Scope** card setting.

Commit by Statement (-CS or -CSTMT)

Use the Commit by Statement adapter command (-CSTMT) to commit or roll back the transaction after the execution of every statement, or, optionally, after a specified number of statements. A statement is one of the following:

- An **INSERT** or **UPDATE** statement as generated in an output card or **PUT** function.
- The execution of a stored procedure.
- The statement specified in a **DBLOOKUP**, **DBQUERY**, or **GET** function.

In an example for a target, if an output card is a database table for which the Update adapter command (-UPDATE) has also been specified, the Commit by Statement adapter command (-CSTMT) would cause each row to be committed as it is inserted or updated into the database.

For a source, the Commit by Statement adapter command (-CSTMT) can be used when the input is a query that executes a stored procedure performing inserts, updates, or deletes.

-CSTMT [*number*]

Value

Description

number

Specify the number of statements to be executed before committing the data.

The Commit by Statement adapter command (-CS) is an extra call to the database and may degrade performance. However, this command may be beneficial in situations in which the number of statements is large because the database server writes all such updates to its rollback segment, which increases continually with each call.

Commit by Card (-CCARD) and Commit by Statement (-CSTMT) are mutually exclusive. Also, if -CSTMT is specified, it overrides the value in the **PUT > Target > Transaction > Scope** card setting that was compiled into the map.

Delete (-D or -DELETE)

Use the Delete adapter command (-DELETE) to delete all rows in the output table before processing the output data. Use this command for a target or in a **PUT** function.

-DELETE

Database Name (-DN or -DBNAME)

Use the Database Name adapter command (-DBNAME) to specify the name of a database defined in the **.mdq** file that is specified using the Database/Query File adapter command (-MDQ). The Database Name adapter command can be used for a source or target, or in a **DBLOOKUP**, **DBQUERY**, **GET**, or **PUT** function.

-DBNAME *database_name*

The *database_name* you supply is case-sensitive. Also, if -DBNAME is specified, it overrides the value in the **GET > Source > DatabaseQueryFile > Database** card setting.

Database Adapter Type (-DT or -DBTYPE)

Use the Database Adapter Type adapter command (-DBTYPE) to specify the database adapter type you are using. This command can be used for a source or target, or in a **DBLOOKUP**, **DBQUERY**, **GET**, or **PUT** function.

-DBTYPE {ODBC|ORACLE|SQLSVR|SYBASE|DB2|DB2MVS|OLEDB|IFMX}

The adapter type must be specified if the original card is not a database and no database is specified using the Database/Query File adapter command (-MDQ) and the Database Name adapter command (-DBNAME).

Value

Description

ODBC	ODBC database adapter
ORACLE	Oracle database adapter
SQLSVR	Microsoft SQL Server database
SYBASE	Sybase SQL Server database
DB2®	DB2 database on a Windows or UNIX platform
DB2MVS	DB2 database on an z/OS® (formerly MVS™) platform
OLEDB	OLE DB database adapter
IFMX	Informix® database adapter

Data Source (-DS or -SOURCE)

Use the Data Source adapter command (-SOURCE) to specify the name of the ODBC, Microsoft SQL Server, Sybase SQL Server, DB2®, or Informix® data source. This command can be used for a source or target, or in a **DBLOOKUP**, **DBQUERY**, **GET**, or **PUT** function.

-SOURCE datasource|server\\database

Value

Description

datasource	Specify the ODBC, DB2 (Windows/UNIX), or Informix data source.
server\\database	Specify the Microsoft SQL Server or Sybase SQL Server data source.

File (-F or -FILE)

Use the File adapter command (-FILE) for database sources, or in a **GET** function to write the retrieved data to a temporary file that is read by either the Command Server or Launcher. This is useful when the amount of data retrieved by the adapter is large, possibly resulting in high memory consumption. By default, that is, without this command, data is passed between the adapter and the database server in memory. The default specification for this adapter command is for the temporary file to be created in the same directory as the map. The temporary file is then deleted after the map completes.

-FILE [directory]

Value

Description

directory	Specify the directory in which the temporary file should be created.
-----------	--

It is recommended that you specify a value for the StreamMaxMemLimit Resource Manager setting in the **config.yaml (/runtime/stream/MaxMemLimit)** file to limit the memory consumption of the process you are running, instead of using the File database adapter command (-F or -FILE).

If you are currently running processes that are using the File database adapter command (-F or -FILE) and are retrieving data without resulting in high memory consumption, you can continue to use this command.

Do not use the File database adapter command (-F or -FILE) when you are running maps in burst mode. Instead, use the StreamMaxMemLimit Resource Manager setting in the **config.yaml** file to limit memory usage.

Global Transaction Management (-GTX)

Use the Global Transaction adapter command (-GTX) to indicate that the operations on a card, whether input or output, should be processed within a global transaction.

This command is available only for Oracle, Oracle AQ, and Microsoft SQL Server database adapters.

-GTX

For more information about global transaction management, refer to the Global Transaction Management documentation.

Database/Query File (-M or -MDQ)

Use the Database/Query File adapter command (-MDQ) to specify the name of the **.mdq** file that contains the database definition for the database to be used, as specified with the **-DBNAME** adapter command. This Database/Query File adapter command can be used for a source or target, or in a **DBLOOKUP**, **DBQUERY**, **GET**, or **PUT** function.

-MDQ *mdq_file_name*

When this adapter command is used, the **.mdq** file must be available at map execution time. Also, if the **.mdq** file is not in the same directory as the compiled map file, the full pathname is required.

If the Database/Query File adapter command (**-MDQ**) is specified, it overrides the value in the **GET > Source > DatabaseQueryFile > File** setting.

Password (-PW or -PASSWORD)

Use the Password adapter command (**-PASSWORD**) to specify the password that authenticates the user logon. This command can be used for a source or target, or in a **DBLOOKUP**, **DBQUERY**, **GET**, or **PUT** function.

-PASSWORD *password*

If you specify the parameter as `$ (env_var)` where `env_var` represents the name of an environment variable, the actual value of the password is retrieved from this environment variable.

Stored Procedure Name (-PR or -PROC)

Use the Stored Procedure Name adapter command (**-PROC**) to specify the name of a stored procedure to be executed in an output of a map. Use this command either for a target or in a **PUT** function. If the Stored Procedure Name adapter command (**-PROC**) is specified, it overrides the value in the **DatabaseQueryFile > Table** output card setting.

-PROC *procedure_name*

Query (-Q or -QUERY)

Use the Query adapter command setting (**-QUERY**) for a source to specify the name of a query in an **.mdq** file as specified using the Database/Query File adapter command (**-MDQ**) and the database defined by the Database Name adapter command (**-DBNAME**).

-QUERY *query_name*

The name of the query (`query_name`) you supply is case-sensitive. Also, if the Query adapter command (**-QUERY**) is specified, it overrides the value in the **DatabaseQueryFile > Query** input card setting.

Row Count (-RC or -ROWCNT)

For ODBC, Oracle, DB2® for Windows, and DB2 for UNIX adapters only.

Use the Row Count adapter command (**-ROWCNT**) to specify the number of rows to be retrieved in one fetch from the database by the database adapter. This command can be used for a source or in a **DBLOOKUP**, **DBQUERY**, or **GET** function.

This adapter command specifies the number of rows that the adapter should retrieve from the database. It should not be confused with the **FetchAs > FetchUnit** card setting that specifies the number of rows the adapter will pass to the Command Server. For more information about the **FetchUnit** card setting, refer to the Database Interface Designer documentation.

The time to fetch is optimized in accordance with the increase in the number of rows to be retrieved. Therefore, a significant performance improvement may be gained by retrieving as many rows as possible in a single fetch. The number of rows to be retrieved is limited by the memory requirement, because the return buffer must be large enough to hold the rows that are retrieved.

-ROWCNT *row_count*

Without using this command, the default number of rows to be retrieved per fetch is:

- for ODBC, one row
- for DB2, one row
- For Oracle, a 64K buffer is allocated. Then, a calculation is performed to determine the number of rows that can be held in the buffer.

SQL statement (-S or -STMT)

Use the SQL Statement adapter command (**-STMT**) either for a source or in a **GET** function to specify an SQL statement or a call to a stored procedure using either the database-independent or native call syntax to be executed. This execution will provide the input data to a map.

-STMT *SQL_statement*

The "database-independent" syntax for invoking a stored procedure requires the query statement to commence with a "call" command. Following the "call" command, you must specify the input, output and return parameters. Use parentheses to enclose the input and output parameters, separating consecutive parameters with a comma. To indicate a return or an output parameter, use a "?" mark. If you do not need the return value, simply omit it, and if you want to disregard an output parameter, replace it with an 'X'.

Input parameters can be specified as they are or enclosed within single quotation marks, especially when the input contains spaces. If the input data itself includes a single quote, escape it by doubling it. For example:

```
-STMT "call sp_input_param ('I''m escaping a single quote in this input value.')
```

In cases where both input and output parameters are necessary, place the output parameter's "?" mark first, and delimit it from the input parameter using a forward slash character, '/'.

For example, consider the following statement that invokes a stored procedure with an input and output parameter. The first parameter is for input only, while the second one is for both input and output:

```
-STMT "call sp_params (param1,?/param2)"
```

If the above stored procedure includes a return value, it will be ignored unless you precede the stored procedure name with "?=". For example:

```
"call ?=sp_params (...)"
```

Table Name (-TB or -TABLE)

Use the Table Name adapter command (-TABLE) to specify the table to which or in which the map will insert or update data, respectively. Use this command either for a target or in a PUT function. If the Table Name adapter command is specified, it overrides the value in the **DatabaseQueryFile > Table** output card setting.

```
-TABLE table_name
```

Trace (-T or -TRACE)

Use the Trace adapter command (-TRACE) to produce a database trace file. The recorded Trace information details the database adapter activity. This command can be used either for a source or target, or in a DBLOOKUP, DBQUERY, GET, or PUT function.

To produce a trace, you must specify the Trace adapter command (-TRACE) or Trace Error adapter command (-TRACEERR). A database trace is not produced if only map trace is enabled, as was true in previous versions of the database adapters.

The default is for a database trace file to be produced in the directory in which the map is located, using the full name of the map file and a .dbl extension (**map_name.dbl**). Optionally, you can append the trace information to an existing database trace file, specify a file name, or specify the full pathname for the file.

```
-TRACE[+] [full_path]
```

Value**Description**

+

Append trace information to an existing trace file.

full_path

Create a trace file with the specified name in the specified directory. By default, the directory is where the map is located and the file name is **map_name.dbl**.

Trace Error (-TE or -TRACEERR)

Use the Trace Error adapter command (-TRACEERR) to produce a database trace file containing only the database errors that occurred during map execution. This command can be used either for a source or target, or in a DBLOOKUP, DBQUERY, GET, or PUT function.

When this adapter command is specified, a database trace file is produced in the directory in which the map is located, using the full name of the map file and a .dbl extension. Optionally, you can append the trace information to an existing trace file, specify a file name, or specify the full pathname for the database trace file.

The Trace Error adapter command (-TRACEERR) and the Trace adapter command (-TRACE) are mutually exclusive.

```
-TRACEERR[+] [full_path]
```

Value**Description**

+

Append trace information to an existing trace file. Do not put a space between -TRACEERR and the plus [+] sign.

full_path

Create a trace file with the specified name in the specified directory. By default, the directory is where the map is located and the file name is **map_name.dbl**.

Trigger (-TR or -TRIG)

This command is available only for the Oracle and Microsoft SQL Server database adapters.

Use the Trigger adapter command (-TRIG) for a source to specify the trigger string for a trigger specification. For more information about specifying database triggers, refer to the Database Interface Designer documentation.

The Trigger adapter command (-TRIG) can be used as a trigger specification for a query that has no defined trigger in the Database Interface Designer. It can also be used to override conditions that are already defined in a trigger specification associated with a query.

The pipe character (|) is used to represent OR and the ampersand character (&) is used to represent AND. The combination of characters &| represents an AND/OR statement. The value inserted as the `when_clause` must be a valid SQL syntax with `table.column_name` enclosed in square brackets.

```
-TRIG {[R] [I tablename1 [{&|} tablename2]...]
[D tablename1 [{&|} tablename2]...
[U tablename1 [{&|} tablename2]...]
[W when_clause [[table.column_name]]]...}
```

Value

Description
R
This indicates that row-based triggering is being used as opposed to table-based.
I tablename1 [{& } tablename2]...
The insertion of rows into the specified table(s) serves as an input event trigger to the map using this query as a source.
D tablename1 [{& } tablename2]...
The deletion of rows from the specified table(s) serves as an input event trigger to the map using this query as a source.
U tablename1 [{& } tablename2]...
Updating values in the specified column(s) serves as an input event trigger to the map using this query as a source.
W when_clause [[table.column_name]]]
This is an expression that will be evaluated when the other events have occurred. If this expression evaluates to TRUE, the conditions of the trigger specification are met and the map is launched. If the expression is not TRUE, the state is restored as it was prior to any events occurring.
This expression may contain any SQL expressions that are valid for the target database. If database columns are referenced, the column name must be qualified with the table name and enclosed in square brackets.

The following is an example of specifying trigger events and conditions using the adapter:

```
-TR I ACCT_INFO | TRANSACTIONS D ACCT_INFO U ACCT_INFO
W [TRANSACTION.TRANS_TYPE]='PY'
```

This example causes the following to occur:

- Launch the map that uses this query as a data source when one of the following conditions is met:
 - there is insertion into the ACCT_INFO TRANSACTIONS tables
 - there is a deletion from the ACCT_INFO table
 - ACCT_INFO is updated,

but only when a row exists in TRANSACTION with the column TRANS_TYPE set to 'PY'.

You cannot combine both | and & in an event class (*Insert into, Delete from, or Update of*) if more than two tables or columns are being combined. For example, the following command is invalid:

```
-TR I TABLE1 & TABLE2 | TABLE3
```

Update (-UP or -UPDATE)

Use the Update adapter command (-UPDATE) to enable update mode. Update mode causes update operations to be performed for tables or views based upon the defined update keys and key columns to update. For information about defining updating keys and key columns to be updated, refer to the Database Interface Designer documentation.

If enabled, the default is to update each row, if possible, and upon failure, to insert the data. Optionally, you can specify to insert a row or update each row, if possible, and upon failure, to not insert the data. This command can be used either for a target or in a **PUT** function.

If used in a **PUT** function, you must define your update keys and columns to update in an **.mdq** file. Then specify this **.mdq** file in the **PUT** function using the Database/Query File adapter command (-MDQ), together with the Database Name adapter command (-DBNAME), the Table Name adapter command (-TABLE), and the Update adapter command (-UPDATE). The **.mdq** file must be available at runtime.

```
-UPDATE [OFF|ONLY]
```

Value

Description
OFF
Insert a row. Insertions are allowed regardless of success or failure.
ONLY
Update each row, if possible. Upon failure, do not insert the data.

User ID (-US or -USER)

Use the User ID adapter command (-USER) to specify the user ID for connection to the database. This command can be used either for a source or target, or in a **DBLOOKUP**, **DBQUERY**, **GET**, or **PUT** function.

If you specify the parameter as `$ (env_var)` where `env_var` represents the name of an environment variable, the actual value of the password is retrieved from this environment variable.

```
-USER username
```

Variable (-V or -VAR)

Use the Variable adapter command (-VAR) for a source at runtime to pass a value for a variable defined in the SQL SELECT statement. It can also be used in a **DBLOOKUP**, **DBQUERY**, or **GET** function.

```
-VAR name=value
```

Generating schemas using mtmaker

This documentation discusses using the **mtmaker** application to generate schemas.

- [Overview](#)
- [Generating a schema script](#)
- [General parameter rules](#)
- [List of parameters](#)

Overview

The **mtmaker** application helps generate schemas for a query, table, view, or stored procedure when you do not have connectivity between the Database Interface Designer and your non-Windows database.

Use **mtmaker** to produce schema script files (.mts) on your non-Windows database host system that you will subsequently transfer to your development PC in binary mode.

On your development PC, run the Type Tree Maker using the transferred schema script file as input. The Type Tree Maker generates the schema for you.

Generating a schema script

When using **mtmaker**, schema scripts can be created in either of two modes:

- by using a database/query file (.mdq)
- by not using an .mdq file

Control how you want the tree script file to be generated by specifying **mtmaker** parameters.

- [Using mtmaker with a database/query file](#)
- [Using mtmaker without a database/query file](#)

Using mtmaker with a database/query file

You can use a database/query file (.mdq) that was created using the Database Interface Designer and transfer it, in binary mode, to the system hosting your database. The **.mdq** file is then used as a source of information about a particular query, database, or stored procedure.

To generate a tree script file (.mts) for a database query, specify the following **mtmaker** parameters. For specific information about each **mtmaker** parameter, refer to the [List of Parameters](#).

```
mtmaker -M mdq_file  
-Q query_name [-B database_name]  
-F mtt_file_name  
-O mts_file_name  
[-X {F|O|T|FOT}]
```

To generate a tree script file (.mts) for a table, view, or stored procedure, specify the following **mtmaker** parameters:

```
mtmaker -M mdq_file  
-B database_name  
-T table_name|-G stored_procedure_name  
-F mtt_file_name  
-O mts_file_name  
[-X {F|O|T|FOT}]
```

- [Using mtmaker with an .mdq File](#)

Using mtmaker with an .mdq File

The following is an example of how to use **mtmaker** with a database/query file to generate a schema script file for a table:

```
mtmaker -m mdqfile.mdq -b mybase -t mytable -f tree.mtt  
-o mtsfile.mts -x f
```

Using mtmaker without a database/query file

Schemas can be generated without using database/query files. However, you must specify all of the necessary information so that **mtmaker** can generate the tree scripts for queries, tables, views, or stored procedures.

To generate a schema script file (.mts) for a query, specify the following **mtmaker** parameters:

```
mtmaker -S "sql_select_statement"  
-Z database_adapter_type  
[-U user_id]  
[-P password]  
[-D {datasource|connect_string|server\\database}]  
-O mts_file_name  
-F mtt_file_name  
-N type_name  
[-X [F][O][T]]
```

To generate a schema script file for a table, view, or stored procedure, specify the following **mtmaker** parameters:

```
mtmaker -T table_name|-G stored_procedure_name  
-Z database_adapter_type  
[-U user_id]  
[-P password]  
[-D {datasource|connect_string|server\\database}]  
-O mts_file_name  
-F mtt_file_name  
[-X [F][O][T]]
```

- [Using mtmaker without an .mdq File](#)

Using mtmaker without an .mdq File

The following is an example of how to generate a schema script for a query without using a database/query file and specifying an Oracle connect string with the Data Source parameter (-D).

The -D parameter can also be used to specify a data source for ODBC, Sybase SQL Server, or DB2®, as appropriate.

```
mtmaker -s "select * from AddressList" -u yourid -p password  
-z oracle -d "oracle connect string" -o addlist.mts  
-f maillist.mtt -n adddata
```

As part of the SQL Statement parameter (-s) format, the SELECT statement must be enclosed in a double quotation mark.

General parameter rules

To properly enter **mtmaker** parameters, you must observe the correct syntax, as well as follow these general rules:

- Each parameter must begin with a hyphen, for example, -X.
- All parameters can be either upper or lower case, but cannot be mixed case.
- A space is required between the parameter and its value. Refer to the following example:

```
-z oracle
```

- The order of parameters is unimportant.
- If you do not specify all required parameters, **mtmaker** prompts for any missing information. The displayed parameter prompts depend upon the mode in which you are operating, whether using a database/query file or not.

List of parameters

The following table lists the **mtmaker** parameters that can be used for generating schemas, the parameter syntax, and if each one can be used (✓) with or without a database or query file. To access the list of **mtmaker** parameters and the syntax for each parameter, enter **mtmaker** parameters -H or ? on the command line.

Parameter Name	Parameter Syntax	With .mdq	Without .mdq
Database Name (-B)	-B database_name	✓	
Data Source (-D)	<ul style="list-style-type: none"> -D datasource -D connect_string -D server\database 		✓ ✓ ✓
ODBC and DB2®			
Oracle			
Sybase SQL Server			
Type Tree File (-F)	-F mtt_file_name	✓	✓
Stored Procedure Name (-G)	-G stored_procedure_name	✓	✓
Database/Query File (-M)	-M mdq_file	✓	
Category Type (-N)	-N type_name		✓
Tree Script File (-O)	-O mts_file_name	✓	✓
Password (-P)	-P password		✓
Query Name (-Q)	-Q query_name	✓	
SQL Statement (-S)	-S "sql_statement"		✓
Table Name (-T)	-T table_name	✓	✓
User ID (-U)	-U user_id		✓
Variable Name (-V)	-V name=value	✓	✓
Format (-X) *	-X [F][O][T]	✓	✓
Database Adapter Type (-Z)	-Z database_adapter_type		✓

*

The **O** option of the Format parameter (-x) can be used only with the Oracle adapter.

- [Database Name \(-B\)](#)
- [Data Source \(-D\)](#)
- [Schema File \(-F\)](#)
- [Stored Procedure Name \(-G\)](#)
- [Database/Query File \(-M\)](#)
- [Category Type \(-N\)](#)
- [Schema Script File \(-O\)](#)
- [Password \(-P\)](#)
- [Query Name \(-Q\)](#)
- [SQL statement \(-S\)](#)
- [Table Name \(-T\)](#)
- [User ID \(-U\)](#)
- [Variable Name \(-V\)](#)
- [Format \(-X\)](#)
- [Database Adapter Type \(-Z\)](#)

Database Name (-B)

Specify the name of the database as defined in the **.mdq** file.

-B database_name

Data Source (-D)

Specify the name of the ODBC, DB2®, Oracle, or Sybase SQL Server data source.

-D {datasource|connect_string|server\\database}

Option

Description

datasource

Specify the ODBC or DB2 data source.

connect_string

Specify the Oracle host connect string.

server\\database

Specify the Sybase SQL Server data source.

Schema File (-F)

Specify the name of the schema (.mtt) file to be generated by the Type Tree Maker. Enter the file name in the Windows file system format. For example, if a full path name is specified, use the backslash directory separator (\).

-F mtt_file_name

Stored Procedure Name (-G)

Specify the name of the stored procedure for which you want to generate the schema script (.mts).

`-G stored_procedure_name`

Database/Query File (-M)

Specify the name of the database/query file created using the Database Interface Designer. This **.mdq** file contains the definition of either the database, specified by the **-B** parameter, or the query, specified by the **-Q** parameter.

`-M mdq_file`

Category Type (-N)

Specify the name of the category type from which the types defining the table or query will stem.

`-N type_name`

Schema Script File (-O)

Specify the name of the tree script file (.mts) to be generated by **mtsmaker** and used by the Type Tree Maker to create the schema.

`-O mts_file_name`

Password (-P)

Specify the password used to connect to the data source. This parameter is required if it is necessary for your database security environment. Otherwise, it is optional.

`-P password`

Query Name (-Q)

Specify the name of the query in the **.mdq** file for which you want to generate the schema script (.mts).

`-Q query_name`

SQL statement (-S)

Specify an SQL statement or a call to a stored procedure using either the database-independent or native call syntax that is passed to the database management system to determine the content and format of the database output. This statement should be the same as the statement you are using to generate input for your map. However, you may exclude SQL WHERE clauses because they do not affect the format of the result. The SQL statement must be enclosed in double quotation marks.

`-S "sql_statement"`

Table Name (-T)

Specify the name of the table, or the view that can be updated, for which you want to generate the schema script file (.mts).

`-T table_name`

User ID (-U)

Specify the user ID to use to connect to the data source. This parameter may be required based upon your database security.

-U user_id

Variable Name (-V)

Specify this parameter whenever needing variable name substitution in an SQL query that is contained in an MDQ file. The equal sign (=) must be included to properly associate the substituted value with the variable name.

-V name=value

whereby:

name is the name of the variable in the SQL query.

value is the value that should be substituted.

For example:

If using the following query from an .mdq file, select * from BigTable where Identifier=#ID# then specify the following option on the mtsmaker command line:

-V ID=2

In this way, the query to be executed would be as follows:

```
select * from BigTable where Identifier=2
```

Note that variable name values are read by default from the .mdq file, unless overridden on the command line.

See the Database Interface Designer documentation for more information regarding the definition of variables in SQL statements.

Format (-X)

Specify a format for the schema to be generated. Do not use the default, which is delimited.

-X [F] [O] [T]

Option

Description

F

Generate a fixed row format schema.

O

(Oracle only) Generate a 1.4.x format schema.

T

Generate a schema with dates represented as text items. If this option is not specified, the default is to represent dates as date items.

Database Adapter Type (-Z)

Specify the database adapter type to connect to.

-Z {ODBC|ORACLE|SQLSVR|OLEDB|SYBASE|DB2|DB2MVS|IFMX}

Option

Description

ODBC

ODBC database adapter

ORACLE

Oracle database adapter

SQLSVR

Microsoft SQL Server database

OLEDB

OLE DB database adapter

SYBASE

Sybase SQL Server database adapter

DB2*

DB2 (Windows/UNIX) database adapter

DB2MVS

DB2 (z/OS®) database adapter

IFMX

Informix® database adapter

Return codes and error messages

Adapter return codes and messages provide information on the adapter operations, adapter command syntax errors, and reasons for unsuccessful transactions. See the associated adapter documentation for specific error message information.

- [General adapter messages](#)
- [Database-specific adapter messages](#)

General adapter messages

The following is a list of all the codes and messages that are common among all resource adapters. Return codes with positive numbers are warning codes that indicate a successful operation. Return codes with negative numbers are error codes that indicate a failed operation.

See [Database-specific Adapter Messages](#) for information on database-specific return codes and error messages.

Return Code	Message
-1	Unknown error code
-2	Exception occurred
-3	Function failed
-4	Invalid property specified
-5	Property type mismatch
-6	Property not set
-7	Invalid index
-8	Null argument
-9	Invalid argument
-10	Failed to allocate memory
-11	No properties defined for the object
-12	Invalid type specified
-13	Connection failed
-14	File open failed
-15	File write failed
-16	File read failed
-17	File position function failed
-18	Invalid object specified
-19	Null object specified
-20	Illegal function call
-21	Unexpected end-of-file
-22	Only one top level element allowed
-23	Not a top level element
-24	3rd party function failed
-25	Attempt was made to resize non-resizable memory
-26	Running map failed
-27	Loading map failed
-28	Item requested does not exist
-29	Failed to load library

```

-30      Invalid seek performed
-31      Buffer too small - data truncated
-600     Internal error
         Message indicates nature of problem.

```

Database-specific adapter messages

After database adapter activity completes, messages display indicating the results. These messages may also be recorded in the appropriate files as specified, which may include audit logs, trace files, and execution summary. These codes and messages can be returned for the following database adapters:

- DB2® (z/OS® ODBC)
- DB2 (z/OS)
- DB2 (Windows/UNIX)
- Informix®
- Microsoft SQL Server
- ODBC
- OLE DB
- Oracle
- SQL/MP
- SQL/MX
- Sybase SQL Server

The following is a listing of all the codes and messages that can be returned as a result of using the database adapters for sources or targets.

You can also troubleshoot database-specific adapters by using the adapter Trace (-T or -TRACE) or the Trace Error (-TE or -TRACEERR) command.

Adapter return codes with positive numbers are warning codes that indicate a successful operation. Adapter return codes with negative numbers are error codes that indicate a failed operation.

Return Code	Message
0	Success
12	Database -b argument is required with arguments mdqfile -m and query -q
1001	One or more records could not be processed
1002	No data found
-1001	Memory allocation error
-1002	Failed to allocate ODBC environment
-1003	Failed to allocate ODBC statement
-1004	Failed to allocate ODBC connection
-1005	Failed to connect to the database
-1006	Failed to prepare the SQL statement
-1007	Failed to obtain definition of column
-1008	Failed to bind parameters to the SQL statement
-1009	Failed to execute the SQL statement
-1010	Failed to open a cursor
-1011	Failed to insert a row into the database
-1012	Failed to read file
-1013	Failed to open file
-1014	The input buffer was of incorrect form
-1015	Failed to write to file
-1016	Database type is unknown
-1017	Failed to parse SQL statement
-1018	Failed to define host variables

-1019 Failed to logoff the database
-1020 Failed to close the cursor
-1021 Failed to bind parameters to the SQL statement
-1022 Failed to fetch a long data value
-1023 Failed to perform a file seek
-1024 Failed to create file
-1025 The specified query could not be found
-1026 The specified database could not be found
-1027 Failed to load DLL
-1028 Failed to get address of function
-1029 Failed to fetch a row of data
-1030 Failed to get data from the database
-1031 Failed to get data from the database
-1032 No rows were updated
-1033 The number of rows affected was undeterminable
-1034 The supplied statement was syntactically invalid
-1035 Failed to create an execution thread
-1036 Failed to get extended error text
-1037 Table is not accessible
-1038 Failed to disconnect from the database
-1039 Failed to free the database connection
-1040 Failed to free the environment
-1041 MDQ file is invalid
-1042 Failed to close file
-1043 No database connection could be found
-1044 No database type was specified with -DBTYPE
-1045 Failed to find column attributes
-1046 Failed to get the number of result columns
-1047 Failed to get large data
-1048 Failed to cleanup the database connection
-1049 Databases are not supported
-1050 Failed to load DLL
-1051 Failed to get address of function
-1052 Failed to commit or rollback the transaction
-1053 Failed to free the statement
-1054 Insufficient data was provided
-1055 Failed to bind parameters to the SQL statement
-1056 Failed to define large data value
-1057 Failed to put large data value
-1058 Failed to commit the transaction

-1059 Failed to rollback the transaction
-1060 The update statement was invalid
-1061 No type could be found to match column name
-1062 No suitable presentation could be found for a column
-1063 No suitable presentation could be found for a column
-1064 No suitable presentation could be found for a column
-1065 Internal error
-1066 No more rows were returned
-1067 No trigger was defined for the input
-1068 Data was unexpectedly terminated
-1069 Required command line options missing
-1070 The database command line was invalid
-1071 Could not find environment variable
-1072 Failed to attach to database
-1073 Failed to convert datatype to native database type
-1074 Failed to get list of SQL servers
-1075 Failed to get list of databases
-1076 Failed to initialize stored procedure
-1077 Failed to set parameter to stored procedure
-1078 Failed to execute stored procedure
-1079 Failed to get column information
-1080 Failed to declare host variables
-1081 Command line token could not be found
-1082 Failed to create an event
-1083 The command line is too long
-1084 The function/procedure has no return value
-1085 A variable length binary column is not permitted
-1086 Error occurred writing to a LOB column
-1087 Error occurred reading from a LOB column
-1088 Connection to Open Server failed
-1089 Poll to Open Server failed
-1090 The map contains outdated DB parameters. Rebuild the map
-1091 An output date was of an incorrect format
-1092 Internal error occurred
-1093 Burst complete
-1094 Type already exists and override was not specified
-1095 No Row or ProcedureCall group can be found
-1096 Failed to format output from an object type
-1097 Failed to create an object
-1098 An object is not permitted in fixed schema

-1099 Failed to convert database parameters from ASCII to EBCDIC
-1100 One or more columns/parameters are of an unsupported datatype
-1101 Failed to configure run-time environment.
-1102 Failed to Allocate context structure, unknown reason or not enough memory
-1103 A trigger could not be defined for a watch event
-1104 Unique table columns are required for row-based watch events
-1105 Procedure is not accessible
-1106 The configured transaction manager is not supported
-1107 System error occurred while waiting for a watch event
-1108 Triggering is not supported for this adapter.
-1109 User terminated database access
-1110 The trace file name is too long, or the running directory is too deep
The database trace file name and/or the directory name in which the failed map is located, may be longer than 128 characters.
Ensure the file and directory name are shorter than 128 characters.
-1302 A trigger event could not be registered
-2002 Insufficient data passed to output card

Apache ActiveMQ Adapter

This document provides an overview of the Apache ActiveMQ message-broker service and the Apache ActiveMQ adapter.

Overview

With the ActiveMQ Adapter, users can access any of the queues of the Active MQ Adapter to send and receive messages. Users can also create a new queue altogether for sending and receiving messages.

Introduction

Apache ActiveMQ picks up messages from a message queue—breaks-up a message in a way that the message does not give away much information of the sender-application to the application that picks up the message. After breaking up, it places the message to the queue to be picked up by any application. The Apache ActiveMQ can then collect the response from the recipient-application, break down the response in a way that reveals the least information about the recipient-application—and, then place the transformed response to the message queue. In doing so, it effectively implements decoupling of the two communicating applications by minimizing the mutual awareness that these two applications should have of one another to be able to exchange messages. Apache ActiveMQ is an open source message broker and an asynchronous messaging system, written in Java with Java Message Service (JMS) client.

The ActiveMQ adapter establishes a connection to the ActiveMQ server from an application, through which users of the application can access any of the queues of the ActiveMQ server to send and receive messages. With this adapter, users can send the following three types of messages to a queue: Text, Byte, and Blob. Messages are received one at a time in the order of their insertion.

- **[Authentication connections](#)**
The ActiveMQ adapter supports username and password-based authentication to establish the connections to the ActiveMQ broker. The username and password credentials need to be specified in the Username and Password properties in the connection definition for the adapter.
 - **[Adapter properties and commands](#)**
This section lists the properties supported by the adapter and Java Message Service(JMS) headers.
 - **[Examples](#)**
-

Authentication connections

The ActiveMQ adapter supports username and password-based authentication to establish the connections to the ActiveMQ broker. The username and password credentials need to be specified in the Username and Password properties in the connection definition for the adapter.

In addition to the authentication of the user based on the username and password credentials, the adapter supports certificate-based authentication with the ActiveMQ broker.

- The adapter can be configured to request the ActiveMQ broker to provide a certificate that the adapter validates using the truststore information specified in the **Truststore Location** and **Truststore Password** properties.
- The adapter can be configured to present its certificate using the keystore information specified in the Keystore Location, Keystore Password, and Key Password properties when the broker is configured to request clients to provide their certificate when connecting to it. The Key Password is applicable when the private key in

the keystore is protected by a password different from the keystore password.

Adapter properties and commands

This section lists the properties supported by the adapter and Java Message Service(JMS) headers.

This section lists the properties supported by the adapter.

URL

Specifies the URL or the broker URL that allows to run a configured broker using a single URL for all the configuration (Example of default broker URL: `tcp://localhost:61616`). This is a mandatory property. It is required at the time of setting the connection.

The corresponding adapter command is **-URL**.

Login Username

Specifies the username to use for authentication when connecting to the message broker. The corresponding adapter command is **-U username** (or **-USERNAME username**).

Login Password

Specifies the password value to use for authentication when connecting to the message broker. The corresponding adapter command is **-P password** (or **-PASSWORD password**).

Truststore File

Specifies the path to the truststore in PKCS#12 or JKS format that the adapter uses to validate the broker certificate when SSL/TLS is enabled. The truststore must contain the chain of certificates up to the CA certificate that is used to sign the broker certificate. The corresponding adapter command is **-TSL path** (or **-TRUSTSTORELOCATION path**).

Truststore Password

Specifies the password for the adapter to use to access the specified truststore when SSL/TLS is enabled. The corresponding adapter command is **-TSP password** (or **-TRUSTSTOREPASSWORD password**).

Truststore Type

Specifies the type of truststore (PKCS#12 or JKS). The corresponding adapter command is **-TST type** (or **-TRUSTSTORETYPE type**).

Keystore File

Specifies the path to the keystore in PKCS#12 or JKS format that the adapter uses to retrieve the private key and certificate with the public key. Which is required, if the broker is configured to request clients to provide a certificate for SSL/TLS connections. The corresponding adapter command is **-KSL path** (or **-KEYSTORELOCATION path**).

Keystore Password

Specifies the password for the adapter to access the specified keystore when SSL/TLS is enabled. The corresponding adapter command is **-KSP password** (or **-KEYSTOREPASSWORD password**).

Key Password

Specifies the password for the adapter to access the private key in the specified keystore when SSL/ TLS is enabled. The password is required, if the private key is secured with a password different from the keystore password. The corresponding adapter command is **-KP password** (or **-KEYPASSWORD password**).

Keystore Type

Specifies the type of Keystore (PKCS#12, JKS, or JVM). The corresponding adapter command is **-KST type** (or **-KEYSTORETYPE type**).

Read Mode

Specifies whether to keep or delete the messages in ActiveMQ queue after a successful read operation. The corresponding adapter command is **-RM keep/delete** (or **-READMODE keep/delete**).

Note: The default is `keep`, so it is not necessary to specify **-READMODE keep**.

Queue Name

Specifies the queue name to and from which messages are to be sent and received. This is mandatory parameter. Queue name property can be passed to adapter input or output cards using **-Q** (or **-QUEUE**) command.

Create Queue

This is a boolean flag. It is used at the time of sending a message to the ActiveMQ. If this flag is set to true and the particular queue is not present, it will create the queue. If this flag is set to false and the queue is not present, user will get invalid queue name exception. If the value is not mentioned for this flag, then by default the value is false. The corresponding adapter command is **-CQ** (or **-CREATEQUEUE**).

Validate Queue

Specifies whether to validate queue before reading or sending message. The corresponding adapter command is **-V** (or **-VALIDATE**).

Message Type

Specifies the type of message being sent to the ActiveMQ adapter. The currently supported message types are: Text, Blob, Byte. Examples of Blob file are a large image file, and a PDF file among others. An 'invalid message type' exception will occur, if a user tries to set a message type other than the supported ones. Though this is an

optional parameter, it is recommended to set a message type at the time of creating an output card. Message type property can be passed to adapter output cards using -**MT** (or **-MSGTYPE**) command.

Delay Time

Specifies the time duration after the lapse of which, the adapter will look out for the next message. A timeout of zero never expires, and the call blocks indefinitely. The timeout value is specified in milliseconds. This property is optional but recommended to set when receiving message from ActiveMQ (used in input card) with a minimum value (1) to avoid getting stuck in case no message is present in the queue. The Delay-time property can be passed to adapter input using -**DT** (or **-DELAYTIME**) command.

Logging

Specifies the level of logging to use for the log (trace) file produced by the adapter. The default is Off. The value Information means log informational, the value Errors Only means log error messages only, and the value Verbose means log debug and trace level messages along with the informational and error messages.

The corresponding adapter command is:

-T [E|V] [+|] [file_path]

-T -> Log adapter informational messages.

-TE -> Log only adapter errors.

-TV -> Use verbose (debug) logging. The log file records all activity that occurs while the adapter is producing or consuming messages.

+ -> Appends the trace information to the existing log file. Omit this argument to create a new log file.

file_path -> The full path to the adapter trace log. If you omit this keyword, the adapter creates the m4activeMQ.trc log file in the map directory

Append Log

Specifies the flag that indicates the action to be taken when the specified log file already exists. When set to true, the log messages are appended to the file. When set to false: the file is truncated, and the messages are written to the empty file. The default value is true.

Log File Name

This is the name of the log file, where the log messages are written. If not specified, the default log file name m4activeMQ.trc is used, and the file is stored to the directory in which the executed compiled map resides.

Java Message Service(JMS) headers

Enables users to include standard Java Message Service(JMS) headers and user-defined properties in the messages produced to the ActiveMQ broker and retrieve them from messages consumed from the broker.

Messages in link, which include header information, are always represented as JSON documents.

The version of the header is stored in the version JSON element, while the actual payload of the ActiveMQ messages are stored in the payload JSON element. Binary payloads are stored using hex digit pairs or base64 encoding.

The JMS headers are represented by JSON elements with elements that match the JMS headers, except they are in lowercase and use an underscore as a word separator.

The JMS properties are represented by JSON elements that start with an underscore ("_") character.

JSON Template

The following JSON template captures the version element, and all supported JMS headers, two user-defined message properties (title and size), and the payload element. This element can be used as the starting point to create JSON schemas for use in ActiveMQ adapter cards and nodes.

```
{
  "version": 1,
  "destination": "",
  "reply_to": "",
  "type": "",
  "delivery_mode": 0,
  "priority": 0,
  "message_id": "",
  "timestamp": 0,
  "correlation_id": "",
  "expiration": 0,
  "redelivered": false,
  "_title": "",
  "_size": 0,
  "payload": ""
}
```

Table 1. JMS header properties and command

Adapter property	Command	Source	Target
Message Header	-HDR version	✓	✓
Message Header Elements	-MHE list	✓	
Message Payload Encoding	-MPE encoding	✓	✓
Destination	-DES name		✓
Reply To	-RTO name		✓
Type	-TYP type		✓
Delivery Mode	-DM mode		✓
Priority	-PRI priority		✓
Message Id	-MID id		✓

Adapter property	Command	Source	Target
Timestamp	-TMS time		✓
Correlation Id	-CID id		✓
Expiration	-EXP duration		✓
Redelivered	-RED		✓

Message Header

Specifies the version of the JSON document used to represent ActiveMQ messages. The default value None means that the entire data passed to and from the adapter represents the message payload. No header information is present, and the message is not represented as a JSON header document. The value Version 1 indicates the JSON document version 1 is used. The corresponding adapter command is **-HDR version** (or **-HEADER version**).

Message Header Elements

Specifies a comma-separated list of elements to include in the JSON document representation of the consumed messages. Any whitespace characters in the list are NOT ignored and are considered part of the element names. The order of elements in the provided list will match the order of elements in the generated JSON document. Note that the version and payload elements must be explicitly included in the list for the message payload to be included in the JSON document. The corresponding adapter command is **-MHE list** (or **-MESSAGEHEADERELEMENTS list**)

Message Payload Encoding

Specifies the encoding of the payload element value in the JSON message representation. The corresponding adapter command is **-MPE encoding** (or **-MESSAGEPAYLOADENCODING encoding**).

The supported values are:

Text: The value is a string representation of the message payload. It assumes that the actual message payload is already in text format, encoded in the current system locale character set. The corresponding adapter command value is **text**.

JSON: The value is in JSON format. This can be an atomic JSON value, a JSON object, or a JSON array. It assumes that the actual message payload is already in JSON format, encoded in the current system locale character set. The corresponding adapter command value is **json**.

Hex Pairs: The value is a string consisting of consecutive hex-digit pairs, where each pair corresponds to a single byte from the message payload. The corresponding adapter command value is **hex_pairs**.

Base64: The value is a string that represents base64 encoded message payload's binary content. The corresponding adapter command value is **base64**.

Destination

Specifies the name of the destination queue to set for the messages produced. It corresponds to the JMSDestination standard JMS header. The value can also be set on a per-message basis using the destination element in the JSON message document. The corresponding adapter command is **-DES name** (or **-DESTINATION name**).

Reply To

Specifies the name of the response queue to set for the produced messages. It corresponds to the JMSReplyTo standard JMS header. The value can also be set on a per-message basis using the reply_to element in the JSON message document. The corresponding adapter command is **-RTO name** (or **-REPLYTO name**).

Type

Specifies the user-defined message type to set for the produced messages. It corresponds to the JMSType standard JMS header. The value can also be set on a per-message basis using the type element in the JSON message document. The corresponding adapter command is **-TYP type** (or **-TYPE type**).

Delivery Mode

Specifies the mode of delivery to set for the produced messages. It corresponds to the JMSDeliveryMode standard JMS header. The value can also be set on a per-message basis using the delivery_mode element in the JSON message document. The following values are supported:

Non Persistent: For instructing the broker that it is not required to store messages in transit to stable storage. The corresponding adapter command value is 1.

Persistent: For instructing the broker to store messages in transit in to stable storage. It is the default value, and the corresponding adapter command value is 2.

The corresponding adapter command is **-DM mode** (or **-DELIVERYMODE mode**).

Priority

Specifies the priority value to set for the produced messages. It corresponds to the JMSPriority standard JMS header. The value can also be set on a per-message basis using the priority element in the JSON message document. The corresponding adapter command is **-PRI priority** (or **-PRIORITY priority**).

Message Id

This property specifies the message identifier value to set for the messages that are produced. It corresponds to the JMSMessageID standard JMS header. The value can also be set on a per-message basis using the message_id element in the JSON message document. The corresponding adapter command is **-MID id** (or **-MESSAGEID id**).

Timestamp

This property specifies the timestamp value to set for the messages that are produced. It corresponds to the JMSTimestamp standard JMS header. The value can also be set on a per-message basis using the timestamp element in the JSON message document. The corresponding adapter command is **-TMS time** (or **-TIMESTAMP time**).

Correlation Id

This property specifies the correlation identifier value to set for the messages produced. It corresponds to the JMSCorrelationID standard JMS header. The value can also be set on a per-message basis using the correlation_id element in the JSON message document. The corresponding adapter command is **-CID id** (or **-CORRELATIONID id**).

Expiration

This property specifies the message expiration duration in milliseconds to set for the messages produced. It corresponds to the JMSExpiration standard JMS header. The value can also be set on a per-message basis using the expiration element in the JSON message document. The corresponding adapter command is **-EXP duration** (or **-EXPIRATION duration**).

EXPIRATION *duration*).

Redelivered

Specifies that the message is being resent. It corresponds to the JMSRedelivered standard JMS header. The value can also be set on a per-message basis using the delivery_mode element in the JSON message document. The corresponding adapter command is **-RED** (or **-REDELIVERED**).

Examples

Examples of GET map function

In the following example, assume that the adapter being used in a GET function is defined as follows:

This will pick the first message in the queue and transform according to the adapter mentioned in the output card.

```
GET("ACTIVEMQ", "-URL tcp://localhost:61616 -Q TEST_QUEUE -DT 5 -TV C:\temp\m4activemq.log")
```

Now, assume the GET map function was defined as follows (the specified queue name is not present):

```
GET("ACTIVEMQ", "-URL tcp://localhost:61616 -Q MISSING_QUEUE -DT 5 -TV C:\temp\m4activemq.log")
```

In this case, the adapter throws error: Queue MISSING_QUEUE doesn't exist, as it cannot find the queue name specified.

Examples of PUT map function

In the following example, assume that the adapter being used in a PUT function is defined as follows:

If the queue is not present, it will create the queue and put the message; if the queue is present, it will simply put the message into the queue TEST_QUEUE. Here, the message type is byte.

```
PUT("ACTIVEMQ", "-URL tcp://localhost:61616 -Q TEST_QUEUE -CQ -MT byte -TV C:\temp\m4activemq.log", in_data_type)
```

Now, assume that the message type is not mentioned.

```
PUT("ACTIVEMQ", "-URL tcp://localhost:61616 -Q TEST_QUEUE -CQ -TV C:\temp\m4activemq.log", in_data_type)
```

In this case, it will throw the following error: Message type must be blob/byte/Text.

Now, assume that the adapter is used in a PUT function, but the queue name mentioned is not present.

```
PUT("ACTIVEMQ", "-URL tcp://localhost:61616 -Q TEST_QUEUE -MT byte -TV C:\temp\m4activemq.log", in_data_type)
```

It will throw the following error message: Queue TEST_QUEUE does not exist. By default, the create queue flag is false. It needs to be set to true explicitly, if the queue is not present and the user wants to create the queue.

Apache HDFS Adapter

The Apache HDFS Adapter allows the IBM Transformation Extender to access the HDFS file system in Apache Hadoop environments.

Input cards and GET functions in the IBM Transformation Extender can be configured to read files from HDFS. Similarly, output cards and PUT functions can be configured to write data to HDFS files. Also, IBM Transformation Extender Launcher watches can be defined to detect creation of a new or modified file in HDFS, and to trigger the map to read and process that file.

- **[System requirements](#)**
See the release notes for the HDFS Adapter requirements.
- **[HDFS Adapter commands overview](#)**
The HDFS Adapter can be used to read and write files in HDFS. The adapter uses WebHDFS API to access HDFS. This is a REST API and uses HTTP protocol.
- **[HDFS Adapter commands](#)**
The Hadoop cluster must be configured to enable WebHDFS access. The WebHDFS server and port to which the adapter connects, the file to read or write, and various options for tuning these operations are specified by the adapter commands described in this documentation.
- **[HDFS Adapter transaction settings](#)**
This documentation describes how transaction settings in input and output HDFS cards affect the operation of the adapter.

System requirements

See the release notes for the HDFS Adapter requirements.

HDFS Adapter commands overview

The HDFS Adapter can be used to read and write files in HDFS. The adapter uses WebHDFS API to access HDFS. This is a REST API and uses HTTP protocol.

Note: The following is a high level overview of the HDFS Adapter commands. See [HDFS Adapter commands](#) for a detailed description of the adapter commands.

- **-S, -H and -P** commands - These commands specify the protocol scheme, the host, and the port to which to connect, respectively. The **-S** command is optional, and it defaults to *http*, which is the only supported value. Both the **-H** and the **-P** commands are required (no default).
- **-U** command - Specifies the name of the user known to HDFS.
- **-F** command - Specifies the path to the file in HDFS to read or write.
- **-CS** - command - Specifies the size (in bytes) of chunks of data the adapter reads from or writes to the file at a time.
- **-SCM** command -Enables special mode in a read or write operation where the entire file is read from or written to in a single chunk.
- **-IWM** command - Enables another special mode of write operation where the data is written to the target file immediately as it becomes available, instead of being buffered and written when the buffered chunks reach the specified size.
- **-BLS** command - Specifies the block size to use for the target file.
- **-BUS** command - Specifies the buffer size (in bytes) to use at the transport layer.
- **-OW** command - Used to allow overwriting the target file, if the file exists.
- **-CSRFH** command -Used to specify the header to set when Cross-Site Request Forgery prevention is enabled in HDFS.
- **-XATTR** command - Used to set extended attributes on the target file, or to specify extended attributes criteria for processing files in the Launcher.
- **-PER** command - Specifies the permissions to set on the target file.
- **-REP** command - Specifies the replication factor for the target file.
- **-TSL, -TSP, -KSL, -KSP, -AS, -AC and -VH** commands - Used to configure SSL.
- **-KM, -LCFL, -KCFL, -KAU, -KAP and -KCN** adapter commands - Used to configure Kerberos.
- **-BAU** and **-BAP** adapter commands - Used for basic HTTP authentication.
- **-T** command - This command, and its variants, is used to manage adapter tracing.

HDFS Adapter commands

The Hadoop cluster must be configured to enable WebHDFS access. The WebHDFS server and port to which the adapter connects, the file to read or write, and various options for tuning these operations are specified by the adapter commands described in this documentation.

Adapter commands summary list

Table 1. Adapter commands summary list

Short name	Long name	Description
-S	-SCHEME	Scheme
-H	-HOST	Host
-P	-PORT	Port
-U	-USER	User
-F	-FILE	File
-CS	-CHUNKSIZE	Chunk size
-SCM	-SINGLECHUNKMODE	Single chunk mode
-IWM	-IMMEDIATEWRITEMODE	Immediate write mode
-OW	-OVERWRITE	Overwrite
-PER	-PERMISSION	Permission flags
-REP	-REPLICATION	Replication factor
-CSRFH	-CSRFH	Cross-Site Request Forgery prevention header
-XATTR	-XATTR	Extended input and output attributes
-TSL	-TRUSTSTORELOCATION	Truststore full path
-TSP	-TRUSTSTOREPASSWORD	Truststore password
-KSL	-KEYSTORELOCATION	Keystore full path
-KSP	-KEYSTOREPASSWORD	Keystore password
-KP	-KEYPASSWORD	Key password
-AS	-AUTHSERVER	Authenticate server
-AC	-AUTHCLIENT	Authenticate client
-VH	-VERIFYHOSTNAME	Verify hostname
-KM	-KERBEROSMODE	Kerberos mode authentication
-LCFL	-LOGINCONFIGFILELOCATION	Login configuration file location
-KCFL	-KERBEROSCONFIGFILELOCATION	Kerberos configuration file location
-KAU	-KERBEROSAUTHUSER	Kerberos authentication user
-KAP	-KERBEROSAUTHPASSWORD	Kerberos authentication password
-KCN	-KERBEROSCONFIGNAME	Kerberos configuration namem
-BAU	-BASICAUTHUSER	Basic authentication user
-BAP	-BASICAUTHPASSWORD	Basic authentication password
-T	-TRACE	Adapter log full path - info level log
-T+	-TRACE+	Adapter log full path - append to log file
-TV	-TRACEVERBOSE	Adapter log full path - verbose level log
-TV+	-TRACEVERBOSE+	Adapter log full path - verbose level log - append to log file
-TE	-TRACEERROR	Adapter log full path - error log only
-TE+	-TRACEERROR+	Adapter log full path - error log only - append to log file

- **Protocol scheme -S (-SCHEME)**

Use the **-S** scheme command to determine the protocol scheme to use.

- **WebHDFS host name -H (-HOST)**

Use the **-H** hostname command to identify the WebHDFS host name.

- **[WebHDFS port number -P \(-PORT\)](#)**
Use the **-P** *port* command to identify the WebHDFS port number.
- **[Username -U \(-USER\)](#)**
Use the **-U** *user* command to identify the username, for simple username based authorization
- **[Path prefix -PP \(-PATHPREFIX\)](#)**
Use the **-PP** *prefix* command to identify the path prefix in the URL.
- **[Filename -F \(-FILE\)](#)**
The **-F** *filename* command indicates the path of the file to read from, or write to.
- **[Chunk size -CS \(-CHUNKSIZE\)](#)**
Use the **-CS** *size* command to indicate the chunk size in bytes.
- **[Single chunk mode -SCM \(-SINGLECHUNKMODE\)](#)**
Use the **-SCM** command to instruct the adapter to read data from the HDFS file and write data to the HDFS file in a single chunk.
- **[Immediate write mode -IWM \(-IMMEDIATEWRITEMODE\)](#)**
Use the **-IWM** command to instruct the adapter to write data to the target HDFS file as soon as it becomes available from the engine without buffering it for later write.
- **[Block size -BLS \(-BLOCKSIZE\)](#)**
Use the **-BLS** *size* command to set the block size, in bytes, to use for the target file when the adapter creates a new file.
- **[Buffer size -BUS \(-BUFFERSIZE\)](#)**
Use the **-BUS** *size* command to set the buffer size, in bytes, to use at the HTTP transport layer.
- **[Overwrite -OW \(-OVERWRITE\)](#)**
Use the **-OW** command to set target file overwrite.
- **[Permission flags -PER \(-PERMISSION\)](#)**
Use the **-PER** *flags* command to set permission flags on the target file created by the adapter.
- **[Replication factor -REP \(-REPLICATION\)](#)**
Use the **-REP** *factor* command to set the replication factor to specify for the target file created by the adapter.
- **[Cross-Site Request Forgery Prevention Header -CSRFH \(-CSRFH\)](#)**
Use the **-CSRFH** [*header[:value]*] command to specify the header in each HTTP request to comply with Cross-Site Request Forgery prevention when enabled in the HDFS.
- **[Extended attributes -XATTR \(-XATTR\)](#)**
Use the **-XATTR** *name=value[,name=value]** command to specify extended input and output attributes.
- **[Truststore file full path -TSL \(-TRUSTSTORELOCATION\)](#)**
The **-TSL** *path* command indicates the Truststore file full path for SSL handshake.
- **[Truststore password -TSP \(-TRUSTSTOREPASSWORD\)](#)**
The **-TSP** *password* command indicates the Truststore password.
- **[Keystore file path -KSL \(-KEYSTORELOCATION\)](#)**
Use the **-KSL** *path* command to specify the Keystore file full path.
- **[Keystore password -KSP \(-KEYSTOREPASSWORD\)](#)**
Use the **-KSP** *password* to specify the Keystore password.
- **[Key password -KP \(-KEYPASSWORD\)](#)**
Use the **-KP** *password* command to specify the Key password.
- **[Authenticate server -AS \(-AUTHSERVER\)](#)**
Use the **-AS** command to authenticate to the server.
- **[Authenticate client -AC \(-AUTHCLIENT\)](#)**
Use the **-AC** command for client certificate validation.
- **[Verify host name -VH \(-VERIFYHOSTNAME\)](#)**
Use the **-VH** command to instruct the adapter to verify the hostname value in the server certificate presented by the WebHDFS server during the SSL handshake.
- **[Kerberos mode -KM \(-KERBEROSMODE\)](#)**
Use the **-KM** command to perform Kerberos authentication and to perform WebHDFS operations as the authenticated user.
- **[Login configuration file location -LCFL \(-LOGINCONFIGFILELOCATION\)](#)**
Use the **-LCFL** *path* command to identify the login configuration file location.
- **[Kerberos configuration file location -KCFL \(-KERBEROSCONFIGFILELOCATION\)](#)**
Use the **-KCFL** command to specify the location of the Kerberos configuration file.
- **[Kerberos authentication user -KAU \(-KERBEROSAUTHUSER\)](#)**
Use the **-KAU** *username* command when the login configuration file is configured to prompt the user for username and password, as opposed to using ticket cache or keytab file.
- **[Kerberos authentication password -KAP \(-KERBEROSAUTHPASSWORD\)](#)**
Use the **-KAP** *password* command to authenticate the Kerberos authentication password.
- **[Kerberos configuration file -KCN \(-KERBEROSCONFIGNAME\)](#)**
Use the **-KCN** *name* command to specify the name of the configuration section in the login configuration file.
- **[Basic authentication user -BAU \(-BASICAUTHUSER\)](#)**
Use the **-BAU** *username* command when HTTP basic authentication is required on the connection.
- **[Basic authentication password -BAP \(-BASICAUTHPASSWORD\)](#)**
Use the **-BAP** *password* command when HTTP basic authentication is required on the connection.
- **[Adapter log file full path -T, -T+, -TV, -TV+, -TE, -TE+](#)**
Use the **-T**, **-T+**, **-TV**, **-TV+**, **-TE**, and **-TE+** commands to specify the adapter log file full path.

Protocol scheme -S (-SCHEME)

Use the **-S** *scheme* command to determine the protocol scheme to use.

The two supported schemes are *http* and *https*. Default is *http*.

WebHDFS host name -H (-HOST)

Use the **-H** *hostname* command to identify the WebHDFS host name.

This is the name of the HDFS Name Node server.

WebHDFS port number -P (-PORT)

Use the **-P** *port* command to identify the WebHDFS port number.

This is the port on which Name Node listens for WebHDFS HTTP requests. This port is typically 5870 or 50070, depending upon Hadoop distributions.

Username -U (-USER)

Use the **-U** *user* command to identify the username, for simple username based authorization

Path prefix -PP (-PATHPREFIX)

Use the **-PP** *prefix* command to identify the path prefix in the URL.

This is the part of the URL located between the *scheme://hostname:port/* part and the HDFS file path part. Default: *webhdfs/v1*

Filename -F (-FILE)

The **-F** *filename* command indicates the path of the file to read from, or write to.

In case of the Launcher, the path specified on the input card on which a source event watch is defined. The path consists of the directory path and glob pattern file name, for example: */user/user1*.txt*

Chunk size -CS (-CHUNKSIZE)

Use the **-CS** *size* command to indicate the chunk size in bytes.

The HDFS Adapter reads data from the HDFS file and writes data to the HDFS file in chunks of this size. Read and write for each chunk is a separate WebHDFS REST call. The value must be greater than zero. The maximum supported value is 2147483647. Default size: 65536

Single chunk mode -SCM (-SINGLECHUNKMODE)

Use the **-SCM** command to instruct the adapter to read data from the HDFS file and write data to the HDFS file in a single chunk.

Setting this property results in the adapter ignoring the value specified in the **-CS** command.

Immediate write mode -IWM (-IMMEDIATEWRITEMODE)

Use the **-IWM** command to instruct the adapter to write data to the target HDFS file as soon as it becomes available from the engine without buffering it for later write.

When the target card is configured to run in burst mode, the adapter immediately writes all available data in each burst to the target HDFS file.

This command is for Output operations only.

Block size -BLS (-BLOCKSIZE)

Use the **-BLS** *size* command to set the block size, in bytes, to use for the target file when the adapter creates a new file.

The value must be greater than 0 (zero). The maximum value supported by the adapter is 9223372036854775807. See your Hadoop distribution documentation for information about supported values in your HDFS environment.

This command is for Output operations only.

Buffer size -BUS (-BUFFERSIZE)

Use the **-BUS size** command to set the buffer size, in bytes, to use at the HTTP transport layer.

If this value is not specified, the default buffer size as defined in the HDFS is used. Contrast this to **-CS** which is the size of chunks of data the adapter uses to read or write to HDFS. The value must be greater than 0 (zero). The maximum supported value is 2147483647.

Overwrite -OW (-OVERWRITE)

Use the **-OW** command to set target file overwrite.

When the Overwrite command is set, the adapter overwrites the target file if **OnSuccess** card option is set to **Create**, or to **CreateOnContent**, and the file the adapter is about to create already exists. If the command is not specified, the adapter does not overwrite the target file but reports an error instead. This command is for Output operations only.

Permission flags -PER (-PERMISSION)

Use the **-PER flags** command to set permission flags on the target file created by the adapter.

The flag value is an octal (base 8) integer between 0 and 1777, inclusive. If the command is not specified, the target file is created with the default permissions flags configured in HDFS, typically 644.

This command is for Output operations only.

Replication factor -REP (-REPLICATION)

Use the **-REP factor** command to set the replication factor to specify for the target file created by the adapter.

The value must be greater than 0 (zero). The maximum value supported by the adapter is 32767, however, consult your Hadoop distribution documentation for information about supported values in your HDFS environment.

This command is for Output operations only.

Cross-Site Request Forgery Prevention Header -CSRFH (-CSRFH)

Use the **-CSRFH [header[:value]]** command to specify the header in each HTTP request to comply with Cross-Site Request Forgery prevention when enabled in the HDFS.

If the *header* argument is omitted, the X-XSRF-HEADER header is set to empty string value. If *header* is provided, but *value* is not the specified, the header is set to the empty string value. If both *header* and *value* are provided, the specified header is set to the specified value. The adapter sets the header in each HTTP request that it makes to WebHDFS server. See the HDFS documentation for more information about Cross-Site Request Forgery prevention.

Extended attributes -XATTR (-XATTR)

Use the **-XATTR name=value[,name=value]*** command to specify extended input and output attributes.

When used on the output, the adapter sets the specified extended attributes on the target file. If any of the attributes are already defined on the target file, they are overwritten with the specified new value.

When used on the input, it is only applicable in the Launcher when it is set on the input card on which source event watch is defined. In this case, the adapter inspects the watch file before it is reported as an event. This is to ensure that it has the specified extended attributes set to the specified values. Only those files that meet this criteria are reported.

For more information about extended attributes in HDFS see the HDFS documentation.

Truststore file full path -TSL (-TRUSTSTORELOCATION)

The **-TSL path** command indicates the Truststore file full path for SSL handshake.

Applicable only when the **-SCHEME** adapter command is set to *https* and the **-AS** adapter command is specified.

Default path:

JAVA_HOME/lib/security/cacerts

Truststore password -TSP (-TRUSTSTOREPASSWORD)

The **-TSP** password command indicates the Truststore password.

Applicable only when the **-SCHEME** adapter command is set to *https* and the **-AS** adapter command is specified.

Default password: *changeit*

Keystore file path -KSL (-KEYSTORELOCATION)

Use the **-KSL** *path* command to specify the Keystore file full path.

Applicable only when the **-SCHEME** adapter command is set to *https* and the **-AC** adapter command is specified.

Default path:

JAVA_HOME/lib/security/cacerts

Keystore password -KSP (-KEYSTOREPASSWORD)

Use the **-KSP** *password* to specify the Keystore password.

Applicable only when the **-SCHEME** adapter command is set to *https* and the **-AC** adapter command is specified.

Default password: *changeit*

Key password -KP (-KEYPASSWORD)

Use the **-KP** *password* command to specify the Key password.

Applicable only when the **-SCHEME** adapter command is set to *https*, the **-AC** adapter command is specified, and the key password differs from the keystore password.

Default password: *changeit*

Authenticate server -AS (-AUTHSERVER)

Use the **-AS** command to authenticate to the server.

The Authenticate Server command is applicable only when the **-SCHEME** adapter command is set to *https*. This command notifies the adapter that client server certificate validation will take place during the SSL handshake.

Authenticate client -AC (-AUTHCLIENT)

Use the **-AC** command for client certificate validation.

The Authentication Client command is applicable only when the **-SCHEME** adapter command is set to *https*. The command notifies the adapter that client certificate validation will take place during SSL handshake.

Verify host name -VH (-VERIFYHOSTNAME)

Use the **-VH** command to instruct the adapter to verify the hostname value in the server certificate presented by the WebHDFS server during the SSL handshake.

Applicable only when the **-S** command is set to *https* and the **-AS** command is specified.

Kerberos mode -KM (-KERBEROSMODE)

Use the **-KM** command to perform Kerberos authentication and to perform WebHDFS operations as the authenticated user.

Login configuration file location -LCFL (-LOGINCONFIFILELOCATION)

Use the **-LCFL** *path* command to identify the login configuration file location.

Applicable when the **-KM** adapter command is set. Specifies the location of the JAAS login configuration file which specifies the Kerberos security module and parameters to use for authentication for testing and debugging purposes. The preferred way for specifying the location of this file is by providing the `Djava.security.auth.login.config=file_path` JVM parameters for the Java process in which the adapter runs. When the adapter command is used, the adapter sets the `java.security.auth.login.config` system property each time it initializes a connection client to HDFS, and it applies to the entire JVM process. This is important in cases with multiple threads or multiple adapters running in the JVM. The operation of these adapters might be affected by the value of this system property. Each time this system property is set, it becomes the active value for all threads and adapters that run the JVM.

Kerberos configuration file location -KCFL (-KERBEROSCONFIGFILELOCATION)

Use the **-KCFL** command to specify the location of the Kerberos configuration file.

Applicable when the **-KP** adapter command is set. Use this adapter command only for testing and debugging. The preferred way for specifying the location of this file is by providing the `-Djava.security.krb5.conf=file_path` JVM parameter for the Java process in which the adapter runs. When the adapter command is used, the adapter sets the `java.security.krb5.conf` system property each time it initializes the connection client to HDFS and it applies to the entire JVM process. This is important for cases with multiple threads or multiple adapters running in the JVM. In such cases, operation might be affected by the value of this system property. Each time this system property is set, it becomes the active value for all threads and adapters running in the JVM. The name of this file is typically, `krb5.conf`.

Kerberos authentication user -KAU (-KERBEROSAUTHUSER)

Use the **-KAU** *username* command when the login configuration file is configured to prompt the user for username and password, as opposed to using ticket cache or keytab file.

Kerberos authentication password -KAP (-KERBEROSAUTHPASSWORD)

Use the **-KAP** *password* command to authenticate the Kerberos authentication password.

Used when the login configuration file is configured to prompt the user for username and password as opposed to using ticket cache or keytab file.

Kerberos configuration file -KCN (-KERBEROSCONFIGNAME)

Use the **-KCN** *name* command to specify the name of the configuration section in the login configuration file.

The Kerberos client authentication section is typically named *Client*.

Basic authentication user -BAU (-BASICAUTHUSER)

Use the **-BAU** *username* command when HTTP basic authentication is required on the connection.

This might be the case, for example, when connecting to the HDFS through a proxy server as opposed to connecting directly.

Basic authentication password -BAP (-BASICAUTHPASSWORD)

Use the **-BAP** *password* command when HTTP basic authentication is required on the connection.

This might be the case, for example, when connecting to the HDFS through a proxy server as opposed to connecting directly.

Adapter log file full path -T, -T+, -TV, -TV+, -TE, -TE+

Use the **-T**, **T+**, **-TV**, **-TV+**, **-TE**, and **-TE+** commands to specify the adapter log file full path.

-T *path*(-TRACE)
-T+ *path*(-TRACE+)
-TV *path*(-TRACEVERBOSE)
-TE *path*(-TRACEERROR)
-TE+ *path*(-TRACEERROR+)

Adapter log file full path. **-T** is for info level log **-TV** is for verbose (debug) level log and **-TE** is for logging errors only. If **+** is included, it means to append to the log file if it exists. If it does not exist a new file is created. The path argument can be omitted, in which case the default trace file name, **m4hdfs.mtr**, is used and the file is created in the executable directory of the map.

HDFS Adapter transaction settings

This documentation describes how transaction settings in input and output HDFS cards affect the operation of the adapter.

The only supported transaction scope for input and output cards is Map. This means that the specified transaction action takes place only once per card, per map, prior to completing the map.

OnSuccess and **OnFailure** actions for HDFS input and output cards are described as follows:

OnSuccess actions for input cards

The **OnSuccess** actions for HDFS input cards are **Delete**, **Keep**, and **KeepOnContent**.

- **Delete** - Delete the source file.
- **Keep** - Keep the source file.
- **KeepOnContent** - Keep the source file if it is not empty. If it is empty, delete the file.

OnFailure actions for input cards

The only supported **OnFailure** action for HDFS input cards is **Commit**. The **Commit** action completes the write operation by flushing any data still staged in the adapter but not yet written to the target file.

OnSuccess actions for output cards

The **OnSuccess** actions for HDFS output cards are **Create**, **CreateOnContent**, **!Create**, and **Append**.

- **Create** - Create the target file if target file does not exist. If the target file with the same name already exists, overwrite the file if the -OW adapter command is included, otherwise report an error.
- **CreateOnContent** - Same description as **Create**, except the file is not created if there is no data to be written to it.
- **!Create** - Do not create the target file, even if the target file does not exist. The data provided to the adapter is not sent to HDFS, but is silently ignored. If the target file exists, it is left intact.
- **Append** - Append data to the target file, if it already exists. If the target file does not exist, the file is created, and data is written to the file.

OnFailure actions for output cards

The **OnFailure** actions for HDFS output cards are **Commit**, and **Rollback**.

- **Commit** - Preserves all data that was written to the file prior to the failure.
- **Rollback** - Delete the target file if the target file was created by the adapter in the current map. Otherwise, preserve the target file including any data that was written to it in the current map.

Apache Kafka Adapter

With the Apache Kafka adapter, Transformation Extender maps can connect to a Kafka cluster to consume and produce messages. You also can configure Transformation Extender Launcher watches to detect the arrival of new messages on Kafka topics and trigger maps to process those messages.

See the release notes for the Kafka adapter requirements.

- [Authenticating Connection](#)
- [Adapter properties and commands](#)
This section lists the properties supported by the adapter.
- [Adapter commands for consumers and producers](#)
Kafka adapter commands for consumers and producers are valid for input data sources and output data targets. See the related Kafka configurations for both consumers and producers in the Apache Kafka documentation for additional details.
- [Adapter commands for producers](#)
Kafka adapter commands for producers are valid for output target data. For additional details, see the producer configuration information in the [Apache Kafka documentation](#).
- [Adapter commands for consumers](#)
Kafka adapter commands for consumers are valid for input data sources. For additional details, see the consumer configuration information in the [Apache Kafka documentation](#).

Authenticating Connection

Apache Kafka adapter as a source

As a source (in consumer), the adapter can receive Avro messages from Kafka in form of Avro **GenericRecord** objects and convert them to JSON strings.

Apache Kafka adapter as a target

As a target (in producer) the adapter can convert JSON strings to Avro **GenericRecord** objects based on the Avro schema provided by the user.

Adapter properties and commands

This section lists the properties supported by the adapter.

In maps:

- Use adapter commands on input cards and GET functions to configure the adapter to consume messages from Kafka topics.
- Use adapter commands on output cards and PUT functions to configure the adapter to publish messages to Kafka topics.

Many Kafka adapter commands are based on Apache Kafka consumer or producer configurations. Those adapter command descriptions include the name of the related Apache Kafka configuration. See the [Apache Kafka documentation](#) for details.

Apache Kafka adapter command aliases

Use KAFKA as the adapter command alias on input and output cards and in GET and PUT rules. For example:

Input source override execution command	-IAKAFKA card_num
Output target override execution command	-OAKAFKA card_num

Headers for data payloads

You can specify the optional **-HDR** command to prepend one of the following header versions to the messages. The numbers in brackets denote the number of bytes:

- [4] is 32-bit int
- [8] is 64-bit int

Version

Header structure

1

```
[4]           Total header size (excluding this field): 2*4 + size(key)
[4]
[4]key        Key size followed by the key value
```

2

```
[4]           Total header size (excluding this field): 4*4 + 2*8 + size(key) + size(topic)
[4]
[4]key        Key size followed by the key value
[4]topic      Topic size followed by the topic
[4]
[4]           Partition number
[4]
[8]           Offset
[8]           Timestamp
```

3

```
[4]           Total header size (excluding this field): 4*4 + 2*8 + size(key) + size(topic) + sum(2*4 + size(hkey_i) +
size(hvalue_i))
[4]
[4]key        Header Version: 3
[4]key        Key size followed by the key value
[4]topic      Topic size followed by the topic
[4]
[4]           Partition number
[4]
[8]           Offset
[8]           Timestamp
[4]
The size of the Kafka headers section (excluding this field), followed by an array of Kafka header key,
value pairs in the following format:
[4]hkey      Kafka header key size followed by Kafka header key
[4]hvalue    Kafka header value size followed by Kafka header value
```

The message header is always followed by the payload size and payload content:

```
[4]payload    Payload size followed by payload content
```

Adapter commands for consumers and producers

Kafka adapter commands for consumers and producers are valid for input data sources and output data targets. See the related Kafka configurations for both consumers and producers in the Apache Kafka documentation for additional details.

Server Hostname

Identifies one or more servers in a Kafka cluster that the adapter is to establish an initial connection with. After the initial connection, the adapter discovers and uses the full set of servers. If the connection to the first server fails, the adapter attempts to connect to each subsequent server in the list until it succeeds.

Related Kafka configurations: bootstrap.servers

The corresponding adapter command is **-SRV hostname:port [,hostname:port[,hostname:port, ...]]** (or **-SERVER hostname:port [,hostname:port[,hostname:port, ...]]**).

Topic Name

The target topic to publish to, or one or more source topics to consume from. Separate multiple topic names with spaces. This command is required unless producers identify the topic for each message separately using message header version 2 or later. See "Headers for data payloads" and the **-HDR** command.

topicname

Publish the target message to a partition of the topic selected by the Kafka cluster.

topicname:partition

Publish the target message to the specified partition of the topic.

To consume messages, specify one or more source topics to consume from, using any combination of the following:

topicname

Consume all source messages from the topic.

topicname:partition

Consume all source messages from the specified partition of the topic.

topicname:*

Consume source messages from all partitions of the topic.

topicname:partition-offset

Consume the source message at the specified offset of the specified partition of the topic.

- If the offset does not exist in the specified partition, the adapter consumes messages based on the policy specified by the [-AOR](#) command.
- If the -AOR command is not specified, the adapter consumes the message from the most recent offset position.

Client ID

A logical application name to identify the source of requests in Kafka server logs. The default value is a null string. The corresponding adapter command is -CID clientID (or - CLIENTID ClientID)

Related Kafka configurations: client.id

Header

The version of the header that precedes the message payload size and message payload data. See "Headers for data payloads" for details about the header versions. This is an optional property. The corresponding adapter command is -HDR {1 | 2 | 3} (or -HDR {1 | 2 | 3})

Security Protocol

Specifies the security protocol. Default is Plain Text. The corresponding adapter command is -SP {PLAINTEXT | SSL | SASL_PLAINTEXT| SASL_SSL} (or -SECURITYPROTOCOL {PLAINTEXT | SSL | SASL_PLAINTEXT| SASL_SSL}).

- PLAINTEXT: Use a plain connection.
- SSL: Use an SSL connection for host authentication and data encryption. With SSL connections, you must also provide the path (-TSL command) and password (-TSP command) of the truststore.
- SASL_PLAINTEXT: Use a Simple Authentication Security Layer (SASL) mechanism for authentication over a plain connection.
- SASL_SSL: Use a SASL mechanism for authentication over an SSL connection.

See the security section in the Apache Kafka documentation for more details.

Login Configuration File Location

Specifies the location of the Java Authentication and Authorization Service (JAAS) login configuration file, which contains information about the security model and parameters to use for authentication. The corresponding adapter command is -LCFL file_path (or -LOGINCONFIGFILELOCATION file_path). This command is valid when the -SP command is set to sasl_plaintext or sasl_ssl.

Use the -LCFL command only for testing and debugging. This command sets the java.security.auth.login.config system property each time it connects to Kafka, and the property applies to the entire JVM process and all threads and adapters that run in it.

In a production environment, specify the location of the login configuration file with the Java Virtual Machine (JVM) Djava.security.auth.login.config=file_path parameter for the Java process in which the adapter runs.

Kerberos Configuration File Location

Specifies the location of the Kerberos configuration file, which contains information about the security model and parameters to use for authentication. The corresponding adapter command is -KCFL file_path (or -KERBEROSCONFIGFILELOCATION file_path).

This command is valid when the -SP command is set to sasl_plaintext or sasl_ssl and the -SM command is set to gssapi. The file is typically named krb5.conf.

Use the -KCFL command only for testing and debugging. This command sets the java.security.krb5.conf system property each time it connects to Kafka, and the property applies to the entire JVM process and all threads and adapters that run in it.

In a production environment, specify the location of the Kerberos configuration file with the Java Virtual Machine (JVM) Djava.security.krb5.conf=file_path parameter for the Java process in which the adapter runs.

Truststore Location File Path

Specifies the full path to the truststore. The corresponding adapter command is **-TSL** *file_path* (or **-TRUSTSTORELOCATION** *file_path*). This command is valid only when the **-SP** command is set to **ssl** or **sasl_ssl**.

Truststore Password

Specifies the truststore password.

The corresponding adapter command is **-TSP** *password* (or **-TRUSTSTOREPASSWORD** *password*). This command is valid only when the **-SP** command is set to **ssl** or **sasl_ssl**.

Keystore Location File Path

Specifies the full path to the keystore. The corresponding adapter command is **-KSL** *file_path* (or **-KEYSTORELOCATION** *file_path*). This command is valid only when the **-SP** command is set to **ssl** or **sasl_ssl**.

Keystore Password

Specifies the key password. The corresponding adapter command is **-KP** *password* or **(-KEYPASSWORD** *password***)**.

Logical Message Mode

Specifies logical message mode, in which the adapter processes multiple physical Kafka messages as a single record (*a logical message*). In this mode, the message payload of each physical message within the logical message is preceded with the 4-byte size of the payload, regardless of whether the **-HDR** command is specified. The corresponding adapter command is **-LMM** **(-LOGICALMESSAGEMODE)**.

Add Profile Filename

Specifies the name of a file that contains additional Kafka producer or consumer configurations in Java™ properties format. The values specified in the file override those specified on the adapter command. The corresponding adapter command is **-APF** *filename* (or **-ADDPROPFILE** *filename*).

Add Profile

Specifies one or more Kafka producer or consumer configurations. Each *key=value* pair is separated by spaces. The values specified on the **-AP** command override the values specified on the adapter command and the configurations in the file specified by the **-APF** command. The corresponding adapter command is **-AP** *key=value [key=value [key=value]]* (or **-ADDPROP** *key=value [key=value [key=value]]*).

Logging

This property specifies the level of logging to use for the log (trace) file produced by the adapter.

The corresponding adapter command is:

-T [E|V] [+] *[file_path]*

-TRACE or **-T** -> Log adapter informational messages.

-TRACEERROR or **-TE** -> Log only adapter errors.

-TRACEVERBOSE or **-TV** -> Use verbose (debug) logging. The log file records all activity that occurs while the adapter is producing or consuming messages.

+ -> Appends the trace information to the existing log file. Omit this argument to create a new log file.

file_path -> The full path to the adapter trace log. If you omit this keyword, the adapter creates the m4jdbc.mtr log file in the map directory.

Key Handling Mode

Specifies the mode for handling message keys. The corresponding adapter command is **-KHM** {*bytes* | *string* | *object_string* | *avro_json*} (or **-KEYHANDLINGMODE** {*bytes* | *string* | *object_string* | *avro_json*}).

The supported modes are:

- *bytes*: The adapter assumes message keys to be represented as Java byte array instances and it passes message key bytes through without any modification. This applies to both the consumer and producer scenarios. This is the default mode.
- *string*: In consumer, the adapter assumes message keys to be provided as Java String instances and converts them to bytes using the current system encoding. In producer, the adapter converts bytes to Java String instances using the current system encoding.
- *object_string*: In consumer, the adapter assumes message keys to be provided as Java Object instances, calls **toString()** method on them to obtain their string representation and converts them to bytes using the current system encoding. In producer, this mode behaves the same as string mode.
- *avro_json*: In consumer, the adapter assumes message keys to be provided as Java Avro **GenericRecord** objects, calls **toString()** method on them to obtain their JSON string representation and converts them to bytes using current system encoding. In producer, the adapter converts bytes to Java String instances, and assumes that they are formatted as JSON documents. It then converts them to Avro **GenericRecord** instances based on the Avro schema specified with **-KASF** adapter command.

For all modes other than bytes mode, the adapter must be configured to use an appropriate deserializer (consumer case) or serializer (producer case) that can perform the respected key data deserialization and serialization. The adapter commands **-AP** and **-APF** can be used to specify the deserializer or serializer to use.

For example, for string mode, in case of consumer, the value **key.deserializer=org.apache.kafka.common.serialization.StringDeserializer** can be included in the **-AP** command to specify a deserializer that will deserialize consumed message key bytes to String format before providing them to the adapter. Similarly, in case of producer,

the value key.serializer=org.apache.kafka.common.serialization.StringSerializer can be included in the -AP command to specify a serializer that will receive message keys in String format from the adapter before serializing them as key bytes in the produced messages.

When the specified serializer or deserializer implementation resides in an external JAR that is not part of the standard Kafka client included with the adapter, the JAR and its dependencies need to be saved in the class path for the adapter. This can be done for example by adding them to extra or libs/extra sub-directory of the product home directory.

See the security section in the Apache Kafka documentation for more details.

Value Handling Mode

Specifies the mode for handling message values (payloads). The corresponding adapter command is -VHM {bytes | string | object_string | avro_json} (or -VALUEHANDLINGMODE {bytes | string | object_string | avro_json}).

The supported modes are:

- bytes: The adapter assumes message values to be represented as Java byte array instances and it passes message value bytes through without any modification. This applies to both the consumer and producer scenarios. This is the default mode.
- string: In consumer, the adapter assumes message values to be provided as Java String instances and converts them to bytes using the current system encoding. In producer, the adapter converts bytes to Java String instances using the current system encoding.
- object_string: In consumer, the adapter assumes message values to be provided as Java Object instances, calls **toString()** method on them to obtain their string representation and converts them to bytes using the current system encoding. In producer, this mode behaves the same as string mode.
- avro_json: In consumer, the adapter assumes message values to be provided as Java Avro **GenericRecord** objects, calls **toString()** method on them to obtain their JSON string representation and converts them to bytes using current system encoding. In producer, the adapter converts bytes to Java String instances, and assumes that they are formatted as JSON documents. It then converts them to Avro **GenericRecord** instances based on the Avro schema specified with -VASF adapter command.

For all modes other than bytes mode, the adapter must be configured to use an appropriate deserializer (consumer case) or serializer (producer case) that can perform the respected value data deserialization and serialization. The adapter commands -AP and -APF can be used to specify the deserializer or serializer to use.

For example, for string mode, in case of consumer, the value value.deserializer=org.apache.kafka.common.serialization.StringDeserializer can be included in the -AP command to specify a deserializer that will deserialize consumed message value bytes to String format before providing them to the adapter. Similarly, in case of producer, the value value.serializer=org.apache.kafka.common.serialization.StringSerializer can be included in the -AP command to specify a serializer that will receive message values in String format from the adapter before serializing them as value bytes in the produced messages.

When the specified serializer or deserializer implementation resides in an external JAR that is not part of the standard Kafka client included with the adapter, the JAR and its dependencies need to be saved in the class path for the adapter. This can be done for example by adding them to extra or libs/extra sub-directory of the product home directory.

See the security section in the Apache Kafka documentation for more details.

Adapter commands for producers

Kafka adapter commands for producers are valid for output target data. For additional details, see the producer configuration information in the [Apache Kafka documentation](#).

Acknowledgments Level

Specifies the level of acknowledgment required for message delivery to be considered successful. The corresponding adapter command is -ACK *level* (or -ACKNOWLEDGEMENTS *level*). The default acknowledgment level is 1.

0: No acknowledgment required. At this acknowledgment level, the -RET command is applicable.

1: Acknowledgment is required from the leader broker only.

-1

All: Acknowledgment is required from the leader broker and all replica brokers.

Related Kafka producer configuration: acks

Buffer Memory

Specifies the maximum amount of memory, in bytes, that the producer uses to buffer records that are queued to be sent to the server. The default buffer size is 33,554,432. The corresponding adapter command is -BM *bytes* (-BUFFERMEMORY *bytes*).

Related Kafka producer configuration: buffer.memory

Batch Size

The default size, in bytes, of a group of records being sent to the same partition in a single request. When a batch of records reaches this size, the producer sends the batch (regardless of the -LM setting). Records that exceed this size are sent immediately. The corresponding adapter command is -BS *bytes* (or -BATCHSIZE *bytes*).

A single publish operation sends multiple record batches: one batch for each partition on the server that the producer has data for.

The default batch size is 16384.

Related Kafka producer configuration: batch.size

Linger Milliseconds

Specifies the amount of time, in milliseconds, that the producer waits for additional records to arrive in order to batch the records. The default is 0 ms, meaning no delay. The corresponding adapter command is -LM *ms* (or -LINGERMILLISECONDS *ms*).

Related Kafka producer configuration: linger.ms

Compression Type

Specifies the compression type for full batches of data generated by the producer. The default is none. The corresponding adapter command is -CT {none | gzip | snappy | lz4} (or -COMPRESSIONTYPE {none | gzip | snappy | lz4}).

Related Kafka producer configuration: compression.type

Retries Count

Specifies the number of times the producer attempts to resend a message, in case the broker reports a transient error. The range of retries is 0 - 2147483647. The default is 0 retries. The corresponding adapter command is -RET *count* (or -RETRIES *count*).

Related Kafka producer configuration: retries

Enables Idempotence

Enables idempotence. Idempotent delivery ensures that messages are delivered exactly once to a particular topic partition during the lifetime of a single producer. The corresponding adapter command is -EI (or -ENABLEIDEMPOTENCE). When you specify this command:

- The retry count (-RET command) must be greater than zero.
- The acknowledgment level (-ACK command) must be -1 or all.

Otherwise, a configuration exception occurs.

Related Kafka producer configuration: enable.idempotence

Transactional ID

Enables transactional delivery. Together with idempotence (-EI command), allows producers to send data to multiple partitions so that either all messages are successfully delivered, or none of them are.

The corresponding adapter command is -TID *transactionalID* (or -TRANSACTIONALID *transactionalID*).

Related Kafka producer configuration: transactional.id

Key Avro Schema File file_path

Specifies location of the JSON file that represents Avro schema to use when converting message keys from JSON string representation to Avro **GenericRecord** instances when producing messages and -KHM avro_json command is specified. If the specified location is a relative file path, it is treated as the path relative to the directory in which the map is deployed. The corresponding adapter command is -KASF *file_path* (-KEYAVROSCHEMAFILE *file_path*).

Value Avro Schema File file_path

Specifies location of the JSON file that represents Avro schema to use when converting message values (payloads) from JSON string representation to Avro **GenericRecord** instances when producing messages and -VHM avro_json command is specified. If the specified location is a relative file path, it is treated as the path relative to the directory in which the map is deployed. The corresponding adapter command is -VASF *file_path* (-VALUEAVROSCHEMAFILE *file_path*).

JSON Avro Decoder Class Name

Specifies the Java class to use for converting JSON to Avro representation. This command is optional and if not specified the default class org.apache.avro.io.JsonDecoder is used. When specified, the provided value must be a fully qualified name of the Java class that implements a public constructor which takes two arguments: org.apache.avro.Schema which represents the Avro schema to use for the conversion and java.io.InputStream which represents the stream from which to read JSON content. The jar file that implements this class must be included in the classpath in use for the adapter.

The corresponding adapter command is -JADCN *name* (or -JSONAVRODECODERCLASSNAME *name*).

Adapter commands for consumers

Kafka adapter commands for consumers are valid for input data sources. For additional details, see the consumer configuration information in the [Apache Kafka documentation](#).

Group ID

Specifies the unique string that identifies the consumer group that a consumer belongs to. This property enables Kafka group management, which is used when the -TP command specifies only topic names without topic partitions. The default is a null string. The corresponding adapter command is -GID *groupID* (or -GROUPID *groupID*).

Related Kafka consumer configuration: group.id

Offset Commit Strategy

Specifies the policy that the adapter uses to commit an offset when it consumes a message. The corresponding adapter command is -OCS (or -OFFSETCOMMITSTRATEGY).

The policy is one of the following:

- Manual: The Kafka adapter commits the offset when it commits the transaction, as defined by the Transaction Scope setting on the input card of the map. This is the default.
- Auto: The Kafka cluster periodically commits the offset.
- Never: Neither the Kafka cluster nor the Kafka adapter commits the offset.

Offset Commit Retry Count

Specifies the number of times to ignore offset commit failures and proceed consuming messages. The corresponding adapter command is -OCRC (-OFFSETCOMMITTRYCOUNT).

When the offset commit operation fails and the specified count is not reached, the adapter increments the internal counter of failed commits and proceeds to consume messages as if the commit was successful. If the error topic was specified and the value for this command is different from 0, the adapter sends the error message with the topic, partition, and offset information of the failed commit operation to the error topic. The adapter retries to commit the pending offsets again the next time it is scheduled to perform the commit operation. The default value for this command is 0, which instructs the adapter to treat offset commit failures as fatal severity errors and not attempt retries. The special value S is supported and is used to specify unlimited number of retries.

Skip Duplicates

Instructs the adapter to keep track of all messages that it consumed in the current process, and to ignore messages that it has previously consumed. The corresponding adapter command is -SD (-SKIPDUPPLICATES).

q

If the error topic is configured, the adapter sends error messages indicating detected duplicate messages (their topic, partition and offset) to the error topic.

-OFFSETCOMMITTRYCONSTRAINT value [value] -OCC value [value]

Specifies constraints the adapter must meet when committing pending offsets. One of the following two values can be specified, or both can be specified, separated by a space:

apo: Stands for Assigned Partitions Only and means that the adapter consumer should commit only those pending offsets that belong to the partitions currently assigned to it. The offsets that belong to a partition that the consumer owned at some earlier point but that has subsequently been assigned to another consumer are skipped.

ssc: Stands for Skip Stale Commits and means that the adapter should not commit pending offsets on a partition that are older than the currently committed offsets on the same partition, for the same consumer group. Such old offsets are considered stale and are possible if the consumer loses a partition due to partition rebalancing, and another consumer in the same consumer group is assigned the partition and consumes and commits offsets past the last offset consumer by the original consumer. If the original consumer later regains ownership of the partition, the offsets that it still considers pending for that partition will be stale. If this constraint is specified and the consumer detects stale offsets, it does not commit them and it discards (forgets) them.

-LOGERRORMESSAGES (-LEM)

Specifies that Kafka error messages published to the error topic should also be written to the adapter trace file.

Error Message Format Version

Specifies the format of messages sent to the error topic. The corresponding adapter command is -EMFV version (or -ERRORMESSAGEFORMATVERSION version).

The supported values are:

- 1: The default version, each message is in format t:p:o where t, p and o are the topic, partition and offset of the consumed message for which the processing failed and for which the error record is produced to the error topic
- 2: Each message is in JSON format: {"topic": "t", "partition": "p", "offset": "o", "code": "c"} where t, p and o are the topic, partition and offset values of the consumed message for which the processing failed, and c is the error code indicating the reason for the failure, and is one of the following:
 - DUPLICATE: Indicates the message was skipped because it was already consumed earlier for the same consumer group in the current process, and -SKIPDUPPLICATES (-SD) adapter command was specified COMMIT_FAILED - indicates the message was processed successfully, but the offset commit operation for it failed ROLLBACK - indicates that an error was reported while processing the specified message, for example when performing database write operation in one of the map outputs in the same transaction scope in which the message was consumed.

Fetch Minimum Bytes

Specifies the minimum amount of data, in bytes, that must accumulate before the server returns the data to the consumer. The default data size is one byte, which means the server responds to a data request as soon as any data is available to return. The corresponding adapter command is -FMIB size (or -FETCHMINBYTES size).

Related Kafka consumer configuration: `fetch.min.bytes`

Fetch Maximum Bytes

Specifies the maximum amount of data, in bytes, that the server should return for a fetch request. The corresponding adapter command is -FMAB size (or -FETCHMAXBYTES size).

Related Kafka consumer configuration: `fetch.max.bytes`

Auto Offset Reset

The offset position to automatically use when no previous offset exists. The corresponding adapter command is -AOR (-AUTOOFFSETRESET).

The offset is one of the following:

- Latest: Starts consuming new messages from the most recent offset. This is the default.
- Earliest: Starts consuming from the oldest available record by automatically resetting the offset to the earliest offset.
- None: Reports an error.

Related Kafka consumer configuration: auto.offset.reset

Synchronized

Specifies that the Launcher listener thread is to operate synchronously. The listener waits to notify a map of an event (a message added to a topic) until the map acknowledges that it processed the previous event. The corresponding adapter command is -SYNC (or -SYNCHRONIZED).

If the map fails, the listener does not report subsequent events unless the -ETP command is specified. With the -ETP command, when the Launcher listener records the failed event on the error topic, the listener is unblocked and proceeds to report new events.

Error Topic

Specifies the name of the error topic where messages that result in an error are recorded. The corresponding adapter command is -ETP *topic* (or -ERRORTOPIC *topic*).

- The Command Server records the error at the time of transaction rollback. In a map, the input card Transaction_Scope setting controls when a transaction failure is processed by the adapter.
- The Launcher records the error when the listener thread is blocked because it's waiting for the status of the event processing, as specified by the -SYNC command.

After recording the error, the listener resumes processing new events.

Isolation Level

Controls the visibility of messages that are part of a transaction. The corresponding adapter command is -IL (or -ISOLATIONLEVEL).

- read_committed: Only messages from transactions that are committed (and messages that were not part of a transaction) are visible to consumers.
- read_uncommitted: All messages are visible to consumers, even if they were part of an aborted transaction. This is the default.

Related Kafka consumer configuration: isolation.level

Quantity

Specifies the number of messages to consume, or S to consume all available messages. The corresponding adapter command is -QTY (or -QUANTITY).

The default value is 1. If you specify a value other than 1, you must set the FetchAs input card setting to Burst and the FetchUnit input card setting to 1 in the map.

Listen

Specifies the wait time for messages. The corresponding adapter command is -LSN (-LISTEN)

- Seconds: The number of seconds that the consumer waits for messages to arrive.
- S: Wait for an unlimited time for messages to arrive. This is the default.
- 0: Consume all available messages and do not wait for new messages to arrive.

Logical Message Count

Specifies the logical message count. Specifies the number of Kafka messages to concatenate and return as a single logical message from the adapter. By default, the adapter returns each Kafka message as a separate logical message. To concatenate all available messages, specify 0. The corresponding adapter command is -LMC *count* (or -LOGICALMESSAGECOUNT *count*).

This command is valid only in logical message mode (-LMM command). Logical message mode is not valid in a Launcher scenario.

Logical Message Bytes

Logical message buffer-size limit, in bytes. The adapter buffers messages until it exceeds this limit, then returns all buffered messages as a single logical message. The corresponding adapter command is -LMS *bytes* (or -LOGICALMESSAGESIZE *bytes*)

This command is valid only in logical message mode (-LMM command). Logical message mode is not valid in a Launcher scenario.

Deserialize Failure Mode

Specifies deserialize failure mode. It determines the action to take when a deserializer registered with the adapter throws exception while deserializing Kafka messages before providing them to the consumer in the adapter. The corresponding adapter command is -DFM {fail | warn | ignore} (or -DESERIALIZEFAILUREMODE {fail | warn | ignore}).

The mode is one of the following:

- fail: The original deserialization exception is re-thrown. This is the default mode.
- warn: The deserialization error is logged (presuming trace command is provided) but the value is serialized as zero-length byte array.

- ignore: Same as warn, except the error is not logged.

Empty Key Substitution value

Specifies literal string value the adapter will present to the map as the key of the message when it is retrieved from topic as null or empty array value. The corresponding adapter command is -EKS value (or -EMPTYKEYSUBSTITUTION value).

This includes null values obtained for messages which failed deserialization in the deserialized registered with the consumer, when -DFM warn or -DFM ignore command was specified. For example, if processing Avro messages from a topic with keys in JSON format, and some of them are invalid keys that do not comply with the Avro schema associated with them, then the combination of commands -DFM ignore -EKS {} will result in adapter ignoring deserialization errors, and returning nulls as the message keys. Subsequently it will replace those nulls with {} string literals prior to returning them from the consumer.

Empty Value Substitution value

Specifies literal string value the adapter will present to the map as the value of the message when it is retrieved from topic as null or empty array value. The corresponding adapter command is -EVS value (or -EMPTYVALUESUBSTITUTION value).

This include null values obtained for messages which failed deserialization in the deserialized registered with the consumer, when -DFM warn or -DFM ignore command was specified. For example, if processing Avro messages from a topic as JSON messages, and some of them are invalid messages that do not comply with the Avro schema associated with them, then the combination of commands -DFM ignore -EVS {} will result in adapter ignoring deserialization errors, and returning nulls as the message values. Subsequently it will replace those nulls with {} string literals prior to returning them from the consumer.

Avro JSON Conversion Mode

Specifies the mode for converting Avro records to JSON representation. The default mode is Simple (case-insensitive) and results in producing the default simple JSON. The other mode is Strict (case-insensitive) and results in producing JSON in compliance with the Avro schema. Avro schema must be specified when using Strict mode. When this command is omitted, the default value is Simple. In most cases, the two modes will produce equivalent JSON representation, but in some cases, such as when the Avro schema contains elements of union types, the JSON produced in Strict mode will be more verbose. That will make it less convenient for processing, but it will allow converting it back to Avro in compliance with the same Avro schema.

The corresponding adapter command is -AJCM mode (or -AVROJSONCONVERSIONMODE mode).

Avro JSON Encoder Class Name

Specifies the Java class to use for converting Avro to JSON representation. This command is optional and if not specified the default class org.apache.avro.io.JsonEncoder from the Apache Avro library is used internally by the adapter. When specified, the provided value must be a fully-qualified name of the Java class that implements a public constructor which takes two arguments: org.apache.avro.Schema which represents the Avro schema to use for the conversion, and java.io.OutputStream which represents the stream to which to write JSON content. The jar file that implements this class must be included in the classpath in use for the adapter.

The corresponding adapter command is -AJECN name (or -AVROJSONENCODERCLASSNAME name).

Amazon S3 Adapter

With the Amazon S3 or Amazon Simple Storage Service adapter, a user can access S3 buckets to fetch the stored data and send new data to S3 buckets for storing.

- [Introduction](#)
This document provides an overview and the functional specifications of the Amazon S3 Adapter.
- [Adapter Properties and Commands](#)
This chapter provides a detailed description of the adapter.
- [Examples](#)

Introduction

This document provides an overview and the functional specifications of the Amazon S3 Adapter.

An S3 bucket is a public cloud object-storage offering. In Amazon S3, data is stored as an object, which is the fundamental storage unit. An object is comprised of data and the descriptive metadata of the data. These objects are organized into buckets. Like file folders contain files, these buckets store objects. A single object can be up to 5 terabytes in size. The S3 buckets are located at various geographic regions as served by Amazon Web Services (AWS).

The function of an Amazon S3 adapter is to fetch objects or data from S3 buckets and send data to S3 buckets for storing as objects. The S3 adapter needs to be first authenticated by an AWS access key and secret key (based on bucket permission) for gaining access to an S3 bucket. The AWS denotes every geographic region by a region-code. This code needs to be in the exact format to that of AWS's when specifying a region in the adapter properties.

The Amazon S3 adapter does not support the predefined-schema-import option. An AWS Identity and Access Management (IAM) user can be created in AWS for accessing the S3 services. This user represents a person or application that will interact with AWS. It is not the same as the AWS account root user. For more information, refer: https://docs.aws.amazon.com/IAM/latest/UserGuide/id_users.html.

Amazon S3 adapter as a source

Amazon S3 adapter as a source:

- Reads a Blob
- Reads and deletes a Blob

Amazon S3 adapter as a target

- Creates an Object
- Overrides an Object

Troubleshooting large cloud object storage files

When you download large multi-megabyte files from Cloud Object Storage, the complete file is not downloaded. The S3 adapter saves only a portion of the designated file instead of the complete file. The following error message occurs during map processing:

Warning: Not all bytes were read from the S3ObjectInputStream, aborting HTTP connection. This is likely an error and may result in sub-optimal behavior. Request only the bytes you need via a ranged GET or drain the input stream after use.

The error message indicates that the S3 adapter closes the connection prematurely without fully downloading the file content.

Adapter Properties and Commands

This chapter provides a detailed description of the adapter.

Access Key

Specifies the key of AWS user for the connection. This access key is used to sign programmatic requests to AWS API calls through the adapter. The corresponding adapter command is **-AK** (or **-ACCESSKEY**).

Secret Key

Specifies the key of AWS user for the connection. Like the username and password, both the access key ID and secret access key are mandatory to authenticate S3 object for a Get or Put request. The corresponding adapter command is **-SK** (or **-SECRETKEY**).

Region Name

Specifies the name to connect to an AWS region for the IAM user account. The corresponding adapter command is **-R** (or **-REGION**).

Bucket Name

Specifies the name in which an S3 object would reside. During the action configuration, S3 Adapter provides options to select bucket name among available buckets. The corresponding adapter command is **-B** (or **-BUCKET**).

Folder Name

Specifies the folder name where S3 object may reside inside a bucket. This property is optional because an object may directly reside inside a bucket or under folder. When the folder property is not specified, adapter takes the direct path of the bucket to access the object. Otherwise, the folder name is prefixed to the key name. The corresponding adapter command is **-F** (or **-FOLDER**).

Object Name

Specifies the name of the S3 object to be accessed. A key can reside directly in a bucket or a folder inside the bucket. The corresponding adapter command is **-K** (or **-KEY**).

Content Type

Specifies the field that indicates the media type of the entity-body sent to Amazon S3. If the type is not set, adapter takes the default type, which is a standard MIME-type that describes the format of the content. For more information, go to https://www.w3.org/Protocols/rfc1341/4_Content-Type.html.

Type: String

Default: binary/octet-stream

Valid Values: MIME types

Constraints: None

The corresponding adapter command is **-CT** (or **-CONTENTTYPE**).

Content Language

Sets the Content-Language HTTP header, which specifies the natural language(s) of the intended audience for the enclosed entity. If no language is set, S3 adapter takes the default value 'en'. The corresponding adapter command is **-CL** (or **-CONTENTLANG**).

Encryption

Specifies the server-side encryption algorithm while encrypting the object using the AWS-managed keys. When the property is not specified, data uploaded to S3 remains unencrypted. The corresponding adapter command is **-E** (or **-ENCRYPTION**).

Valid Values: AES256, KMS.

AES256: 256 bit – Advanced Encryption Standard is a data or file encryption technique. The result of the process is downloadable in a text file. AES performs all its computations on bytes rather than bits. Hence, AES treats the 256 bits of a plaintext block as 16 bytes. It uses 14 rounds for 256-bit keys. For more information, see <https://docs.aws.amazon.com/AmazonS3/latest/dev/UsingServerSideEncryption.html>.

KMS: It represents the AWS Key Management Service (AWS KMS). The master keys that the user creates in AWS KMS are used to encrypt or decrypt the object uploaded to S3. For more information on KMS encryption technique, see <https://docs.aws.amazon.com/kms/latest/developerguide/concepts.html>.

Logging

Specifies the level of logging to use the log (trace) file produced by the adapter. The default is Off. The value Information means log informational, the value Errors Only means log error messages only, and the value Verbose means log debug and trace level messages along with the informational and error messages.

Logging can be specified using command:

-T [E|V] [+][file_path]

-T Log adapter informational messages.

-TE Log only adapter errors during map execution.

-TV Use verbose (debug) logging. The log file records all activity that occurs while the adapter is producing or consuming messages.

+ Appends the trace information to the existing log file. Omit this keyword to create a new log file.

file_path The full path to the adapter trace log. If you omit this keyword, the adapter creates the m4azblob.mtr log file in the map directory.

Append Log

Specifies the flag that indicates the action to take when the specified log file already exists. When set to true, the log messages are appended to the file. When set to false, the file is truncated, and the messages are written to the empty file. The default value is true.

Read Mode

Specifies the mode of operation for reading from the specified bucket. The supported values are:

- List: Return a list of keys of the objects detected in the bucket, separated by a linefeed character
- Keep: Return the contents of the object with the specified key and preserve the object in the bucket after reading it
- Delete: Return the contents of the object with the specified key and delete the object in the bucket after reading it

The default value is Keep.

The corresponding adapter command is -RM list|keep|delete (or -READMODE list|keep|delete). The command values are case-insensitive.

Key Prefix

Specifies the prefix to use for filtering objects from the specified bucket when the Read Mode property is set to value List. Only the objects with keys that begin with the specified prefix are included in the returned list. By default, no filtering is applied, and all the objects are returned in the list.

The corresponding adapter command is -KP prefix (or -KEYPREFIX prefix) where prefix is the key prefix value to use for the filtering.

For example, presume the bucket contains objects with the following keys:

key1

keyX

mykey1

mykeyX

test.txt

If -RM list -KP mykey commands are specified, the adapter will return the following list:

mykey1

mykeyX

Key Delimiter

Specifies the delimiter to use for filtering objects from the specified bucket when the Read Mode property is set to value List. Only the objects with keys that do not contain the specified delimiter will be included in the returned list. By default, no filtering is applied, and all the objects are returned.

The corresponding adapter command is -KD delimiter (or -KEYDELIMITER delimiter) where delimiter is the key delimiter value to use for the filtering.

For example, presume the bucket contains objects with the following keys:

dir1/obj1

dir1/obj2

dir1/subdirA/obj1

key1

key2

keyX

mykey1

mykeyX

test.txt

If **-RM list -KD** / commands are specified, the adapter will return the following list:

key1

key2

keyX

mykey1

mykeyX

test.txt

If **-RM list -KP /dir1/ -KD** / commands are specified, the adapter will take the specified key prefix **dir1/** into account as well, and will return the following list:

dir1/obj1

dir1/obj2

Endpoint

Specifies the endpoint to connect and access the S3 or S3 compatible buckets by the Amazon S3 adapter. If not specified, Amazon S3 adapter connects to the Amazon S3 Service endpoint based on the region specified.

To access an S3 compatible bucket hosted by an object storage service other than the default Amazon S3 service, Endpoint option must be specified. Region option is optional. The corresponding adapter command is **-EP** (or **-ENDPOINT**).

Log File Path

Specifies the location of the log file to which the log messages are written. If not specified, the default log file name m4s3.trc is used, and the file is stored to the directory in which the executed compiled map resides.

Convert Data

Specifies to convert data to parsed form. if enabled, support for converting CSV, Avro, Excel, Parquet and JSON documents into sets of data records in the CSV format. This means that the data produced by the converters can be loaded directly into relational databases.

Data Format

Specifies the data format. This is a mandatory property. Select one of the following as per the data format type. This property field is applicable when Convert Data property toggle is enabled.

- CSV
- Excel
- Avro
- Parquet
- JSON

Header

If enabled, specifies whether the CSV data contains a header row.

Quote

Specifies the quote character. Default value is ".

Delimiter

Specifies the field delimiter character. Default value is ,.

Escape

Specifies the escape character.

Separator

Specifies the row separator characters. Default value is \r\n.

Character Set

Specifies the character set of the data. Default value is UTF-8.

Limit

Specifies the maximum number of records to return.

Sample Size

Specifies the number of records to analyze when determining the data structure. Default value is 100.

Worksheet Index

Specifies the index of the worksheet. The index of the first worksheet is 1 (default).

Select Source Tables

Select which arrays of objects in the AVRO data you wish to map to tables in the target.

Find Array

If enabled, specifies that the output will iterate on the first array found in the JSON document.

Array Path

Specifies the path of the JSON array within the document. Omit if the document is an array. For example, "/resources".

Examples

GET example

GET example without **-FOLDER/-F** command:

```
GET("S3", " -AK AKSOMEACCESSKEY -SK SkSomeSecretKey/dBLo -R US_EAST_1 -B bucket-for-test -K persons.csv", inputdata)
```

As the **-FOLDER/-F** command is not present, the adapter looks for the S3 object directly under the mentioned bucket.

GET example with **-FOLDER/-F** command:

```
GET("S3", " -AK AKSOMEACCESSKEY -SK SkSomeSecretKey/dBLo -R US_EAST_1 -B bucket-for-test -F testfolder -K persons.csv", inputdata)
```

In this case, the adapter prepends the folder name to the key name passed, and the object is accessed from the folder residing under the specified bucket.

PUT example

PUT example without **-ENCRYPTION** command:

```
PUT("S3", " -AK AKSOMEACCESSKEY -SK SkSomeSecretKey/dBLo -R US_EAST_1 -B bucket-for-test -K persons.csv ", inputdata)
```

As **-ENCRYPTION** command is not present, the object uploaded to Amazon S3 is stored in its raw format.

PUT example with **-ENCRYPTION** command:

```
PUT("S3", " -AK AKSOMEACCESSKEY -SK SkSomeSecretKey/dBLo -R US_EAST_1 -B bucket-for-test -K persons.csv -ENCRYPTION KMS ", inputdata)
```

In this case, the adapter validates the algorithm passed to it and uses the same to set object metadata, which stores the data using the opted encryption technique to Amazon S3.

Amazon SQS Adapter

With the Amazon SQS or Amazon Simple Queue Service adapter, a user can access Amazon Simple Queue Services. The adapter connects to the fully managed Amazon Simple Queue Service system, using AWS keys. The SQS adapter can send and fetch messages to and from the queues to which a user has access.

Overview

This document provides an overview and the functional specifications of the Amazon SQS Adapter. The Amazon Simple Queue Service (SQS) is a distributed message-queueing service supporting programmatic sending of messages via web service applications. This message-queueing service is entirely manageable and scalable and can dissociate microservices, distributed systems, and server-less applications. The function of an Amazon SQS adapter is to access queues located at various Amazon Web Services-served geographic regions to fetch messages from standard queues and also store messages in standard queues in Amazon SQS.

IAM users can access SQS messages if they have the correct AWS access key and secret key, and also the queue-access permission. The SQS Adapter supports the sending and fetching of message attributes along with the message texts. SQS adapter supports standard queues having at-least-once delivery. For more information, see <https://aws.amazon.com/sqs/>.

The Amazon SQS adapter does not support the predefined-schema-import option. An AWS Identity and Access Management (IAM) user can be created in AWS for accessing the SQS services. This user represents a person or application that will interact with AWS. It is not the same as the AWS account root user. For more information, see https://docs.aws.amazon.com/IAM/latest/UserGuide/id_users.html.

Authenticating connections

The Amazon SQS adapter uses AWS SDK programmatically, to verify the user identity in programmatic calls, the user needs to provide AWS access keys. Typically, access keys are either temporary (short-term) credentials or long-term credentials, such as those for IAM users or the root user of an AWS account. In many cases, there are alternatives to long-term keys to consider.

Amazon SQS adapter as a source

Amazon SQS adapter as a source supports:

- Read message from the queue.
- Read message from the queue and delete after reading.

Amazon SQS adapter as a target

Amazon SQS adapter as a target supports:

- Send message to the existing queue.
- Send message to the queue and create the queue, if does not exist.

Adapter properties and commands

This section provides the details of the adapter properties to be set to define connections and actions and import schemas in the new web UI of IBM Sterling Transformation Extender.

Access key

Specifies the access key of AWS user for the connection. This is a mandatory property. The access key is used to sign programmatic requests to AWS API calls through the adapter. The corresponding adapter command is -AK *access_key* (or -ACCESSKEY *access_key*).

Secret key

Specifies the secret key of AWS user for the connection. For the username and password, both the access key ID and secret access key are mandatory to authenticate SQS queue GET or PUT request. The corresponding adapter command is -SK *secret_key* (or -SECRETKEY *secret_key*).

Region name

Specifies the region name to connect to an AWS region for the IAM user account. This is a mandatory property. The corresponding adapter command is -R *region_name* (or -REGION *region_name*).

Queue

Specifies the SQS queue name to or from which messages need to be accessed. This is a mandatory property. During action configuration, SQS adapter provides options to select queue name among available queues in the provided region. The corresponding adapter command is -Q *queue_name* (or -QUEUE *queue_name*).

Create queue

If specified, this property indicates the SQS adapter to create a queue with the name passed in the 'queue_name' property in case it does not exist. This property is optional. When the CreateQueue property is not specified, adapter checks if the queue exists; if not, it throws an error. The corresponding adapter command is -CQ (or -CREATEQUEUE).

Attribute

Specifies if the SQS message should be accessed along with the corresponding message attributes. When the attributes flag is set to true in output card, the passed data should be in predefined JSON format, else data should be in a plain text message.

```
{"message": "text message", "attributes": [{"name": "attribute_name", "type": "String", "value": "attribute_value"}]}
```

When the attributes property is present in input card, SQS adapters get the data in the above JSON format; otherwise, the text message is fetched from SQS queue. If it is omitted, attributes are not used. The corresponding adapter command is -A (Or -ATTRIBUTES). For more information on Amazon SQS message attributes, see <https://docs.aws.amazon.com/AWSSimpleQueueService/latest/SQSDeveloperGuide/sqs-message-attributes.html>.

Timeout

Specifies the time duration (in seconds) to wait for before retrieving the message from the queue. The timeout property of SQS adapter is applicable and utilized only in IBM Sterling Transformation Extender deployments. The SQS adapter would listen to the queue for specified seconds. The corresponding adapter command is -LSN *name*.

Limit

Specifies the total number of messages to consume. Minimum value should be 1. The value S means all available messages. If no limit is specified, adapter fetches only one message at a time from the queue. The limit property is applicable and utilized only in IBM Sterling Transformation Extender deployments. The corresponding adapter command is -QTY *limit*.

Logging

Specifies the level of logging to use for the log (trace) file produced by the adapter. The default is Off. The value Information means log informational, the value Errors Only means log error messages only, and the value Verbose means log debug and trace level messages along with the informational and error messages.

The corresponding adapter command is:

```
-T [E|V] [+|] [file_path]
```

- T** -> Log adapter informational messages.
- TE** -> Log only adapter errors.
- TV** -> Use verbose (debug) logging. The log file records all activity that occurs while the adapter is producing or consuming messages.
- +** -> Appends the trace information to the existing log file. Omit this argument to create a new log file.
- file_name** -> Specifies the location of the log file to which to write log messages. If not specified, the adapter creates the m4sq.sqs.log file in the map directory.

Append log

Flag indicating what to do if the specified log file already exists. When set to true, the log messages are appended to the file. When set to false, the file is truncated, and the messages are written to the empty file. The default value is true.

Log file name

Specifies the location of the log file to which to write log messages. If not specified, the default log file name m4sq.sqs.log is used, and the file is stored to the directory in which the executed compiled map resides.

Examples

Examples of the GET map function

In the following example, assume that the adapter being used in a GET function is defined as follows:

- `GET("SQS", " -AK AKSOMEACCESSKEY -SK SkSomeSecretKey/dbLo -R US_EAST_1 -Q queue-for-test A", inputdata)`

Since the **-A/-ATTRIBUTES** command is not present, the adapter only text message from the queue.

- Assume the GET map function defined as follows:

```
GET("SQS", " -AK AKSOMEACCESSKEY -SK SkSomeSecretKey/dbLo -R US_EAST_1 -Q queue-for-test -A", inputdata)
```

- In this case the adapter retrieves message attributes along with the text message and the output data defined as follows:

```
{"messages": "It's get example", "attributes": [{"name": "attribute1", "type": "String", "value": "attribute example"}]}
```

Examples of the PUT map function

In the following example, assume that the adapter being used in a PUT function defined as follows:

- Example of create queue property use in PUT map function:

```
PUT("SQS", " -AK AKSOMEACCESSKEY -SK SkSomeSecretKey/dbLo -R US_EAST_1 -Q queue-for-test", inputdata)
```

In this case if the 'queue-for-test' does not exist in specified region, the adapter will show error, since the **-CQ/-CREATEQUEUE** is not present.

- Assume the PUT map function defined as follows:

```
PUT("SQS", " -AK AKSOMEACCESSKEY -SK SkSomeSecretKey/dbLo -R US_EAST_1 -Q queue-for-test -CQ", inputdata)
```

In this case the adapter creates the queue and send message to the newly created queue.

- Example of attribute property use in PUT map function:

```
PUT("SQS", " -AK AKSOMEACCESSKEY -SK SkSomeSecretKey/dbLo -R US_EAST_1 -Q queue-for-test", inputdata)
```

Since the **-A/-ATTRIBUTES** command is not present, input data is treated as message body to be sent over SQS queue

- Assume the PUT map function defined as follows:

```
PUT("SQS", " -AK AKSOMEACCESSKEY -SK SkSomeSecretKey/dbLo -R US_EAST_1 -Q queue-for-test -A", inputdata)
```

- In this case the adapter assumes the input data is passed in required JSON format; input data defined as follows:

```
{
  "message": "It's put example",
  "attributes": [
    {"name": "attribute1", "type": "String", "value": "attribute example"}
  ]
}
```

Amazon SNS Adapter

With the Amazon SNS adapter, a user can access the Amazon Simple Notification Services. This adapter connects to the Simple Notification Services (SNS) topics, using AWS keys. The adapter can fetch and send notifications from and to those AWS SNS topics, to which the user has subscribed.

This document provides an overview and the functional specifications of the Amazon SNS Adapter. The Amazon Simple Notification Service is a notification service provider that can deliver messages in mass, predominantly to mobile users. The Amazon SNS allows for creating topics. A topic is a logical communication-channel to launch mass delivery of a particular category of message. A topic supports high-throughput, push-based, and many-to-many messaging. The function of an SNS Adapter is to access the SNS topics the user has subscribed to for sending and fetching notifications to and from those topics. The SNS topics reside in various geographic regions as served by the Amazon Web Services (AWS). The AWS denotes every geographic region by a specific region code. The same code must be used—in the exact format—in the adapter properties for specifying any region. For more information on AWS SNS, refer <https://aws.amazon.com/sns/>.

The Amazon SNS adapter uses the PUT functionality to push notifications to a specified topic as a target in an output card. An SNS adapter cannot be used as a source in an input card. The notifications can be only in text format. The Amazon SNS adapter does not support the predefined-schema-import option.

An AWS Identity and Access Management (IAM) user can be created in AWS for accessing the SNS services. This user represents a person or application that will interact with AWS. It is not the same as the AWS account root user. For more information, refer: https://docs.aws.amazon.com/IAM/latest/UserGuide/id_users.html

Adapter properties and commands

This section provides an overview of the adapter properties for defining connections and actions and importing schemas in the new web UI in HIP. The following table lists all the properties, their scope, whether they are required or not, and the adapter commands they map to for use in the compiled maps and map executions. "DI" scope means the property is dynamically enumerated - the adapter is invoked during action configuration to provide the set of allowed values for that property. A detailed description of the properties is provided after the table.

Table 1. SNS Adapter properties

Adapter property	Scope	Required	Corresponding adapter command must be used directly in GET and PUT functions
	<ul style="list-style-type: none">• CO: Connection• DI: Discovery• IC: Input card• OC: Output card• G: GET function• P: Put function		
accessKey= access_key	CO	Yes	-AK access_key
secretKey= secret_key	CO	Yes	-SK secret_key
region= region_name	CO	Yes	-R region_name
topicArn= topic_arn	DI/OC/P	Yes	-TARN topic_arn
create Topic= create_topic	OC/P	No	-CT
logging=info errors verbose off	OC/P	No	-T[E V]?[+]? [file_name]
append_log=true false	IC/OC/G/P	No	-T[E V]?[+]? [file_name]
log_file_name=file_name	OC/P	No	-T[E V]?[+]? [file_name]

accessKey= access_key

This property specifies the access key of AWS user for the connection. The access key is used to sign programmatic requests to AWS API calls through the adapter. The access key can be used with **-ACCESSKEY** or **-AK** command.

secretKey= secret_key

This property specifies the secret key of AWS user for the connection. Like the user name and password, both the access key ID and secret access key are mandatory to authenticate SNS topic PUT requests. The secret key can be passed to the adapter with **-SECRETKEY** or **-SK** command.

region= region_name

The adapter uses the region name to connect to an AWS region for the IAM user account. A region can be passed to the adapter using **-REGION** or **-R** command.

topicArn= topic_arn

Topic Amazon Resource Name (ARN) format is used to identify any topic in AWS. This property specifies the topic ARN where notifications are stored. During action configuration, the SNS Adapter provides options to select topic ARN among available topics in that region. Also, the topic_arn property can be passed to adapter input or output cards using the **-TOPICARN** or **-TARN** commands.

For creating SNS topic, refer the link: <https://docs.aws.amazon.com/gettingstarted/latest/deploy/creating-an-sns-topic.html>.

createTopic= create_topic

If this property is specified, the SNS adapter is indicated to create a topic with the name passed in the 'topic_arn' property, if it doesn't already exist. This property is optional. When the create_topic property is not specified, adapter checks if the topic already exists; if not it throws error. Create topic option can be passed to adapter input or output cards using **-CREATETOPIC** or **-CT** commands.

logging=info|errors|verbose|off

This property specifies the level of logging to be used for the log (trace) file produced by the adapter. The default is off. The value info means log informational and error messages, the value error means log error messages only, and the value verbose means log debug and trace level messages along with the informational and error messages.

append_log=true|false

This property flags to indicate the action to be taken when the specified log file already exists. When set to true, the log messages are appended to the file. When set to false: the file is truncated, and the messages are written to the empty file. The default value is true.

log_file_name=file_name

This is the name of the log file, where the log messages are to be written. If not specified, the default log file name m4s3.trc is used, and the file is stored to the directory in which the executed compiled map resides.

Examples

Examples of PUT map function (GET not supported in SNS Adapter)

In the following example, assume that the adapter is used in a PUT function defined as follows:

```
PUT("SNS", "-AK AKSOMEACCESSKEY -SK SkSomeSecretKey/dBLo -R US_WEST_2 -TARN arn:aws:sns:us-west-2:123456789012:gsg-signup-notifications",  
inputdata) In this case if the topic 'gsg-signup-notifications' doesn't exist in specified region, adapter will throw error since -CREATETOPIC/-CT is not present.
```

Now, assume the PUT map function was defined as follows:

```
PUT("SNS", "-AK AKSOMEACCESSKEY -SK SkSomeSecretKey/dBLo -R US_WEST_2 -TARN arn:aws:sns:us-west-2:123456789012:gsg-signup-notifications -  
CT",inputdata)
```

In this case, the adapter creates the topic ARN, and then pushes notification to the newly created topic.

Azure Service Bus Adapter

The Azure Service Bus adapter supports accessing queue on the Microsoft Azure Service Bus, a fully managed message broker service on the Microsoft Azure cloud.

- [Introduction](#)
- [Defining Connections](#)
- [Adapter Properties](#)
This section lists the properties supported by the adapter.
- [Examples](#)

Introduction

The Azure Service Bus adapter is used to send and receive messages from the Azure Service Bus queues. The messages can be JSON, XML, CSV messages, or of any other format, which must be captured in the schema specified when defining the adapter inputs and outputs.

Defining Connections

The adapter provides the Endpoint property for defining the connection string for the queue entity to access. The full URL for the entity endpoint can be specified, or a partial endpoint can be specified indicating for example only the namespace of the queue with additional attributes for the connection string, such as shared access key and queue name. Refer to the Endpoint property description for more information.

- [Azure Service Bus adapter as a source](#)
When the adapter is used as a source in a map, it receives messages from the specified queue. To act as a data source in a map, the adapter can be utilized as an input card in the map or it can be specified in the GET map function. The number of messages to retrieve and the maximum time to wait for new message to arrive can be specified. When either of the conditions is met, the adapter stops receiving and reporting new messages.
- [Azure Service Bus adapter as a target](#)

Azure Service Bus adapter as a source

When the adapter is used as a source in a map, it receives messages from the specified queue. To act as a data source in a map, the adapter can be utilized as an input card in the map or it can be specified in the GET map function. The number of messages to retrieve and the maximum time to wait for new message to arrive can be specified. When either of the conditions is met, the adapter stops receiving and reporting new messages.

The adapter supports two modes for receiving messages:

- Browse: To browse messages without deleting them.
- Peek Lock: To lock the messages and delete them from the queue at the successful completion of the transaction.
- Receive and Delete: To delete the messages immediately from the queue at the time of their retrieval. The adapter supports the use of message sessions for receiving the messages from the session-aware queues.

Azure Service Bus adapter as a target

When the adapter is used as a target, it sends messages to the specified queue. To act as a data target in a map, the adapter can be utilized as an output card in the map or it can be specified in the PUT map function. The adapter supports setting content type on the messages. It also supports setting the message session for sending the messages to the session-aware queues.

Adapter Properties

This section lists the properties supported by the adapter.

- [**Endpoint**](#)
Specifies the connection string URL of the endpoint for the adapter to access.
 - [**Shared Access Key Name**](#)
Specifies the name of the shared access key to use for the authentication, for example **RootManageSharedAccessKey**.
 - [**Shared Access Key**](#)
Specifies the value of the key to use for the authentication.
 - [**Queue Name**](#)
Specifies the name of the queue entity to access. When a value is specified for this property, the **;EntityPath=value** segment is automatically appended to the Endpoint property value when constructing connection string for the queue endpoint.
 - [**Content Type**](#)
Specifies the content type for the messages sent to the queue.
 - [**Session Identifier**](#)
When used in a source mode, it specifies the identifier of the message session to use for the receiving messages from a session-aware queue. When used in target mode, it specifies the message session identifier to set on the messages sent to the session-aware queue.
 - [**Receive Mode**](#)
Specifies the mode to use for receiving messages.
 - [**Message Limit**](#)
Specifies the maximum number of messages to receive from the queue.
 - [**Message Timeout**](#)
Specifies the time in seconds to wait for a new message to arrive after receiving the last message.
 - [**Logging**](#)
Specifies the level of logging to use for the log (trace) file produced by the adapter.
 - [**Append Log**](#)
 - [**Log File Path**](#)
-

Endpoint

Specifies the connection string URL of the endpoint for the adapter to access.

The value is in format `sb://namespace_name/;key=value;key=value...` where `namespace_name` is the fully qualified name of the namespace containing the queues to access. The `sb://namespace_name/` portion is required and the `key=value` pairs are optional. The specified connection string URL may be a partial connection string URL for the endpoint to access. For example, when a value is provided for the Queue adapter property, the **;EntityPath=value** segment is automatically appended to the specified Endpoint property when constructing the full connection string of the queue entity endpoint. The corresponding adapter command for this property is **-E url**.

Shared Access Key Name

Specifies the name of the shared access key to use for the authentication, for example **RootManageSharedAccessKey**.

When a **value** is specified for this property, the **;SharedAccessKeyName=value** segment is automatically appended to the specified Endpoint property value. When a value is not provided, the Endpoint property is assumed to contain all the necessary authentication details. The corresponding adapter command for this property is **-SAKN name**.

Shared Access Key

Specifies the value of the key to use for the authentication.

When a **value** is specified for this property, the **;SharedAccessKey=value** segment is automatically appended to the specified Endpoint property value. When a value is not provided, the value provided in the Endpoint property is assumed to contain all the necessary authentication details. The corresponding adapter command for this property is **-SAK key**.

Queue Name

Specifies the name of the queue entity to access. When a value is specified for this property, the **;EntityPath=value** segment is automatically appended to the Endpoint property value when constructing connection string for the queue endpoint.

If the value is not provided, the queue name is assumed to be included in the Endpoint property. The corresponding adapter command for this property is **-QN queue**.

Content Type

Specifies the content type for the messages sent to the queue.

When provided, the value must be in the Content-Type header format as defined in the RFC 2045 specification. The corresponding adapter command is **-CT type**.

Session Identifier

When used in a source mode, it specifies the identifier of the message session to use for the receiving messages from a session-aware queue. When used in target mode, it specifies the message session identifier to set on the messages sent to the session-aware queue.

Refer to the Azure Service Bus documentation for more information about the message sessions. The corresponding adapter command is **-SID id**.

Receive Mode

Specifies the mode to use for receiving messages.

There are three modes:

- Browse: The adapter only browses the message on the queue without deleting it.
- Peek Lock: The adapter does not delete the message from the queue immediately upon reading it. Instead, a lock is created for the message which the adapter completes at end of the transaction, as defined by the transaction scope for the adapter. The completion of the lock in turn results in deleting the message from the queue.
- Receive and Delete: The message is deleted immediately from the queue at the time it is received.

The corresponding adapter command for this property is **-RM**, where mode is one of browse, peek_lock or receive_and_delete (case-insensitive). The default value is browse.

Message Limit

Specifies the maximum number of messages to receive from the queue.

The default value is 1. The special value S is used to indicate no limit (all available messages). The corresponding adapter command is **-QTY limit**.

Message Timeout

Specifies the time in seconds to wait for a new message to arrive after receiving the last message.

The timeout countdown is restarted after each successfully retrieved message. The default value is S, which is a special value to indicate no timeout (indefinite waiting). Another special value is 0 which is used to indicate no waiting, that means once all currently available messages are received, the adapter does not wait for any more messages to arrive. The corresponding adapter command is **-LSN timeout**.

Logging

Specifies the level of logging to use for the log (trace) file produced by the adapter.

The default is Off. The value Information means log informational, the value Errors Only means log error messages only, and the value Verbose means log debug and trace level messages along with the informational and error messages.

The corresponding adapter command for informational level logging is **-T** or **-TRACE path**, where *path* is the optional log file path.

For error level logging, the **-TE** (or **-TRACEERROR**) command should be used instead of **-T**. For verbose level logging, the **-TV** (or **-TRACEVERBOSE**) command is used.

For the append mode, the + character should be added to the command name. For example, to append log messages to the **/tmp/logfile.log** file in verbose mode, the respective adapter command would be: **-TV+ /tmp/logfile.log**

Append Log

The flag indicates what to do if the specified log file already exists. When enabled, the log messages are appended to the file. When disabled, the file is truncated, and the messages are written to the empty file. The property is disabled by default.

Log File Path

Specifies the location of the log file to which to write log messages. This is the location of the log file to which to write log messages. If not specified, the default log file name m4azsb.mtr is used, and the file is stored to the directory in which the executed compiled map resides.

Examples

GET examples

```
GET("AZSB", "-E sb://namespace_name.servicebus.windows.net/ -SAKN RootManageSharedAccessKey -SAK key_value -Q queue_name -RM peek_lock -T /tmp/logfile.log")
```

PUT example

```
PUT("AZSB", "-E sb://namespace_name.servicebus.windows.net/ -SAKN RootManageSharedAccessKey -SAK key_value -Q queue_name -CT application/json -TV+ /tmp/logfile.log")
```

Azure SQL Adapter

Microsoft Azure SQL adapter provides support for performing read, write and discovery operations on Microsoft Azure SQL Database.

- [Introduction](#)
Microsoft Azure SQL Database is a general-purpose relational database, running as a managed service on the Microsoft Azure cloud computing platform
 - [Azure SQL Adapter commands](#)
This chapter provides a detailed description of the adapter.
-

Introduction

Microsoft Azure SQL Database is a general-purpose relational database, running as a managed service on the Microsoft Azure cloud computing platform

Authenticating connections

The adapter supports username/password based authentication method.

Microsoft Azure SQL adapter as a source

Microsoft Azure SQL adapter as a source supports:

- Fetching data from tables and views.
- Executing stored procedures and consuming results.
- Executing query statements and fetching results.

Microsoft Azure SQL adapter as a target

Microsoft Azure SQL adapter as a target supports:

- Inserting and updating rows in tables.
- Executing parameterized stored procedures.
- Executing parameterized query statements.
- Executing custom data manipulation language (DML) statements.

Azure SQL Adapter commands

This chapter provides a detailed description of the adapter.

- [Server Name](#)
- [Port](#)
- [Database](#)
- [User](#)
- [Password](#)
- [Encrypt](#)
- [Trust Server Certificate](#)
- [Hostname in Certificate](#)
- [Login Timeout](#)
- [Autocommit](#)
- [Table](#)
- [Query](#)
- [Procedure](#)
- [DML](#)
- [Schema](#)
- [Logging](#)
- [Append Log](#)
- [Log File Path](#)
- [Pre SQL](#)
- [Post SQL](#)

- [Write Mode](#)
 - [-QTY and -LSN adapter commands](#)
 - [-ARRAYSIZE adapter command](#)
 - [Examples](#)
-

Server Name

Specifies the address of the server to connect to. This is a mandatory property. The corresponding adapter command is -SN (or -SERVERNAME).

For example:

```
-SN azureserver.database.windows.net
```

Port

Specifies the port on which the server is listening for requests. The default is 1433. This is a mandatory property. The corresponding adapter command is -PORT.

Database

Specifies the name of the database instance on the server. This is a mandatory property. The corresponding adapter command is -DB (or -DATABASE).

For example:

```
-DB azuredb
```

User

Specifies the username for the connection. The corresponding adapter command is -USR (or -USER).

Password

Specifies the password for the connection. The corresponding adapter command is -PWD (or -PASSWORD).

Encrypt

Enforce the use of Secure Sockets Layer (SSL) on the connection. Default is true. This is an advanced property. The corresponding adapter command is -ENC (or -ENCRYPT).

Trust Server Certificate

Enables automatic trust in the SSL certificate presented by the database service. Default is false. This is an advanced property. The corresponding adapter command is -TSE (or -TRUSTSERVCERT).

Hostname in Certificate

Specifies the name of the host expected in the SSL certificate presented by the database service. This is an advanced property. The corresponding adapter command is -HNC (or - HOSTNAMECERT).

Login Timeout

Specifies the number of seconds to wait before timing out when establishing connection. The default is 30 seconds. The corresponding adapter command is -LT (or -LOGINTIMEOUT).

Autocommit

Instructs the adapter how to configure the driver in respect to automatic commit of SQL statements performed by the driver. Default is false. When set to true, the adapter explicitly requests the driver to enable and perform auto-commit, and when set to false the adapter explicitly requests the driver to disable and not perform auto-commit.

The corresponding adapter command is -AC (or -AUTOCOMMIT).

Table

Specifies the name of the table or view to access. The corresponding adapter command is -TBL (-TABLE).

Query

Specifies the text of the query statement to execute when fetching data. When GET map function is used with the adapter, the query statement text may contain query bind parameters (as '?' characters), and the values for the parameters are provided as the third argument of the GET function, with '|' character used as the value separator. The corresponding adapter command is -QRY (-QUERY).

Procedure

Specifies the name of the stored procedure to execute. When GET map function is used with the adapter, the stored procedure can have input and input/output parameters, and the values for those parameters are provided as the third argument of the GET function, with '|' character used as the value separator. The corresponding adapter command is -PROC (-PROCEDURE).

DML

Custom DML statement to use for writing rows to the database. When specified, the schema, table and write mode properties are ignored. The number of bind parameters in the statement must match the number of fields in the records pushed to the adapter. The corresponding adapter command is -DML.

Schema

Specifies the name of the database schema. It is case-sensitive. The adapter will automatically surround it with the quoted identifiers, so the specified value must not be quoted manually by the user. The corresponding adapter command is -SCH (-SCHEMA).

Logging

This property specifies the level of logging to use for the log (trace) file produced by the adapter. The default is Off. The value Information means log informational, the value Errors Only means log error messages only, and the value Verbose means log debug and trace level messages along with the informational and error messages.

Append Log

Flag indicating what to do if the specified log file already exists. When set to true, the log messages are appended to the file. When set to false, the file is truncated, and the messages are written to the empty file. The default value is true.

Log File Path

Specifies the location of the log file to which to write log messages. If not specified, the default log file name m4azsql.mtr is used, and the file is stored to the directory in which the executed compiled map resides.

Pre SQL

Specifies one or more SQL statements to execute after connecting to the database and prior to fetching any rows from the database or writing any rows to the database. The statements are committed by the adapter after the last one is executed, or automatically if autocommit is specified or the statements are non-transactional (e.g. DDL statements). Bind parameters are not supported – the statements must be static statements. The adapter assumes that individual statements are separated by a semicolon. Each semicolon and backslash character that is integral part of a statement needs to be escaped by a backslash character. The corresponding adapter command is -PRESQL.

Post SQL

The **T**SPECIES one or more SQL statement to execute after fetching or writing all rows and prior to disconnecting from the database. The statements are committed by the adapter after the last one is executed, or automatically if autocommit is specified or the statements are non-transactional (e.g. DDL statements). Bind parameters are not supported – the statements must be static statements. The adapter assumes that individual statements are separated by a semicolon. Each semicolon and backslash character that is integral part of a statement needs to be escaped by a backslash character. The corresponding adapter command is -POSTSQL.

Write Mode

Specifies the mode of operation to use in PUT map function and output card when the table property is specified. The default value is insert. The meaning of the values is as follows:

- Insert – Each input row is inserted to the target table.
- Update – Each input row is used to update the corresponding row in the target table.

The corresponding adapter command is -RM (or -WRITEMODE).

-QTY and -LSN adapter commands

The -QTY adapter command is used when fetching data from the database and it specifies the maximum number of rows to fetch from the database. The default value is **1**, and the values **S** means all available rows. The -QTY command must always be accompanied by the -LSN 0 command. The -QTY and -LSN commands are managed automatically when the adapter is utilized in input card in the Design Server web UI. But when the adapter command line is specified directly, the -QTY and -LSN commands must be included in it manually.

When using an API to create or overwrite an input Azure SQL card:

- When Fetch As is set to Integral mode, set the Fetch Unit to 1 and specify the following commands so that adapter reads and returns all available rows at one time:
 - -QTY 1
 - -LSN 0
 - -ARRAYSIZE 0
- When Fetch As is set to Burst mode, set the Fetch Unit to 1 and specify the following commands so that adapter reads and returns one row per burst:
 - -QTY S
 - -LSN 0

-ARRAYSIZE adapter command

Specifies the number of table rows the adapter concatenates and returns as a single logical record. When concatenating the rows, the adapter uses linefeed character as the row terminator and the pipe character as the field delimiter. The special value 0 means that all available rows should be concatenated and returned as a single logical record. The -ARRAYSIZE adapter command is managed automatically when the adapter is used in input card in the Design Server web UI

Examples

GET function example for fetching data from a table

```
GET("AZSQL", "-SN servername -DB databasename -PORT portname -USR username -PWD password -SCH schemaname -TBL tablename -QTY 1 -LSN 0 -AS 0")
```

PUT function example for inserting data into a table

```
PUT("AZSQL", "-SN servername -DB databasename -PORT portname -USR username -PWD password -SCH schemaname -TBL tablename -WM insert")
```

Azure Blob Storage Adapter

The Azure Blob Storage adapter provides support to perform read and write operations on the blobs in the block storage managed by the Azure Blob Storage cloud service.

- [Introduction](#)

The Azure Blob Storage adapter enables data integration with the Azure Blob Storage service, through its ability to retrieve data from blobs, and store new data to blobs managed by this service. For more information about the Azure Blob Storage service, refer to the Azure Blob Storage documentation.

- [Adapter Properties and Commands](#)

This section lists the properties supported by the adapter.

- [Examples](#)

Introduction

The Azure Blob Storage adapter enables data integration with the Azure Blob Storage service, through its ability to retrieve data from blobs, and store new data to blobs managed by this service. For more information about the Azure Blob Storage service, refer to the Azure Blob Storage documentation.

Azure Blob Storage adapter as a source

The Azure Blob Storage adapter as a source can be used to read blob data, with the option to delete the blob or to preserve it after reading the data from it.

Azure Blob Storage adapter as a target

The Azure Blob Storage adapter as a target can be used to perform the following operations;

- Create a container
- Create a new blob and store data in it
- Overwrite an existing blob with the provided data
- Append data to an existing blob

Adapter Properties and Commands

This section lists the properties supported by the adapter.

- [Account Name](#)
- [Access Key](#)
- [Endpoint Suffix](#)
- [Endpoints Protocol](#)
- [Container Name](#)
- [Write Mode](#)
- [Block Size](#)
Specifies the size, in kilobytes, of blocks when writing the blob.
- [Read Mode](#)
- [Overwrite Blob](#)
- [Blob Name](#)
- [Logging](#)
- [Log File Path](#)
- [Convert Data](#)
- [Data Format](#)
- [Header](#)
- [Quote](#)
- [Delimiter](#)
- [Escape](#)
- [Separator](#)
- [Character Set](#)
- [Limit](#)
- [Sample Size](#)
- [Worksheet Index](#)
- [Select Source Tables](#)
- [Find Array](#)
- [Array Path](#)

Account Name

Specifies a storage account name on Azure Blob Storage. This is a mandatory property. The corresponding adapter command is -AN *accountName* (or -ACCOUNTNAME *accountName*).

Access Key

Specifies access key to connect to Azure Blob Storage account. The corresponding adapter command is -AK *accessKey* (or -ACCESSKEY *accessKey*).

Endpoint Suffix

Specifies the endpoint suffix to use for establishing the connection. The default value is core.windows.net. The corresponding adapter command is -ES *suffix* (or -ENDPOINTSUFFIX *suffix*).

Endpoints Protocol

Specifies a protocol that is used to make a connection. It has two possible values: HTTP and HTTPS. The corresponding adapter command is -EP *protocol* (or -ENDPOINTSPROTOCOL *protocol*) where *protocol* is one of http or https.

Container Name

Specifies the name of the container for the blobs on which the actions need to be performed. The name must be specified as a lowercase value. The corresponding adapter command is -CN *containerName* (or -CONTAINERNAME *containerName*).

Write Mode

Specifies the mode to use for the blob write operation. The default value is Block Blob. The two supported modes are:

- Block Blob: : To perform write operation on the blob of Block type
- Append Blob: : To perform write operations on the blob of Append type

The corresponding adapter command is -WM *mode* (or -WRITEMODE *mode*) where *mode* is one of block_blob or append_blob.

Block Size

Specifies the size, in kilobytes, of blocks when writing the blob.

The default value is 4096. The corresponding adapter command is -BS *blockSize* (or -BLOCKSIZE *blockSize*)

Read Mode

Specifies whether to delete or keep the blob after reading it. Default is Keep. There are two values:

- Keep: To keep a Blob.
- Delete: To delete a Blob.

The corresponding adapter command is -RM *mode* or (-READMODE *mode*), where *mode* is one of Keep or Delete.

Overwrite Blob

Specifies whether to overwrite a new Block Blob over the existing Block Blob available in the container. Default is true. If this property is set to false, the adapter gives notification to user to overwrite the Blob. The corresponding adapter command is -O (or -OVERWRITE).

Blob Name

Specifies the name of Blob. The corresponding adapter command is -B *blobName* (or -BLOBNAME *blobName*).

Logging

This property specifies the level of logging to use for the log (trace) file produced by the adapter.

The default is Off. The value Information means log informational, the value Errors Only means log error messages only, and the value Verbose means log debug and trace level messages along with the informational and error messages.

Logging can be specified using command:

-T [E|V] [+][file_path]

-T -> Log adapter informational messages.

-TE -> Log only adapter errors.

-TV -> Use verbose (debug) logging. The log file records all activity that occurs while the adapter is producing or consuming messages.

+ -> Appends the trace information to the existing log file. Omit this keyword to create a new log file.

file_path -> The full path to the adapter trace log. If you omit this keyword, the adapter creates the m4azblob.mtr log file in the map directory.

Log File Path

Specifies the location of the log file to which to write log messages. If not specified, the default log file name m4azblob.mtr is used, and the file is stored to the directory in which the executed compiled map resides

Convert Data

Specifies to convert data to parsed form. If enabled, support for converting CSV, Avro, Excel, Parquet and JSON documents into sets of data records in the CSV format. This means that the data produced by the converters can be loaded directly into relational databases.

Data Format

Specifies the data format. This is a mandatory property. Select one of the following as per the data format type. This property field is applicable when Convert Data property toggle is enabled.

- CSV
 - Excel
 - Avro
 - Parquet
 - JSON
-

Header

If enabled, specifies whether the CSV data contains a header row.

Quote

Specifies the quote character. Default value is ".

Delimiter

Specifies the field delimiter character. Default value is ,.

Escape

Specifies the escape character.

Separator

Specifies the row separator characters. Default value is \r\n.

Character Set

Specifies the character set of the data. Default value is UTF-8.

Limit

Specifies the maximum number of records to return.

Sample Size

Specifies the number of records to analyze when determining the data structure. Default value is 100.

Worksheet Index

Specifies the index of the worksheet. The index of the first worksheet is 1 (default).

Select Source Tables

Select which arrays of objects in the AVRO data you wish to map to tables in the target.

Find Array

If enabled, specifies that the output will iterate on the first array found in the JSON document.

Array Path

Specifies the path of the JSON array within the document. Omit if the document is an array. For example, "/resources".

Examples

GET example

```
GET("AZBLOB", "-AN accountName -AK accessKey -ES core.windows.net -EP https -CN containerName -RM keep -B blobName")
```

PUT example

```
PUT("AZBLOB", "-AN accountName -AK accessKey -ES core.windows.net -EP https -CN containerName -WM block_blob -O -B blobName -BS 4096")
```

B2B Advanced Communications Storage Adapter

The B2B Advanced Communications Storage adapter gives IBM® Transformation Extender the ability to read and write payloads that were stored in the IBM Sterling B2B Integrator storage subsystem.

Overview

Table 1. B2B Advanced Communications Storage adapter overview. The following table provides an overview of the B2B Advanced Communications Storage adapter:

Characteristic	Definition
Adapter Name	B2B Advanced Communications Storage Adapter specifics are as follows: <pre><M4Adapter name="IBM B2B Storage" alias="BSTR" id="_home_markdown_jenkins_workspace_Transform_in_SSVSD8_11.0.1_com.ibm.websphere.dtx.adapb2bstorage.doc_b2b_storage_adapter_194" type="app" class="com.ibm.itx/dtx/m4storage"/></pre>
Description	This adapter integrates IBM Transformation Extender with IBM B2B Advanced Communications. This adapter enables communication between the two products and allows IBM Transformation Extender to get data from and put information to B2B Advanced Communications storage.

Characteristic	Definition
Business usage	<p>IBM Transformation Extender uses this adapter to retrieve data from B2B Advanced Communications storage and persist data to B2B Advanced Communications storage.</p> <p>Inbound The inbound integration steps are as follows:</p> <ol style="list-style-type: none"> 1. B2B Advanced Communications sends a business document object to IBM MQ. 2. The IBM Transformation Extender reads the business document object from IBM MQ using the IBM MQ adapter. 3. The IBM Transformation Extender map uses a B2B Advanced Communications schema on the input card to read and parse the business document object. 4. The IBM Transformation Extender map reads the B2B Advanced Communications storage of the input document (if the data is not sent inline) <p>Outbound The outbound integration steps are as follows:</p> <ol style="list-style-type: none"> 1. IBM Transformation Extender adds the message to B2B Advanced Communications storage. 2. The IBM Transformation Extender map creates a business document object. 3. The IBM Transformation Extender map writes the message to IBM MQ.
Preconfigured?	<p>No. The storage-required .jar files and business document object schemas are installed with IBM Transformation Extender. However, you must perform the following steps before using the adapter:</p> <ol style="list-style-type: none"> 1. Create a folder called com/ibm/itx/dtx/m4storage under the tx_install_dir/Jars folder. 2. Copy the following .jar files under the com/ibm/itx/dtx/m4storage folder: <ul style="list-style-type: none"> • m4storage.jar • ibm-b2b-meg-core-apiint-common*.jar • ibm-b2b-meg-core-apiint-common-api*.jar • ibm-b2b-meg-core-components-utils*.jar • ibm-b2b-meg-core-storage-core*.jar • ibm-b2b-meg-core-storage-fs*.jar • ibm-b2b-meg-core-storage-service*.jar 3. Create a Schemas folder under the com/ibm/itx/dtx/m4storage folder. 4. Copy the business document object related schemas into the new Schemas folder: <ul style="list-style-type: none"> • BaseMessages.xsd • BaseTypeDefinitions.xsd • BusinessDocument.xsd • ExchangeVisibilityEvent.xsd • NotificationMessage.xsd 5. Add the following two properties to the tx_install_dir/config.yaml file: <pre style="background-color: #f0f0f0; padding: 5px;">runtime: B2B STORAGE: BASE_FILE_PATH: location_of_root_storage DATA_DOMAIN: data_domain_for_B2B_Advanced_Communications_storage</pre>
Requires third-party files?	No. All required components are shipped with the product.
Platform availability	<ul style="list-style-type: none"> • Windows • Linux®
Related services	None
Application requirements	This adapter requires that you install B2B Advanced Communications version 1.0.0.5 or higher. If you do not install the full B2B Advanced Communications product, at a minimum you must deploy the Proof of Concept installation. Additionally, you must install IBM Transformation Extender V9.0.0.1 or later.
Initiates business processes?	No
Invocation	This adapter can be invoked through the command line.
Business process context considerations	None
Returned status values	None
Restrictions	<ul style="list-style-type: none"> • Data ID, which is provided when data is written to B2B Advanced Communications storage, is required for inbound process flows to read the data from storage. • PUT is not supported for this function.
Persistence level	None
Testing considerations	<p>There are two levels of testing, depending on whether you installed the B2B Advanced Communications Proof of Concept installation or the full version of the product:</p> <ol style="list-style-type: none"> 1. Install B2B Advanced Communications (Proof of Concept) version 1.0.0.5 or higher and perform storage adapter testing. 2. Install B2B Advanced Communications (production) version 1.0.0.5 or higher and test the Launcher with the MQ adapter.

Command line options

Use the following command line options with this adapter:

-BUCKET (-B)

- The name of the bucket.
- ID (-ID)
This is the unique identifier for the message or business document object.
- NOW (-NOW)
Indicates that the adapter is writing data to B2B Advanced Communications storage.
- SERVICE (-S)
This is the name of the adapter.
- ADDDOC (-A)
Adds a business document object to B2B Advanced Communications storage.

Implementing the B2B Advanced Communications Storage adapter

To implement this adapter, follow this procedure:

1. From the command line, go to *B2B_Advanced_Communications_install_dir/members/bin*
2. Invoke `execute member start all`
3. Invoke `execute storage -provision`

Inbound parameters passed from business process to adapter

Field	Description
Action	<p>The inbound storage action.</p> <ul style="list-style-type: none"> • GET - get payload from storage as primary document <p>For example:</p> <pre>=VALID(GET("BSTR", "-B 1st_provisioned -S 1st_provisioned_var01 -ID " + Output2), LASTERRMSG())</pre>
DataId	The unique identifier for the business document object to be retrieved from B2B Advanced Communications storage.

Outbound parameters passed from adapter to business process schema

Field	Description
Action	<p>The outbound storage action.</p> <ul style="list-style-type: none"> • GET - write payload from message to B2B Advanced Communications storage <p>For example:</p> <pre>=VALID(GET("BSTR", "-B 1st_provisioned -S 1st_provisioned_var01 -NOW -ADDDOC", "THIS IS A TEST"), LASTERRMSG())</pre>
DataId	The unique identifier for the business document object to be added to B2B Advanced Communications storage. This value is provided by the adapter.
PrimaryDocument	The business document object. This value is returned by the GET action from B2B Advanced Communications storage.

Batch File and Shell Script adapters overview

The Batch File adapter processes a (Windows) batch file as the data source of an input map card or the data target of an output map card. The Shell Script adapter processes a (UNIX) shell script as the data source of an input map card or the data target of an output map card.

You can achieve seamless data flow between applications regardless of format by combining the use of batch files or shell scripts to control where data is transferred.

Use the adapters with the Command Server, Launcher, Software Development Kit, or in a rule within a map.

- [System requirements](#)
- [Command aliases](#)
- [Commands for the Batch File and Shell Script adapters](#)
- [Syntax summary for Batch File and Shell Script adapters](#)
- [Return codes and error messages](#)

System requirements

See the IBM Transformation Extender [system requirements](#) for details. It is assumed that a Command Server has already been installed on the computer where the adapter is to be installed for runtime purposes.

Command aliases

Adapter commands can be specified by using a command string on the command line or by creating a command file that contains adapter commands. The execution command syntax is:

```
-IA[alias] card_num
-OA[alias] card_num
```

where `-IA` is the Input Source Override execution command and `-OA` is the Output Target Override execution command, `alias` is the adapter alias, and `card_num` is the number of the input or output card. The following table shows each adapter alias and its execution commands.

Adapter	Alias	As Input	As Output
Batch File	BAT	<code>-IABATcard_num</code>	<code>-OABATcard_num</code>
Shell Script	SHL	<code>-IASHLcard_num</code>	<code>-OASHLcard_num</code>

Commands for the Batch File and Shell Script adapters

The following table lists valid commands for the Batch File and Shell Script adapters, the command syntax, and whether the command is supported (✓) for use with data sources, data targets, or both.

Command	Syntax	Source	Target
Audit (-A or -AUDIT)	<code>-AUDIT[+][S] [full_path]</code>	✓	✓
Command Line Interpreter (-CL or -CLI)	<code>-CLI[! cmd_line_interpreter]</code>	✓	✓
Command File (-C or -CMD)	<code>-CMD batch_file_or_shell_script</code>	✓	✓
Transfer File (-F or -FILE)	<code>-FILE [transfer_file]</code>	✓	✓
Inline Output (-I or -INLINE)	<code>-INLINE</code>		✓
Poll Process (-P or -POLL)	<code>-POLL count:interval</code>	✓	✓
Show Mode (-S or -SHOW) (Batch file adapter only)	<code>-SHOW {HIDE MIN MAX NRM api_mode}</code>	✓	✓
Standard Input (-< or -STDIN)	<code>-STDIN "data"</code>	✓	✓
Trace (-T or -TRACE)	<code>-T[E][+][S V] [full_path]</code>	✓	✓

- [Audit \(-A or -AUDIT\)](#)
- [Command line interpreter \(-CL or -CLI\)](#)
- [Batch files in Windows environments](#)
- [Shell scripts in UNIX environments](#)
- [Command file \(-C or -CMD\)](#)
- [Transfer file \(-F or -FILE\)](#)
- [Inline output \(-I or -INLINE\)](#)
- [Poll process \(-P or -POLL\)](#)
- [Show mode \(-S or -SHOW\)](#)
- [Standard input \(-< or -STDIN\)](#)
- [Trace \(-T or -TRACE\)](#)

Audit (-A or -AUDIT)

Use the Audit adapter command (`-A` or `-AUDIT`) to produce an audit log file in the map directory. Audit information includes the command line, process id, adapter return code, number of bytes read/written, elapsed time of the command execution, number of retries and standard input, output or error messages as appropriate, if pipes are used.

`-AUDIT[+][S] [full_path]`

Option

Description
<code>+</code>
Append audit information to the existing audit log file. If an audit log file does not exist, one is created.
<code>S</code>
Record only summary information in the audit log file.

`full_path`

Create audit file with the specified name in the specified directory. By default, the directory is where the map is located and the file name is **m4batch.log** or **m4shell.log** (as applicable).

Command line interpreter (-CL or -CLI)

Use the Command Line Interpreter adapter command (`-CL` or `-CLI`) to specify the command line interpreter used to execute the batch file or shell script.

`-CLI[!|cmd_line_interpreter]`

Option

Description
<code>!</code>
Use no command line interpreter.

`cmd_line_interpreter`

Specify a command line interpreter.

Batch files in Windows environments

In Windows environments, use the Command Line Interpreter adapter command (`-CLI`) as follows. If `-CLI` is not specified, no command line interpreter is assumed.

Entering this

Specifies this command line interpreter

```
-CLI  
    cmd/c  
-CLI!  
    none  
-CLI command  
    command
```

Note: The result of not specifying the Command Line Interpreter adapter command (`-CL`) is the same as specifying the default of `-CLI!` which assumes no command line interpreter.

Shell scripts in UNIX environments

In UNIX environments, use the Command Line Interpreter adapter command (`-cli`) as follows. If `-cli` is not specified, a command line interpreter of `/bin/sh -c` is assumed.

Entering this

Specifies this command line interpreter

```
-cli  
    /bin/sh -c  
-cli!  
    none  
-cli command  
    command
```

Note: The result of not specifying the Command Line Interpreter adapter command (`-cl`) is the same as specifying the default of `-cli`, which assumes a command line interpreter of `/bin/sh -c`.

Command file (-C or -CMD)

Use the Command File adapter command (`-C` or `-CMD`) to specify a batch file or shell script file to run. This command and the file name are required to execute a batch file or shell script.

```
-CMD batch_file_or_shell_script
```

Option

Description

batch_file_or_shell_script
Specify the batch file or shell script file to run.

Transfer file (-F or -FILE)

By default, data is written through pipes (standard input for a source, standard output for a target). You can use the Transfer File adapter command (`-F` or `-FILE`) to specify a temporary holding file where the adapter can place its output data or obtain its input data. If a file name is not specified, the adapter will automatically generate a file name, which is passed as part of the command line to the batch file or shell script.

For example, `doit.bat` is called as `doit.bat transferfile.tmp`. If the process has not finished within the timeout period as specified with the Poll Process adapter command (`-POLI`), the file is not deleted.

For an input, on process completion, the content of the transfer file is used for the source data for the specified input card.

```
-FILE [transfer_file]
```

Option

Description

transfer_file
Specify the name of the transfer file.
For an input, the file name can include wildcards. If wildcards are used, the contents of the multiple files are concatenated (without delimiters) to become the data for the input card.
For an output, wildcards cannot be used.

Inline output (-I or -INLINE)

By default, data is written through pipes (standard input for a target). You can use the Inline Output adapter command (`-I` or `-INLINE`) to specify the data from an output card to be passed as part of the command line to the batch file or shell script.

```
-INLINE
```

Poll process (-P or -POLL)

Use the Poll Process adapter command (-P or -POLL) to specify the number of times and frequency at which the adapter polls for process completion.

-POLL count:interval

Option

Description

count

The number of times to check for process completion.

interval

The number of seconds to wait between each poll.

If the number of specified polling attempts is exhausted and the process has not yet completed, the adapter will exit with the message Command Still Executing. This is a warning for a target, but an error for a source.

Show mode (-S or -SHOW)

Use the Show Mode adapter command (-S or -SHOW) to specify the mode for displaying the command window. Use this command only with the Batch File adapters. If the Show Mode adapter command (-S or -SHOW) is not specified, the default show mode is HIDE.

-SHOW {HIDE|MIN|MAX|NRM|api_mode}

Option

Description

HIDE

The command window is hidden.

MIN

The command window is minimized.

MAX

The command window is maximized.

NRM

The command window is normal.

api_mode

Specify an integer value corresponding to the flags desired, as documented for the ShowWindow API in the Windows API for values.

Standard input (-< or -STDIN)

Use the Standard Input adapter command (-< or -STDIN) when passing the specified data through pipes as standard input (that is, when -file is omitted) during batch file or shell script processing. This is useful when keyboard response is required during the process.

-STDIN "data"

For example, if a batch file or shell script procedure requires a yes keyboard response three times during the process, specify:

-STDIN "yes yes yes"

Trace (-T or -TRACE)

Use the Trace adapter command (-T or -TRACE) to produce a diagnostics file that contains detailed information about Batch File and Shell Script adapters activity.

The default filename is **m4batch.mtr**, **m4shell.mtr**, or **map_name.mtr**, depending on the option(s) you select. The trace file is created in the map directory unless otherwise specified.

-T[E][+][S|V] [full_path]

Option

Description

E

Produce a trace file containing only the adapter errors that occurred during map execution. The adapter trace file is produced by default in the map directory, using the full name of the map file with an .mtr filename extension (map_name.mtr).

+

Append trace information to the existing trace file.

S

Summary mode. Record only minimal information in the log file. This is the default value.

V

Verbose mode. Record in the log file all activity occurring while the adapter is retrieving data. If not specified, summary mode is assumed.

full_path

Creates a trace file with the specified name in the specified directory.

Note: You can override the adapter command line trace options dynamically using the Management Console. Refer to *Dynamic Adapter Tracing* in the *Launcher Reference Guide* for detailed information.

Note: Because summary mode is the default, using -TRACES provides the same output as -TRACE.

Syntax summary for Batch File and Shell Script adapters

- [Data sources](#)
- [Data targets](#)

Data sources

Use the following data source command syntax for the Batch File and Shell Script adapters:

```
-CMD batch_file_or_shell_script
[-AUDIT[+] [S] [full_path]
[-CLI[!|cmd_line_interpreter]]
[-FILE [transfer_file]]|[-STDIN data]
[-POLL count:interval]
[-SHOW {HIDE|MIN|MAX|NRM|api_mode}]
[-T[E] [+][S|V] [full_path]]
```

Data targets

Use the following data target command syntax for the Batch File and Shell Script adapters:

```
-CMD batch_file_or_shell_script
[-AUDIT[+] [S] [full_path]
[-CLI[!|cmd_line_interpreter]]
[-FILE [transfer_file]]|[-INLINE]
[-POLL count:interval]
[-SHOW {HIDE|MIN|MAX|NRM|api_mode}]
[-T[E] [+][S|V] [full_path]]
```

Return codes and error messages

Return codes and messages are returned when the particular activity completes. Return codes and messages may also be recorded as specified in the audit logs, trace files, execution summary files, and so on.

The following is a listing of all the codes and messages that can be returned as a result of using the Batch File and Shell Script adapters for sources or targets.

Note: Adapter return codes with positive numbers are warning codes that indicate a successful operation. Adapter return codes with negative numbers are error codes that indicate a failed operation.

Return Code	Message
0	OK
-1	Insufficient memory to continue
-1	Library Initialization Failed
-2	Invalid Entry Point (AIX® adapters only)
-3BATCH	Could not load adapter
-3	Error Sending Data
-4	Error Receiving Data
-5	BATCH: Insufficient memory to continue
1	No data provided. Create on content specified: command not executed.
7	Command Still Executing (Applies to adapter target)
-1	Incorrect Arguments
-2	Error Writing Output Files
-2	Error Reading Input Data
-5	Insufficient Memory to Continue
-6	Error Executing Command (Return Code XX) XX is return code from command
-7	Command Still Executing (Applies to adapter source)

CICS Adapter overview

The Customer Information Control System (CICS) adapter allows you to retrieve CICS screen data without the difficulties associated with 3270 "screen scraping." Using the CICS adapter with schemas generated from BMS maps you can easily send data to and retrieve data from CICS transactions.

For more information about CICS transactions see [CICS Transaction Server for OS/390® Version 1.3 External Interfaces Guide \(SC33-1944\)](#).

- [System requirements](#)
- [Command alias](#)
- [CICS Adapter commands](#)
- [Syntax summary](#)
- [Using the CICS adapter](#)

System requirements

The minimum system requirements and operating system requirements for the CICS adapter are detailed in the release notes. It is assumed that a IBM Transformation Extender has already been installed on the computer where the adapter is to be installed for runtime purposes.

In addition, the following requirements are necessary to use the CICS adapter:

- A correctly installed and configured CICS TCP/IP system
- IBM® supplied (TCP/IP Port) Listener
- IBM supplied 3270 Bridge transaction

Command alias

Adapter commands can be specified by using a command string on the command line or creating a command file that contains adapter commands. The execution command syntax is:

```
-IM[alias] card_num
-OM[alias] card_num
```

where **-IM** is the Input Source Override execution command and **-OM** is the Output Target Override execution command, *alias* is the adapter alias, and *card_num* is the number of the input or output card. The following table shows the adapter alias and its execution command.

Adapter	Alias	As Input	As Output
CICS	CICS	-IMCICS <i>card_num</i>	-OMCICS <i>card_num</i>

CICS Adapter commands

The following table lists valid commands for the CICS Adapter, the command syntax, and whether the command is supported (✓) for use with data sources, targets, or both.

Command	Syntax	Source	Target
Bridge Facility Like (-BFL)	-BFL <i>terminal_id</i>	✓	✓
Bridge Keep Time (-BKT)	-BKT <i>time</i>	✓	✓
Driver Transaction Name (-DT)	-DT <i>tran_name</i>	✓	✓
Host (-HOST)	-HOST <i>host_name</i>	✓	✓
Password (-PW)	-PW <i>password</i>	✓	✓
Port (-PORT)	-PORT <i>port_number</i>	✓	✓
Service (-SERVICE)	-SERVICE <i>service</i>	✓	✓
Timeout (-TIMEOUT)	-TIMEOUT <i>milliseconds</i>	✓	✓
Trace (-T)	-T [E][+][filename]	✓	✓
Transaction Name (-TRAN)	-TRAN <i>tran_name</i>	✓	✓
User ID (-UID)	-UID <i>user_id</i>	✓	✓

- [Bridge Facility Like \(-BFL\)](#)
- [Bridge Keep Time \(-BKT\)](#)
- [Driver Transaction Name \(-DT\)](#)
- [Host \(-HOST\)](#)
- [Password \(-PW\)](#)
- [Port \(-PORT\)](#)
- [Service \(-SERVICE\)](#)
- [Timeout \(-TIMEOUT\)](#)
- [Trace \(-T\)](#)

- [Transaction Name \(-TRAN\)](#)
- [User ID \(-UID\)](#)

Bridge Facility Like (-BFL)

Use the Bridge Facility Like adapter command (`-BFL`) to specify the four character name of the terminal type. This terminal type is defined by the `FACILITYLIKE` parameter of the **PROFILE** associated with the transaction.

This is an optional command.

`-BFL terminal_id`

For example:

`-BFL cujo`

Where `cujo` is the four character name of the terminal type.

Bridge Keep Time (-BKT)

Use the Bridge Keep Time adapter command (`-BKT`) to specify the time, in milliseconds, for the CICS server to keep information (CICS COMMAREA) for the next transaction in the pseudo conversation.

This is an optional command. The default value is set by your CICS system programmer.

`-BKT time`

For example:

`-BKT 60`

Where `60` is the specified time in milliseconds.

Driver Transaction Name (-DT)

Use the Driver Transaction Name adapter command (`-DT`) to specify the name of the supplied driver transaction. If not provided this will default to MDRV, the default name of the driver transaction.

Check with your CICS system programmer to verify the name of the driver transaction.

Action

Meaning
`transaction_name`

Name of the supplied driver transaction.

Host (-HOST)

Use the Host adapter command (`-HOST` or `-H`) to specify the name of the system or `localhost` to which you want to connect.

This is a required command unless you are running on a native CICS platform. Do not use the `-HOST` command if you are using the CICS adapter within a CICS region.

`-HOST host_name`

For example:

`-HOST Your.CICS.system`

Where `host_name` is the name of the z/OS® system that hosts the CICS to which you want to connect.

Password (-PW)

Use the User Password adapter command (`-PW`) to specify the password associated with the user ID. The password is not displayed in the adapter trace file.

This is a required command.

`-PW password`

Where `password` is the password of the local user specified in the `-UID` adapter command, or the default user, if `-UID` is not specified.

Port (-PORT)

Use the Port adapter command (-PORT or -P) to specify the port number on which the CICS Listener listens. This is customer defined.

This is a required command if -HOST is specified. Do not use the -PORT command if you are using the CICS adapter from within a CICS region.

This command is mutually exclusive with the Service (-SERVICE) command. Either Port or Service must be specified.

-PORT *port_number*

For example:

-PORT 4321

Where 4321 is the port number where the CICS listener listens. Consult your CICS systems programmer for the port number.

Service (-SERVICE)

Use the Service adapter command (-SERVICE or -s) to specify the service name that maps to a port. See the Port (-PORT) command for more information.

This is a required command if -HOST is specified.

This command is mutually exclusive with the Port (-PORT) command. Either Port or Service must be specified.

-SERVICE *service*

For example:

-SERVICE cicslsnr

Where *cicslsnr* is the service name that maps to the specified port.

Timeout (-TIMEOUT)

Use the Timeout adapter command (-TIMEOUT) to specify the time limit for lookup operations in milliseconds. If not specified the timeout is infinite.

-TIMEOUT *milliseconds*

Option

Description

milliseconds

The time limit in milliseconds.

For example, to specify **10,000** milliseconds as the time limit for lookup operations.

-TIMEOUT 10000

Trace (-T)

Use the Trace adapter command (-T) to produce a diagnostics file. This file contains detailed information about adapter activity. By default, the **m4cics.mtr** trace file is created in the directory where the map is located.

This command also determines the trace option that is provided to the server side. The server side transaction and 3270 bridge exit will trace to SYSPRINT.

You can override the adapter command line trace options dynamically using the Management Console. See *Dynamic Adapter Tracing* in the *Launcher* documentation for information.

-T[E][+] [filename]

Option

Description

E

Produce a trace file containing only the adapter errors that occurred during map execution.

+

Appends trace information to the existing trace file.

filename

Creates a trace file with the specified name in the specified directory.

Transaction Name (-TRAN)

Use the Transaction Name adapter command (-TRAN) to specify the customer defined transaction name, configured to use 3270 bridge.

This is a required command.

-TRAN *tran_name*

Where *tran_name* is the customer defined application identifier.

User ID (-UID)

Use the User ID adapter command (-UID) to specify the User ID for connection and transaction authorization.

This command is required.

```
-UID user_id
```

Where *user_id* is the ID for CICS sign on.

Syntax summary

- [Data sources](#)
- [Data targets](#)
- [Example 1](#)
- [Example 2](#)

Data sources

The following is the command syntax of the CICS adapter commands used for data sources:

```
-DT tran_name
-HOST host_name
-PORT port_number|-SERVICE service
-TIMEOUT milliseconds
[-BFL terminal_id]
[-BKT time]
[-PW password]
[-T[E] [+][filename]]
[-TRAN tran_name]
[-UID user_id]
```

Data targets

The following is the command syntax of the CICS adapter commands used for data targets:

```
-DT tran_name
-HOST host_name
-PORT port_number|-SERVICE service
-TIMEOUT milliseconds
[-BFL terminal_id]
[-BKT time]
[-PW password]
[-T[E] [+][filename]]
[-TRAN tran_name]
[-UID user_id]
```

Example 1

For the PUT>Target setting in an output card, select CICS. In the PUT>Target>Command field, enter:

```
-HOST myhost -SERVICE cicslsnr -TIMEOUT 20000 -UID user -PW
pass -TRAN ABCD
```

This Target output command will:

- connect to **host myhost using service cicslsnr**
- change the socket **timeout** to 20 seconds
- use the **userid user and the password pass**
- run **transaction ABCD** passing the data mapped to the output card. The response from the transaction will be discarded, but its success status will be returned from the adapter

Example 2

In a **GET** map function call enter:

```
=GET("CICS",
"-T+ myfile.mtr
-HOST myhost -PORT 3020
-TIMEOUT 30000 -UID user -PW pass
-TRAN DINQ ", PACKAGE( MyPackage ))
```

This **GET** map function call will:

- Enable a trace file called **myfile.mtr** in append mode.
- Connect to **host myhost using port 3020**.
- Change the socket **timeout** to **30 seconds**.
- Use the **userid user** and the **password pass**.
- Run **transaction DINQ** using data supplied in **PACKAGE MyPackage**.

Using the CICS adapter

Use the CICS adapter to send and receive CICS transaction data.

- [Adapter capabilities](#)
- [Adapter limitations](#)
- [Defining a bridge transaction for use with the CICS adapter](#)

Adapter capabilities

The capabilities of the adapter include:

- Communication between the client and the CICS based server software is synchronous.
- The client map sends data to the CICS server, and waits for the reply. The CICS server software receives data, sends a reply, and quits.
- Pseudo conversational and non-conversational transactions are supported.

Adapter limitations

The following limitations exist in the adapter:

- The adapter assumes only one map instance per connection.
- CICS conversational transactions are not supported.

Defining a bridge transaction for use with the CICS adapter

- The details of defining a 3270 bridge transaction are covered in the [CICS Transaction Server for OS/390® Version 1.3 External Interfaces Guide \(SC33-1944\)](#).
- Examples for defining the bridge transaction can be found in the CICS adapter examples **readme.txt**.

Cipher Adapter

The Cipher adapter encrypts and decrypts data by using the Advanced Encryption Standard (AES) established by the U.S. National Institute of Standards and Technology (NIST). A single cryptographic key enciphers (encrypts) and deciphers data. The cryptographic key is stored in a master key file (MKF) in an AES-based encrypted format. As long as the MKF is not compromised or lost, the enciphered data is cryptographically secure.

Encrypted data is preceded by a unique header that identifies the specific master key used for encryption, similar to the following example:

[TX id=13104 iv=7804721d4eb50b8de9185960a587ee45 m=a h=15B333C7]

In the header example:

id=	Encryption master key ID
iv=	GSKit cipher initialization vector (IV)
m=a	ASCII or binary encryption mode
h=	Internal checksum

To decrypt the data, the specified MKF must contain the master key ID that's specified in the header of the encrypted data. The encryption mode can be ASCII or binary. Decryption automatically uses the mode that's appropriate for the encrypted data.

In Transformation Extender, encrypted files are interchangeable between Windows and UNIX operating systems. For example, you can encipher a file on a Windows system and run a map on a UNIX system to decipher it.

Because z/OS® systems use a different encryption format, you cannot decipher file on a z/OS system that you encrypted on a Windows or UNIX system.

- [IBM GSKit requirements](#)

The encryption capabilities of the Cipher adapter require IBM Global Security Kit (GSKit). By default, Transformation Extender installs GSKit locally. A locally installed GSKit enables Transformation Extender to run a specific version of GSKit that is automatically used by Transformation Extender executable programs. Products that Transformation Extender integrates with might use their own local version of GSKit. When you run Transformation Extender with other products that are configured to use GSKit, you can configure the environment to use the latest or most appropriate version.

- [Cipher adapter commands](#)

Cipher adapter commands are valid for input data sources and output data targets.

IBM GSKit requirements

The encryption capabilities of the Cipher adapter require IBM Global Security Kit (GSKit). By default, Transformation Extender installs GSKit locally. A locally installed GSKit enables Transformation Extender to run a specific version of GSKit that is automatically used by Transformation Extender executable programs. Products that Transformation Extender integrates with might use their own local version of GSKit. When you run Transformation Extender with other products that are configured to use GSKit, you can configure the environment to use the latest or most appropriate version.

To use the Cipher adapter in an API or RMI environment, ensure that the library and executable path of your platform points to the local GSKit drivers by running the dtxcommon.bat file (on Windows) or the setup script (on UNIX) before you run a map. In an Integrated Server environment, you must manually adjust the paths to point to the GSKit drivers.

To prevent Transformation Extender from using its locally installed version of GSKit, rename the *tx_install_dir\GSKit* directory to any other name. At runtime, Transformation Extender automatically detects the missing GSKit directory and searches the platform-specific environment path for the GSKit modules.

When you have a globally installed version of GSKit on the same system due to the requirements of another product, renaming the *tx_install_dir\GSKit* directory ensures that the Transformation Extender local GSKit is bypassed. The global version is managed by the GSKit installation program, and makes the security modules available on a system-wide basis.

To determine the version of the local GSKit that Transformation Extender installed:

1. From the command line, run the Transformation Extender dtxcommon.bat file or the setup script.
2. Run the **gsk8ver_64** command.

Cipher adapter commands

Cipher adapter commands are valid for input data sources and output data targets.

Cipher adapter command aliases

Use CIPHER as the adapter command alias on input and output cards and in GET and PUT rules. For example:

Input source override execution command	-IACIPHER <i>card_num</i>
Output target override execution command	-OACIPHER <i>card_num</i>

Cipher adapter commands

-MKF *filename*

Specifies the name of the master key file (MKF) in which to locate an existing master key or store a new master key. When the MKF contains multiple existing master keys, TX uses the last key in the MKF by default. Use the **-ID** option to select a different key.

This keyword is optional. If you omit it, the adapter automatically creates a new cryptographic key in the *tx_install_dir\itx.mkf* file on Windows systems or the *tx_install_dir\configs\itx.mkf* file on UNIX systems.

-ID *key_ID*

Specifies the ID of the cryptographic key to use to decipher encrypted data. This keyword is optional. If you omit it, the adapter uses the last key in the MKF.

-BIN

Specifies binary encryption. This keyword is optional. If you omit it, the adapter uses ASCII encryption.

-ACTION { ENCRYPT | DECRYPT }

Specifies whether to encrypt or decrypt the data. This keyword is optional on **GET** and **PUT** rules. If you omit it, TX encrypts or decrypts the data based on whether the specified file has an encryption header.

-FILE *filename*

Specifies the name of the file to read from or write to. This keyword is required on input cards and output cards.

-T [E | V] [+] *[file_path]*

The adapter trace level and full path to the adapter trace log.

-T

Log adapter informational messages.

-TE

Log only adapter errors during map execution.

-TV

Use verbose (debug) logging. The log file records all activity that occurs while the adapter is enciphering or deciphering data.

+

Appends the trace information to the existing log file. Omit this keyword to create a new log file.

file_path

The full path to the adapter trace log. If you omit this keyword, the adapter creates the m4cipher.mtr log file in the map directory.

Examples

This command enciphers data stored in Mydata text item.

```
= Valid(GET("CIPHER", "-MKF bulk_key_multiple.mkf -ACTION encipher -T", Mydata), FAIL("Err Msg: " + LASTERRORMSG() + " Err Code: " + LASTERRORCODE()))
```

This command deciphers data stored in Myencrypteddata text item.

```
= Valid(GET("CIPHER", "-MKF bulk_key.mkf -ACTION decipher -T", Myencrypteddata), FAIL("Err Msg: " + LASTERRORMSG() + " Err Code: " + LASTERRORCODE()))
```

On a map input card with CIPHER as the input source, the following Source... Command deciphers data in EncryptedInput.txt input file.

```
-FILE EncryptedInput.txt -ACTION decipher -MKF bulk_key.mkf -T
```

COM Adapter overview

You can use the Component Object Model (COM) Automation adapter to create a custom solution, integration, or service and access that solution from within IBM Transformation Extender.

The COM Automation adapter is targeted to those who would consider using the Software Development Kit (SDK) but would rather create the solution in a more natural and object-based model. Unlike the SDK, using the COM Automation adapter allows you to create your solution in any language that supports COM Automation (for example, C, C++, VB, Java™, COBOL and so on), and then provides method-based access to that solution.

- [System requirements](#)
- [COM Adapter command alias](#)
- [COM Adapter commands](#)
- [Syntax summary](#)
- [Using the adapter](#)
- [Troubleshooting](#)
- [Type Library Importer](#)

System requirements

See the IBM Transformation Extender system requirements for the minimum system requirements and operating system requirements for the COM Automation adapter. It is assumed that a Command Server has already been installed on the computer where the adapter is to be installed for runtime purposes.

COM Adapter command alias

Adapter commands can be specified by using a command string on the command line or creating a command file that contains adapter commands. The execution command syntax is:

```
-IA[alias]card_num  
-OA[alias]card_num
```

where **-IA** is the Input Source-Override execution command and **-OA** is the Output Target-Override execution command, *alias* is the adapter alias, and *card_num* is the number of the input or output card. The following table shows the adapter alias and its execution command.

Adapter	Alias	As Input	As Output
COM Automation	COM	-IACOM <i>card_num</i>	-OACOM <i>card_num</i>

COM Adapter commands

The following table lists valid commands for the COM Automation adapter, the command syntax, and whether the command is supported (✓) for use with data sources, targets, or both.

Command	Syntax	Source	Target
Input Card Configuration File (-CFG)	-CFG <i>full_path</i>	✓	
Prequel (-PRE)	-PRE [<i>dispid:vartype:data</i>]	✓	✓
Post (-POST)	-POST [<i>dispid:vartype:data</i>]	✓	✓
Program ID (-PROGID)	-PROGID <i>progid</i>	✓	✓
Set Property (-PROP)	-PROP [<i>dispid:vartype:data</i>]	✓	✓
Trace (-T)	-T[E][+] [<i>full_path</i>]	✓	✓

- [Input card configuration file \(-CFG\)](#)
- [Prequel \(-PRE\) command for COM](#)
- [Post \(-POST\) command for COM](#)
- [Program ID \(-PROGID\) command for COM](#)
- [Set Property \(-PROP\) command for COM](#)
- [Trace \(-T\) command for COM](#)

Input card configuration file (-CFG)

Use the Input Card Configuration File adapter command (-CFG) to allow using the COM Automation adapter on an input card using a pre-prepared metadata file.

```
-CFG full_path
```

Option	Description
--------	-------------

full_path

Specifies the full path and the location of the configuration file for a pre-prepared metadata file (usually created on another output card).
The configuration file can be generated by using the **PACKAGE** function on an imported type tree.

Prequel (-PRE) command for COM

Use the Prequel adapter command (-PRE) to specify if you want one or more methods to be executed before the methods in the type tree.

-PRE [dispid:vartype:data]

Option

Description

[]	The brackets surrounding the parameter are required.
:	Required separator.
dispid	Specify a valid dispatch ID (DISPID) property.
vartype	Specify a valid data type. See Supported Data Types for more information.
data	Data type as represented by the text. A string is a sequence of characters; an integer is a sequence of digits, and so on.

For example, to specify a dispatch ID of 3, a data type of BSTR, and text data:

-PRE [3:VT_BSTR:data]

Note: If any properties exist on the command line, the properties are set before executing any other call on the component. If an error is received setting any of the properties, the process is aborted with errors.

See [Determining the Dispatch ID for a Method/Property](#) for information on how to determine the dispatch ID for a property.

Post (-POST) command for COM

Use the Post adapter command (-POST) to specify execution of one or more methods after the methods in the type tree are executed.

-POST [dispid:vartype:data]

Option

Description

[]	The brackets surrounding the parameter are required.
:	Required separator.
dispid	Specify a valid dispatch ID (DISPID) property.
vartype	Specify a valid data type. See Supported Data Types for more information.
data	Data type as represented by the text. A string is a sequence of characters; an integer is a sequence of digits, and so on.

For example, to specify a dispatch ID of 3, a data type of BSTR, and text data:

-POST [3:VT_BSTR:data]

Note: If any properties exist on the command line, the properties are set before executing any other call on the component. If an error is received setting any of the properties, the process is aborted with errors.

See [Determining the Dispatch ID for a Method/Property](#) for information on how to determine the dispatch ID for a property.

Program ID (-PROGID) command for COM

Use the Program ID adapter command (-PROGID) to override the Class ID (CLSID) and specify a different component.

-PROGID progid

Option

Description

progid	The program ID of the component to use overrides the CLSID and specifies a different component.
--------	---

For example, to override the program ID Nxf.Nxf.1:

-PROGID Nxf.Nxf.1

As part of the type tree importer process, the Class ID (CLSID) of the component to use is stored in the type tree. This CLSID is then passed to the COM Automation adapter at run time to specify the component to use.

An example of where this option could be used would be in the case where you have a number of different components all implementing the same interface and you want to be able to select which component to use at map run time.

The different components might access different files or systems or might do something altogether different; it does not matter to the adapter, provided that the interface has the same Dispatch IDs (DISPIDs), method names, parameters, and so on.

Note: The Program ID (-PROGID) specified must be a compatible component with the same interface.

Set Property (-PROP) command for COM

Use the Set Property adapter command (-PROP) to set specific properties on the component at run time.

-PROP [dispid:vartype:data]

Option

Description

[]

The brackets surrounding the parameter are required.

:

Required separator.

dispid

Specify a valid dispatch ID (DISPID) property.

vartype

Specify a valid data type. See[Supported Data Types](#) for more information.

data

Data type as represented by the text. A string is a sequence of characters; an integer is a sequence of digits, and so on.

For example, to specify a dispatch ID of 3, a data type of BSTR, and text data:

-PROP [3:VT_BSTR:data]

Note: If any properties exist on the command line, the properties are set before executing any other call on the component. If an error is received setting any of the properties, the process is aborted with errors.

See [Determining the Dispatch ID for a Method/Property](#) for information on how to determine the dispatch ID for a property.

Trace (-T) command for COM

Use the Trace adapter command (-T) to produce a diagnostics file. This file contains detailed information about adapter activity. By default, the m4com.mtr trace file is created in the directory where the map is located.

-T[E][+] [full_path]

Option

Description

E

Produce a trace file containing only the adapter errors that occurred during map execution.

+

Append trace information to the existing trace file.

full_path

Creates a trace file with the specified name in the specified directory.

Note: Trace (-T) and Trace Error (-TE) adapter commands are mutually exclusive.

Syntax summary

- [COM adapter data sources](#)
- [COM adapter data targets](#)
- [Determining the Dispatch ID for a Method/Property](#)

COM adapter data sources

```
[-CFG full_path]
[-POST [dispid:vartype:data]]
[-PRE [dispid:vartype:data]]
[-PROGID progid]
[-PROP [dispid:vartype:data]]
[-T[E][+] [full_path]]
```

COM adapter data targets

```
[-POST [dispid:vartype:data]]
[-PRE [dispid:vartype:data]]
[-PROGID progid]
[-PROP [dispid:vartype:data]]
[-T[E] [+]] [full_path]
```

Determining the dispatch ID for a method or property

A number of the command line options for the COM Automation adapter require a dispatch ID (DISPID) for a parameter.

You can determine the DISPID for a method or property by:

- Viewing the IDL file
 - Running the OleView.exe Utility
 - [Viewing the IDL file](#)
 - [Running the oleview utility](#)
-

Viewing the IDL file

A simple way to determine the DISPID for a method or parameter is to view the IDL file.

The following is an excerpt from the IDL file for the included sample component, NxfSrv. The DISPID corresponds to the identification number in the square brackets above the method or property.

Note: There are no properties shown here.

```
interface INxf : IDispatch{
[id(1), helpstring("Open the File")]
HRESULT OpenFile([in, string] BSTR File);
[id(2), helpstring("Close current file (called automatically when object
is destroyed)")]
HRESULT CloseFile();
[id(3), helpstring("Retrieve data associated with Key")]
HRESULT GetEntry([in, string] BSTR Key, [out, retval] BSTR* Data);
[id(4), helpstring("Add a new entry named Key")]
HRESULT AddEntry([in, string] BSTR Key, [in, string] BSTR Data);
[id(5), helpstring("Retrieve count of entries")]
HRESULT GetEntryCount([out, retval] long* Count);
[id(6), helpstring("Get name of entry (Key) by integer index")]
HRESULT GetEntryName([in] long Entry, [out, retval] BSTR* Name);
[id(7), helpstring("Delete entry associated with Key")]
HRESULT DeleteEntry([in, string] BSTR Key);
[id(8), helpstring("Call after numerous adds and deletes")]
HRESULT PackFile([in, string] BSTR File);
[id(9), helpstring("Get entry identified by Key from File")]
HRESULT GetFileEntry([in, string] BSTR File, [in, string] BSTR Key, [in,
out] VARIANT* Data);
};
```

Running the oleview utility

A second alternative for determining the DISPID for a method or parameter is to use the oleview.exe utility. This utility is available from Microsoft as a free download. It is also included with Visual Studio.

To determine DISPID for a method:

1. Start the oleview.exe application.
2. From the **File** menu, select **View typelib**.
3. Choose the type library file.

Note: Select the same type library file that was specified in the Type Library Importer.

Using the adapter

Use the COM Automation adapter to encapsulate your technology in a more object-oriented way and use your objects in a natural way.

The COM Automation adapter adheres to the semantics of the automation interface, IDispatch. This allows the user to provide a custom, in-house solution or service for nearly any integration requirement and then access that service from within IBM Transformation Extender (in other words, the adapter can then be thought of as a conduit to this component).

At **design time**, the user:

- Runs the importer selecting the component to use in a map.
- Selects one or more methods to execute on that component.

At **map runtime**, the adapter:

- Creates an instance of the specified component. The user specifies which component to use when running the importer.
 - Calls the method(s) that the user specified in the importer.
 - Frees the component.
 - [COM Automation advantages](#)
 - [Adapter capabilities](#)
 - [Supported data types](#)
 - [Object persistence](#)
 - [DCOM server considerations](#)
 - [Standard COM Automation terms](#)
-

COM Automation advantages

Implementing the desired service(s) as a COM Automation component has a number of advantages:

- Location transparency - The component can reside anywhere on the network.
 - Language - The user can use any programming language that supports COM Automation (VB, C/C++, Java™, and so on).
 - Runtime selection of services - Simple, well-known tools can specify which component to use (for service upgrades, load balancing, and so on). The map itself need not change.
 - Data types - COM Automation supports a set of well-known data types. See [Supported Data Types](#) for more information.
 - Simple method-based mapping.
-

Adapter capabilities

The following capabilities exist for the COM Automation adapter:

- A card (or **GET** or **PUT** rule) represents all or part of an IDL.
 - Objects created in one card can be used as parameters to methods in subsequent calls. These objects must be created by invoking an operation that returns an object.
 - Reference labels identify objects. Objects that are returned from method calls are assigned a label by the adapter. This label is of the form **Ref.nn**, where **nn** is a number which uniquely identifies the object.
 - The return values of any of the methods can be returned from a **GET** call. The IDL importer provides a mechanism allowing the user to select which methods the return values should be provided for. These return values can be primitive types (for example, **int**), strings, arrays, or local or remote objects. In the case of remote objects, the reference label is returned so that the object can be used in a subsequent call.
 - The adapter provides **GET** and **PUT** methods, which are invoked by the Resource Manager when the adapter is being invoked as an output.
-

Supported data types

COM Automation supports a well-known set of data types such as structures and arrays. The COM Automation adapter supports SAFEARRAY array types. Structures are supported in the preferred form and secondary form.

Other types such as record sets and multidimensional arrays are not supported.

Data Type	Example	Length
VT_I2	32767	Number of characters that make up the number
VT_I4	7729976	Number of characters that make up the number
VT_R4	8822.333	Number of characters that make up the number including decimal point
VT_R8	8822.7776	Number of characters that make up the number including decimal point
VT_DATE	MM/DD/CCYY	10 always
VT_BSTR	A string	Number of characters in the string
VT_ERROR	0x80000001	10 always (must be hex)
VT_BOOL	0, 00, -1	1 or 2 always (2 chars needed for ` -1`)
VT_UI1	X	1 always
VT_VOID	(empty)	0 always
VT_INT	7729976	Number of characters that make up the number
VT_UINT	7729976	Number of characters that make up the number
VT_U12	7729976	Number of characters that make up the number
VT_U14	7729976	Number of characters that make up the number
VT_CARRAY	Not available	Number of elements in an array
VT_USERDEFINED	Ref.123	12 characters maximum

Note: Each of the numerals could include a leading negative (-) sign. All parameter types are variable length.

Object persistence

Object persistence provides the functionality of keeping all object instances in the object pool. This in turn allows multiple cards to access methods of all objects in this pool. This same behavior currently exists in the Java™ Class adapter.

See "[COM Automation Objects](#)" for more information.

DCOM server considerations

Distributed Component Object Model (DCOM) is an extension of the Component Object Model (COM) to support objects distributed across a network.

The DCOM component is created internally using a location context of **CLSCTX_ALL** to permit location transparency. This component can use the location context whether it is an in-process server (a DLL) or local/remote server (an executable). This also permits the use of a DCOM component over a network.

In the case of a DCOM server, the actual component used is determined by the configuration settings made in the Windows NT DCOMCNFG utility.

Standard COM Automation terms

The following table lists standard COM Automation terms and their definitions.

Term	Definition
DISPID	Dispatch ID - Number of characters that make up the number.
ProgID	Program ID - A text string used to identify the component. For example, Nxf.Nxf.1.
CLSID	Class ID - A numeric identifier used to specify a COM component. The class ID's are usually shown within curly braces. For example, {682553C1-20A8-11D2-8711-00A024299A30}.
IDL	Interface Definition Language - A standard syntax for defining objects and interfaces.
CoClass	A creatable object defined in the IDL file. The main interface to the properties and methods exposed by the object.

Troubleshooting

For information about error codes and messages returned by the adapters, see the Return Codes and Error Messages section in the Resource Adapters documentation.

If the COM Automation object returns an exception, which is indicated by the DISP_E_EXCEPTION return code, the exception is handled properly and the values of the EXCEPINFO structure are logged if trace has been enabled.

- [Error code propagation](#)

Error code propagation

There is no dedicated list of adapter specific return codes. Rather, the HRESULT return code from the component is returned.

These codes can be especially useful at map run time if components specify their own particular error codes and you use the **LASTERRORCODE** function to control future map execution based on your return codes.

The error (or warning) code is also sent back to the map as part of the status message in hexadecimal format (for example, **M4COM_DispatchInvoke failed: 0x80020004**). One error code that the adapter itself can return is **E_FAIL**. This indicates an adapter initialization error.

Type Library Importer

The Type Library Importer is accessible through the Importer Wizard. The Importer Wizard is an applet that is run from the Type Designer by selecting Import from the Tree menu.

The Importer Wizard uses a series of maps to convert metadata into a type tree script file (.mts). The Type Tree Maker then processes this type tree script and generates a type tree that contains all the supported types that are defined in the imported metadata.

Many of the type trees that are generated by the Importer Wizard can be immediately used for map development. However, depending on the contents of the interface-specific metadata file, it might be necessary for the generated type tree to be modified using the Type Designer.

Note: You cannot simultaneously have a type tree open in the Type Designer while importing a schema with the same name.

- [Using the Type Library Importer](#)
- [Structures](#)
- [Arrays](#)
- [COM Automation objects](#)

Using the Type Library Importer

The Type Library Importer reads the format of COM Automation Components and Methods to automatically generate type trees for use with the COM Automation Adapter.

To run the Type Library Importer

1. Open the Type Designer application.
2. Select Import a type tree in the Startup window and click Next. Or select Import from the Tree menu at anytime while the Type Designer is running. The Importer Wizard dialog box appears.
3. Select Type Library. Click Next.
The Importer Wizard-Type Library dialog box appears.
4. Select an object from the list or click Browse and select a type library. Click Next.
Note: There will be a number of components that show up in the list that are not compatible with the adapter.
5. Select the desired component and click Next.
6. Select one or more methods from the list and click Next.
Depending on the component and how it is going to be used, you might need to reorder the methods. You can arrange the methods by using a drag-and-drop technique.
7. Enter the destination type tree name in the **File Name** field or click Browse and select a type tree (.mtt) file. Click Next.
Note: If the destination file already exists, it will be overwritten.
8. Click Finish.
9. When prompted, click Yes to open the generated type tree(s). Click No to close the Importer Wizard and return to the Type Designer application.

Structures

Structures are supported in the preferred form as well as the secondary form...

For example:

Preferred Form

```
typedef struct tagTestStruct {  
long lCount;  
float fPortion;  
} TestStruct;  
HRESULT SetWhatever([in] const TestStruct *pX);
```

Secondary Form

```
struct TestStruct {  
long lCount;  
float fPortion;  
}  
HRESULT SetWhatever([in] const struct TestStruct *pX);
```

Structures are presented as an object in the type tree. Access to member fields is only possible through the references to the object pool.

In the map rules, these reference values are indicated by a **Ref.** initiator and the reference number (*nn*) which follows the initiator. The COM Automation adapter uses that initiator to recognize the reference to the object pool and provides functionality for accessing the member fields using that reference.

The following text clarifies structure usage:

- Nested structures (where one of the fields in a structure is in another structure) are supported, and those values are passed as a reference through the **ref** item.
- The use of arrays as fields in the structure is permitted.

For example:

```
typedef struct {  
long lLength;  
float fSum;  
} SimpleStructA;  
typedef struct {  
int iPencils;  
int iPapers;  
} SimpleStructB;  
typedef struct {  
int iAge;  
SimpleStructA simpleStructA;
```

Arrays

Arrays are ordered lists of values that are all of the same type. These values are known as the array's elements. The elements of an array can be any simple type or range type. The SAFEARRAY array type is supported for the COM Automation adapter.

- [SAFEARRAY Arrays](#)

SAFEARRAY Arrays

SAFEARRAY arrays can contain any of the supported simple data types. This type of array is a common way to support arrays in (oleautomation) interfaces.

The number of elements is not known at compile time, however it is known at run time, so they are represented as a group of variable size(s) in the type tree.

In the adapter call of **Invoke** method, an array size is determined according to those attributes.

For example:

```
HRESULT SumOfInts( [in] SAFEARRAY(long) intsToSum, [out] long* sum );
```

COM Automation objects

Apart from scalar types (structures and arrays) the only other objects that can be used as parameters of the interface methods, are the pointers to other interfaces.

This is the only way to access code running in different address spaces because the pointers to C++ classes or pointers to functions are off limits to remote method calls.

The pointers to these objects are stored in the global object pool, and are accessed through the references to them.

The Type Library Importer creates a single item in the type tree for each object used as an `input/output/inout` parameter, and that item contains a reference to the object pool for a given object. That reference is used on the COM Automation adapter side to access the object methods and members, using the global object pool.

Connect:Direct Adapter

The Connect:Direct® adapter integrates IBM® Transformation Extender with IBM Sterling Connect:Direct file transfer.

- [System requirements](#)
- [Configuration requirements](#)
- [Connect:Direct adapter command overview](#)

This table summarizes the adapter commands, syntax, and whether you can use the command with data sources, data targets, or both. You can specify the commands on the adapter command line or through the adapter properties.

System requirements

See the IBM® Transformation Extender [system requirements](#) and the [release notes](#) for Connect:Direct® adapter requirements.

Configuration requirements

To use the Connect:Direct® adapter, you must configure a Sterling Connect:Direct network map (netmap) with a primary node (PNODE) and secondary node (SNODE), and configure users for each node. Verify that you can transfer a file between the Sterling Connect:Direct nodes before you attempt to use the Connect:Direct adapter.

To use the Launcher service on a Microsoft Windows system, run the service as the PNODE user rather than as the local system account.

On Linux® and UNIX systems:

- Set NDMAPICFG environment variable:

```
export NDMAPICFG=/CD_install_dir/ndm/cfg/cliapi/ndmapi.cfg
```

where `CD_install_dir` is the Sterling Connect:Direct installation directory.

- The Sterling Connect:Direct configuration determines whether you must use the Linux or UNIX login credentials instead of the **-PUSER**, **-PPASS**, **-SUSER**, and **-SPASS** commands.

Connect:Direct adapter command overview

This table summarizes the adapter commands, syntax, and whether you can use the command with data sources, data targets, or both. You can specify the commands on the adapter command line or through the adapter properties.

The commands are not case-sensitive, but the command values are case-sensitive. Choices are separated by the OR symbol | . Optional parameters are enclosed in brackets []. The topics that follow provide the command syntax in railroad track format.

Table 1. Connect:Direct adapter command summary

Command	Syntax	Use with Data Source	Use with Data Target
Primary node	{-PNODE -PN} {host_name IP_addr}	Yes	Yes
Primary node user	{-PUSER -PU} user_name	Yes	Yes
Primary node password	{-PPASS -PP} password	Yes	Yes
Secondary node	{-SNODE -SN} {host_name IP_addr}	Yes	Yes
Secondary node user	{-SUSER -SU} user_name	Yes	Yes
Secondary node password	{-SPASS -SP} password	Yes	Yes

Command	Syntax	Use with Data Source	Use with Data Target
File name to GET or PUT	{-FILENAME -F} {filename absolute_path}	Yes	Yes
Connect:Direct® process name	{-PROCNAME -P} process_name	Yes	Yes
File disposition (how to open the file)	{-DISPOSITION -D} {RPL MOD NEW}	Yes	Yes
Binary transfer	{-BINARY -B} {YES NO}	Yes	Yes
Checkpoint to restart interrupted transmissions	{-CKPT -C} size	Yes	Yes
Wait interval	{-WAIT -WI} seconds	Yes	Yes
Directory restriction	{-DIRRESTRICT -DR} directory	Yes	Yes
Create trace file	{-TRACE -T} {file_name}	Yes	Yes
Append trace file	{-TRACE+ -T+} {file_name}	Yes	Yes
Trace adapter errors	{-TRACEERR -TE} {file_name}	Yes	Yes
Append adapter errors	{-TRACEERR+ -TE+} {file_name}	Yes	Yes
Use translation table	{-XLATE -X} {YES NO}	Yes	Yes

- **Command aliases**

Use the **CD** adapter alias to use the Connect:Direct adapter on input and output cards and in GET and PUT map rules.

- **Primary Node (-PNODE) command**

The **-PNODE** command identifies the primary node in a Sterling Connect:Direct connection. The primary node is the server that initiates the connection. A primary node can both send and receive data.

- **Primary node user (-PUSER) command**

The **-PUSER** command identifies the user ID that is configured for the primary node.

- **Primary node password (-PPASS) command**

The **-PPASS** command specifies the password for the primary node.

- **Secondary node (-SNODE) command**

The **-SNODE** command identifies the secondary node in a Sterling Connect:Direct connection. The secondary node is the server that receives the connection. A secondary node can both send and receive data.

- **Secondary node user (-SUSER) command**

The **-SUSER** command identifies the user ID that is configured in Sterling Connect:Direct for the secondary node.

- **Secondary node password (-SPASS) command**

The **-SPASS** command specifies the password for the secondary node.

- **File name (-FILENAME) command**

Specifies the name of the file to **GET** from or **PUT** to the server. This command is required.

- **Process name (-PROCNAME) command**

The **-PROCNAME** command specifies the name of the process for Sterling Connect:Direct to run when the adapter calls Sterling Connect:Direct.

- **File disposition (-DISPOSITION) command**

The **-DISPOSITION** command specifies how to open or write to a file.

- **Binary transfer (-BINARY) command**

The **-BINARY** command specifies whether files are transferred in binary or text mode.

- **Restart checkpoint (-CKPT) command**

The **-CKPT** command specifies the byte interval for checkpoint support, which restarts interrupted transmissions at the last valid checkpoint point and reduces the time to retransmit the file.

- **Wait interval (-WAIT) command**

The **-WAIT** command specifies the number of seconds that Sterling Connect:Direct is to wait for a Sterling Connect:Direct **SUBMIT** process to finish. In addition, the IBM Transformation Extender Connect:Direct listener uses the wait interval as the polling time for the Launcher.

- **Directory restriction (-DIRRESTRICT) command**

The **-DIRRESTRICT** adapter command is required when you configure the Sterling Connect:Direct property to restrict file uploads or downloads to a specified directory.

- **Create trace file (-TRACE) command**

The **-TRACE** command produces a diagnostics file that contains detailed information about adapter activity. The trace file is overwritten each time the trace runs.

- **Append trace file (-TRACE+) command**

The **-TRACE+** appends detailed diagnostic information about adapter activity to an existing trace file.

- **Trace adapter errors (-TRACEERR) command**

The **-TRACEERR** command produces a trace file that contains only the adapter errors that occurred during map execution.

- **Append adapter errors (-TRACEERR+) command**

The **-TRACEERR+** command appends only the adapter errors that occurred during map execution to an existing trace file.

- **Use translation table (-XLATE) command**

The **-XLATE** command specifies whether to use the default Sterling Connect:Direct translation table.

Command aliases

Use the **CD** adapter alias to use the Connect:Direct® adapter on input and output cards and in GET and PUT map rules.

Specify adapter commands by using a command string on the command line or by creating a command file that contains adapter commands. The execution command syntax is:

```
-IA[alias]card_num
-OA[alias]card_num
```

where:

-IA

The Input Source Override execution command

-OA

The Output Target Override execution command

alias

The adapter alias **CD**

card_num
The number of the map card

To override an input card 1:

-IACD1 'adapter_commands'

To override an output card 1:

-OACD1 'adapter_commands'

Primary Node (-PNODE) command

The **-PNODE** command identifies the primary node in a Sterling Connect:Direct® connection. The primary node is the server that initiates the connection. A primary node can both send and receive data.

This command is optional. If you omit this command, the default node is the primary node that is configured in Sterling Connect:Direct.

-PNODE command syntax



host_name
Specifies the host name of the primary node.

IP_addr
Specifies the IP address of the primary node. You can specify an IP address when netmap checking is set to No for the Sterling Connect:Direct PNODE system.

Primary node user (-PUSER) command

The **-PUSER** command identifies the user ID that is configured for the primary node.

Depending on how Sterling Connect:Direct® is configured, this command might be required.

-PUSER command syntax



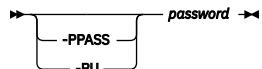
user_ID
The user ID that is configured in Sterling Connect:Direct for the primary node.

Primary node password (-PPASS) command

The **-PPASS** command specifies the password for the primary node.

This command is optional. Depending on how Sterling Connect:Direct® is configured, you might be required to use the Linux® or UNIX login credentials for the Sterling Connect:Direct connection instead of the **-PPASS** command.

-PPASS command syntax



password
See your Sterling Connect:Direct documentation for password requirements.

Secondary node (-SNODE) command

The **-SNODE** command identifies the secondary node in a Sterling Connect:Direct® connection. The secondary node is the server that receives the connection. A secondary node can both send and receive data.

This command is required.

-SNODE command syntax



host_name
Specifies the host name of the secondary node. You can specify an IP address when netmap checking is set to No for the Sterling Connect:Direct SNODE system.

IP_addr
Specifies the IP address of the secondary node.

Secondary node user (-SUSER) command

The **-SUSER** command identifies the user ID that is configured in Sterling Connect:Direct® for the secondary node.

Depending on how Sterling Connect:Direct is configured, this command might be required.

-SUSER command syntax



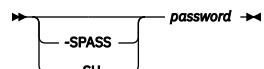
user_ID
The user ID that is configured in Sterling Connect:Direct for the secondary node.

Secondary node password (-SPASS) command

The **-SPASS** command specifies the password for the secondary node.

This command is optional. Depending on how Sterling Connect:Direct® is configured, you might be required to use the Linux® or UNIX login credentials for the Sterling Connect:Direct connection instead of the **-PPASS** command.

-SPASS command syntax



password
See your Sterling Connect:Direct documentation for password requirements.

File name (-FILENAME) command

Specifies the name of the file to **GET** from or **PUT** to the server. This command is required.

-FILENAME command syntax



file_name
Specifies a simple file name to **GET** from or **PUT** to a Sterling Connect:Direct® server. Use the file name only when Sterling Connect:Direct is configured with a directory restriction. Otherwise, use the absolute path to the file.
absolute_path
Specifies the absolute path to a file to **GET** from or **PUT** to a Sterling Connect:Direct server. Use the absolute path when Sterling Connect:Direct is not configured with a directory restriction.

Process name (-PROCNAME) command

The **-PROCNAME** command specifies the name of the process for Sterling Connect:Direct® to run when the adapter calls Sterling Connect:Direct.

This command is required.

-Procname command syntax



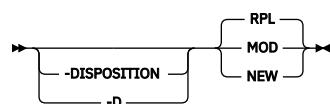
process_name
The name of the Sterling Connect:Direct process that is to run. The Sterling Connect:Direct **Select Statistics** command displays the process names.

File disposition (-DISPOSITION) command

The **-DISPOSITION** command specifies how to open or write to a file.

This command is optional. If you omit it, the default is RPL.

-DISPOSITION command syntax



RPL

Replace the existing file with the current file.

MOD

Add the contents of the current file to the existing file.

NEW

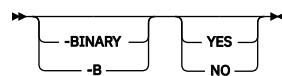
Create the file.

Binary transfer (-BINARY) command

The **-BINARY** command specifies whether files are transferred in binary or text mode.

This command is optional. If you omit it, the adapter does not specify the DATATYPE option On the Sterling Connect:Direct® **SUBMIT** command.

-BINARY command syntax



YES

Use binary file transfer.

NO

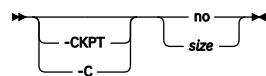
Use text transfer.

Restart checkpoint (-CKPT) command

The **-CKPT** command specifies the byte interval for checkpoint support, which restarts interrupted transmissions at the last valid checkpoint point and reduces the time to retransmit the file.

This command is optional. If you omit it, the default is the value that is specified by the ckpt.interval parameter in the Sterling Connect:Direct® initialization parameters file.

-CKPT command syntax



no

Specifies that no checkpoint is to be taken. See the Sterling Connect:Direct documentation to determine if this keyword is valid for your platform.

size

Specifies the amount data to be transmitted before the transmission is checked for failure.

See the Sterling Connect:Direct documentation for the values that are valid for your platform. Examples of how to specify the size:

100
Specifies 100 bytes.

1K
Specifies 1 KB.

4M
Specifies 4 MB.

1G
Specifies 1 GB.

Wait interval (-WAIT) command

The **-WAIT** command specifies the number of seconds that Sterling Connect:Direct® is to wait for a Sterling Connect:Direct **SUBMIT** process to finish. In addition, the IBM® Transformation Extender Connect:Direct listener uses the wait interval as the polling time for the Launcher.

Coordinate the use of this adapter command with any Sterling Connect:Direct retries that you configure in the Sterling Connect:Direct netmap.cfg file.

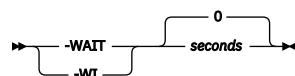
The adapter call fails if the wait interval expires before the **SUBMIT** process finishes.

- On an output card, the IBM Transformation Extender map fails with return code 76: Adapter failed to put data on output. The Sterling Connect:Direct process attempts to complete the copy.
- On an input card, the IBM Transformation Extender map fails with return code 12: Source not available. The Sterling Connect:Direct process attempts to complete the copy.

In each case, the adapter trace file contains the message No successful copy record retrieved from C:D in WAIT time.

This command is optional.

-WAIT command syntax



seconds

The number of seconds that Sterling Connect:Direct is to wait for the **SUBMIT** process to finish.

0

Specifies that Sterling Connect:Direct is to wait for an infinite period of time. This is the default.

When used as a Launcher source event, the **-WAIT** command determines the polling interval. 0 defaults to 10 seconds.

Use this option with caution. It can cause a hang by preventing Sterling Connect:Direct from returning to IBM Transformation Extender. For example, such a hang can occur if the SNODE is not available.

Directory restriction (-DIRRESTRICT) command

The **-DIRRESTRICT** adapter command is required when you configure the Sterling Connect:Direct® property to restrict file uploads or downloads to a specified directory.

This command is required on input cards or GET commands when a directory restriction is configured for the download directory. This command is required on output cards or PUT commands when a directory restriction is configured for the upload directory.

On Microsoft Windows systems, you configure the Sterling Connect:Direct directory restriction property by using the Select window. On Linux® and UNIX systems, you configure the Sterling Connect:Direct directory restriction property in the *C:D_install/ndm/cfg/node_name/userfile.cfg* file.

-DIRRESTRICT command syntax



directory

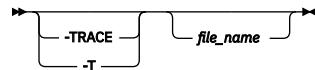
For input cards and GET commands, specifies the download location for the files. For output cards and PUT commands, specifies the upload location for the files.

Create trace file (-TRACE) command

The **-TRACE** command produces a diagnostics file that contains detailed information about adapter activity. The trace file is overwritten each time the trace runs.

This command is optional.

-TRACE command syntax



file_name

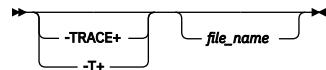
The name and path of the trace file. This parameter is optional. If you omit it, the adapter creates a trace file named m4cd.mtr in the IBM® Transformation Extender map directory.

Append trace file (-TRACE+) command

The **-TRACE+** appends detailed diagnostic information about adapter activity to an existing trace file.

This command is optional.

-TRACE+ command syntax



file_name

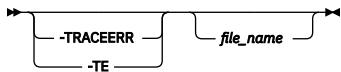
The name and path of the trace file. This parameter is optional. If you omit it, the adapter creates a trace file named m4cd.mtr in the IBM® Transformation Extender map directory.

Trace adapter errors (-TRACEERR) command

The **-TRACEERR** command produces a trace file that contains only the adapter errors that occurred during map execution.

This command is optional.

-TRACEERR command syntax



file_name

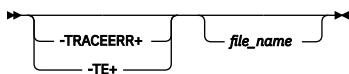
The name and path of the trace file. This parameter is optional. If you omit it, the adapter creates a trace file named m4cd.mtr in the IBM® Transformation Extender map directory.

Append adapter errors (-TRACEERR+) command

The **-TRACEERR+** command appends only the adapter errors that occurred during map execution to an existing trace file.

This command is optional.

-TRACEERR+ command syntax



file_name

The name and path of the trace file. This parameter is optional. If you omit it, the adapter creates a trace file named m4cd.mtr in the IBM® Transformation Extender map directory.

Use translation table (-XLATE) command

The **-XLATE** command specifies whether to use the default Sterling Connect:Direct® translation table.

This command is optional. If you omit it, the adapter does not specify the XLATE option on the **SUBMIT** command.

-XLATE command syntax



YES

Use the default Sterling Connect:Direct translation table.

NO

Do not use the default Sterling Connect:Direct translation table.

DB2 (Windows/UNIX) Adapter overview

Use the DB2® (Windows/UNIX) Adapter to access and manipulate data contained in databases that are DB2 data sources. You can install this adapter on additional systems for remote database connectivity.

Use the adapter with a map in a map rule, or use it with the Command Server, Launcher, or Software Development Kit.

- [System requirements](#)
- [Database columns and types](#)
- [Database Interface Designer settings](#)
- [DB2 \(Windows/UNIX\) Adapter commands](#)
- [Adapter commands for a source](#)
- [Adapter commands for a target](#)
- [Binding values in DBLOOKUP and DBQUERY](#)
- [Bulk copy example files](#)
- [Restrictions and limitations](#)
- [Return codes and error messages](#)

System requirements

The minimum system requirements and operating system requirements for the DB2® (Windows/UNIX) Adapter are detailed in [system requirements](#). See also the [release notes](#). The following tasks are also required:

- Verify that you have installed the DB2 client software.
- Verify that you have configured the DB2 data source or sources. For information about configuring the DB2 data source or sources, see the DB2 documentation.
- Verify that the DB2 environment variables are defined. For information about defining the DB2 environment variables, see your DB2 documentation.
- Set the shared library path environment variable for UNIX.

After installing the adapters on a UNIX platform and running the environment variable setup program (**setup**), add the installation directory to your shared library path environment variable if it is not already there. The following instructions assume your installation directory is defined in the environment variable DTX_HOME_DIR.

Note: After you have initially run the **setup** program, if you make changes to your path and/or library path at a later date, you need to re-run this program.

Note: These instructions assume that your UNIX environment is the Korn (ksh) shell and that the DB2_HOME environment variable points to the DB2 installation directory.

- For HP-UX

```
SHLIB_PATH=$DTX_HOME_DIR/lib:$DB2_HOME/lib:$SHLIB_PATH
export SHLIB_PATH
```

- For Sun Solaris

```
LD_LIBRARY_PATH=$DTX_HOME_DIR/lib:$DB2_HOME/lib:$LD_LIBRARY_PATH
export LD_LIBRARY_PATH
```

- For IBM® RS/6000® AIX®

```
LIBPATH=$DTX_HOME_DIR/lib:$DB2_HOME/lib:$LIBPATH
export LIBPATH
```

Database columns and types

The Database Interface Designer and **mtsmaker** with the Type Tree Maker generate type trees for queries, tables, views, and stored procedures in a DB2® DBMS (Database Management System). Item types will be created in a type tree that represent the data types of the columns of a query, table, view, or stored procedure.

The Database Interface Designer and **mtsmaker** get information about columns by calling DB2 to describe the columns associated with a query, table, view, or stored procedure. DB2 returns the data type, length, and other information to the Database Interface Designer and **mtsmaker**. The Database Interface Designer and **mtsmaker** then map the DB2 data types to types in a type tree.

- [Item type properties](#)
- [Date and Time Formats](#)

Item type properties

The following table lists the DB2® (Windows/UNIX) data types and the values of the item type properties to which they correspond when the type tree is generated.

DB2 Data Type	Interpret as	Item Subclass, Presentation	Length
INTEGER	Character	Number, Integer	11
SMALLINT	Character	Number, Integer	6
FLOAT	Binary	Number, Float	8
REAL	Binary	Number, Float	4
DOUBLE PRECISION	Binary	Number, Float	8
DECIMAL(n) ¹	Character	Number, Integer	*
DECIMAL(n,m) ²	Character	Number, Decimal	*
NUMERIC(n) ¹	Character	Number, Integer	*
NUMERIC(n,m) ²	Character	Number, Decimal	*
CHAR	Character	Text	*
VARCHAR	Character	Text	*
DATE	Character	Date & Time	10
TIME	Character	Date & Time	8
TIMESTAMP	Character	Date & Time	26
GRAPHIC	Binary	Text	*
VARGRAPHIC	Binary	Text	*
BLOB	Binary	Text	*
CLOB	Character	Text	*
DBCLOB	Character	Text	*

¹ $n \geq 1$

² $n \geq 1, m \geq 0$

* The DBMS dictates the length of this type.

Note: The BLOB, CLOB and DBCLOB data types are not supported as parameters for stored procedures.

Date and Time Formats

When using columns defined as date, time or timestamp in DB2® databases, the format for both input and output is:

Column	Format
Date	<code>ccyy-mm-dd</code>
Time	<code>hh:mi:ss</code>
Timestamp	<code>ccyy-mm-dd hh:mi:ss[.fff...]</code>

Formats

Format	Description
cc	two-digit century
yy	two-digit year
mm	two-digit month
dd	two-digit day
hh	two-digit hour
mi	two-digit minute
ss	two-digit second
.fff...	optional fractional seconds
fffff	six-digit fraction of a second
hh24	two-digit hour using a 24-hour day

In the Generate Type Tree from dialog, use the Represent date/time columns as text items check box to define whether to automatically format this information as Date & Time (which is with this check box disabled, the result of which is shown in the Item Subclass, Presentation column) as a text string. When generated as a text string, it might be necessary to use either the TEXTTODATE or TEXTTOTIME function in a map rule to convert the text string to the date and time format required by the database. If you are generating new type trees, it is recommended that you disable this check box.

This Represent date/time columns as text items check box is modal. After it has been disabled, it will remain disabled for all subsequent type tree generations, regardless of source. Therefore, you must be careful in determining this setting.

Database Interface Designer settings

When defining a DB2® (Windows/UNIX) database in the Database Interface Designer, in addition to the common settings available for all of the adapters in the Database Definition dialog, you need to enter information specific to DB2 (Windows/UNIX).

The DB2 (Windows/UNIX) Adapter-specific settings include:

Data Source > Database Interface Designer

The data source you defined in your DB2 (Windows/UNIX) development PC that is used by the Database Interface Designerto access the database information for design-time purposes.

Data Source > Runtime

The data source you defined in your DB2 (Windows/UNIX) database host platform to be used for access to the database for runtime (map execution) purposes both from the Map Designer and Command Server.

Security

- User ID The user ID used to connect to the database.
- Password The authorization password to connect to the DB2 subsystem.
- [Stored procedures native call syntax](#)

Stored procedures native call syntax

A stored procedure can be accessed from within a map by specifying the native call syntax in the following scenarios:

- A query that is specified in an input card.

- The first argument in a DBQUERY or DBLOOKUP function.
This argument does not need to be a literal. The arguments to the stored procedure may be determined at map execution time.
- The SQL Statement adapter command (-**STMT**) in DBQUERY, DBLOOKUP, or GET functions.

For information about using DBLOOKUP or DBQUERY, or for information about using the syntax for device-independent calls to access return values and output adapter commands refer to the Database Interface Designer documentation.

For example, a call to a stored procedure using DB2® in a DBLOOKUP might be:

```
DBLOOKUP ("{call MyAddNameProc(' " + Name:Column + " ',-1)}" ,
"mydb.mdq",
"MyDB")
```

Note: String literals must be contained within single quotation marks; adapter arguments must be separated by commas.

DB2® (Windows/UNIX) Adapter commands

Use the adapter commands when specifying data sources and targets. The applicability of many of the commands depends upon whether you are specifying a source or target, whether a database/query file (.mdq) is used, and the situations in which the usage of the command applies.

Adapter commands can be used in GET->Source->Command or PUT->Target->Command settings in the map designer and Integration Flow Designer, using GET, PUT, DBLOOKUP, or DBQUERY function calls, or overriding a data source or target using execution commands in a **RUN** function or on the command line.

- [Adapter-specific command list](#)

Adapter-specific command list

This documentation lists adapter commands that are DB2-specific database parameters. For a complete listing of all adapter commands, see the Resource Adapters documentation.

- ["Database Adapter Type \(-DBTYPE\)"](#)
- ["Row Count \(-ROWCNT\)"](#)
- ["Data Source \(-SOURCE\)"](#)
- [Database adapter type \(-DBTYPE\) adapter command for DB2 \(Windows/UNIX\)](#)
- [Row count \(-ROWCNT\) adapter command for DB2 \(Windows/UNIX\)](#)
- [Data Source \(-SOURCE\) adapter command for DB2 \(Windows/UNIX\)](#)

Database adapter type (-DBTYPE) adapter command for DB2 (Windows/UNIX)

Use the Database Adapter Type adapter command (-DBTYPE) to specify the database adapter type.

-DBTYPE DB2

Option	Description
DB2®	The database adapter type is DB2 (Windows and UNIX). Note: This command must be specified if the original card is not a database and no database is specified using the Database/Query adapter command (-MDQ) and the Database Name adapter command (-DBNAME).

The database adapter type is DB2 (Windows and UNIX).
Note: This command must be specified if the original card is not a database and no database is specified using the Database/Query adapter command (-MDQ) and the Database Name adapter command (-DBNAME).

Row count (-ROWCNT) adapter command for DB2 (Windows/UNIX)

Use the Row Count adapter command (-ROWCNT) to specify the number of rows to be retrieved per fetch.

-ROWCNT row_count

Option	Description
row_count	The number of rows to be retrieved per fetch. The time to fetch is optimized in accordance with the increase in the number of rows to retrieve. Therefore, a significant performance improvement may be gained by retrieving as many rows as possible in a single fetch.

Note: The number of rows retrieved is limited only by the amount of memory installed on the querying computer.
The default for the number of rows to be retrieved per fetch for the DB2® (Windows/UNIX) Adapter is one row.

Data Source (-SOURCE) adapter command for DB2 (Windows/UNIX)

Use the Data Source adapter command (-SOURCE) to specify the DB2® data source.

-SOURCE datasource

Option

Description

datasource

Specify the DB2 data source.

Adapter commands for a source

This summary shows the syntax of the adapter commands that can be used when defining a data source using an .mdq file or without using one, including both the required and optional adapter commands in the following situations:

- Using the `GET_>Source_>Command` setting in the Map Designer and Integration Flow Designer.
- Overriding a data source using the Input Source Override - Database execution command (`-ID`) using a RUN function or on the command line.
- Using a DBLOOKUP, DBQUERY, or GET function in map or component rules.

Adapter commands for a target

This summary shows the syntax of the adapter commands that can be used when defining a data target using an .mdq file or without using one, including both the required and optional adapter commands in the following situations:

- Using the `PUT_>Target_>Command` setting in the Map Designer and Integration Flow Designer.
- Overriding a data source using the Output Source Override - Database execution command (`-OD`) using a RUN function or on the command line
- Using the PUT function in map or component rules.

Binding values in DBLOOKUP and DBQUERY

When using a DBLOOKUP or DBQUERY function, use the Bind facility to submit similarly constructed SQL statements to the database server so that the statements are syntactically identical. By binding a value to a placeholder in the SQL statement, the actual syntax of the statement can be made static, which may improve performance. For more information about using bind values in database functions, refer to the Database Interface Designer documentation.

- [Limitation when using DB2](#)
- [Specifying the data type](#)

Limitation when using DB2

When binding values using DB2®, a limitation exists because DB2 cannot determine the data type of the column to which the value is being bound at the time the SQL statement executes. Therefore, the data type of the value being passed must be explicitly specified. Because of this DB2 limitation, the DB2 adapter determines the data type of the value according to the following rule:

If the value contains only numeric characters or numeric characters with a decimal point, the value is assumed to be numeric; otherwise, the value is assumed to be text.

This rule provides the expected results in most cases; however, there may be some situations in which the rule may not apply and it may be necessary to specify the exact data type. For example, if the value is a text field consisting solely of numeric characters, the value will be incorrectly interpreted as numeric unless correctly specified.

Specifying the data type

To override the default behavior and to explicitly specify the data type, a data type indicator must precede the bind value. The syntax for this is:

`:bind([T|N], value)`

The following table lists the data type indicators and the relationship between each data type indicator and the DB2® data types.

Data Type Indicator

DB2 Data Type

T or TEXT

Any non-numeric (including DATE, TIME, and TIMESTAMP)

N or NUM or NUMBER

Any numeric (INT, DECIMAL, REAL, and so on)

As an example of specifying a data type, to bind a value that is a text data type containing only digits to a statement, the SQL statement would be entered as:

```
DBLOOKUP ("Select * from Checks where CheckNumber=:bind([TEXT], " +
CheckNumber:Row "+")",
"DB.mdq",
"MyDB")
```

If the [TEXT] data type indicator is not specified, the trace file from the adapter displays the following error message:

```
Message: [IBM] [CLI Driver] [DB2/NT]SQL0301N
The value of a host variable in the EXECUTE or OPEN
```

statement cannot be used because of its data type.
SQLSTATE=07006.

Bulk copy example files

The bulk copy function transfers data between a table and an operation system file at high speeds. It is the most common way to load large amounts of data into a database table.

It is frequently used to transfer data between database vendors, spreadsheet applications, and servers.

The example files are located at *install_dir\examples\adapters\db2*.

Restrictions and limitations

The Database Interface Designer and adapters offer options and functions for accessing and manipulating data contained within a database. However, the following limitations exist:

- Database triggers are not supported. You cannot use a data source as an input event trigger for the Launcher.
- The BLOB, CLOB and DBCLOB data types are not supported as parameters for stored procedures.

Return codes and error messages

Return codes and messages are returned when the particular activity completes. Return codes and messages may also be recorded as specified in the audit logs, trace files, execution summary files, and so on.

For information about error codes and messages returned by database-specific adapters, refer to the Resource Adapters documentation.

Various troubleshooting tools are available in case you encounter problems while using the database adapters. For example, if you attempt to run a map that uses the adapter and encounter problems or do not receive the expected results, use the following adapter troubleshooting tools:

- adapter audit log (.log)
- adapter trace file (.mtr)

DB2 (z/OS ODBC) Adapter overview

Use the DB2® adapter for z/OS® (ODBC) for all z/OS environments except CICS®. You can use the adapter with a Command Server, the Platform API, or in a map rule. (For CICS, use the DB2 (z/OS) adapter and see the DB2 (z/OS) Adapter documentation for specific information.) It is recommended that you use the DB2 (z/OS ODBC) Adapter because it is more consistent with the behavior of other platforms, increasing the ease with which maps can be tested on other platforms.

This adapter also eases maintainability by eliminating the need to bind an IBM® Transformation Extender-specific plan each time a new release or maintenance is applied. You can also install these adapters on additional systems for remote database connectivity.

If you plan to only use mtsmaker without an .mdq file to create each type tree, you do not need to use the Database Interface Designer. However, you will still need to use the Database Interface Designer to create maps on the workstation. For information about using mtsmaker, see the Resource Adapters documentation.

The DB2 (z/OS ODBC) Adapter is available for the z/OS operating system with DB2.

- [System requirements](#)
- [Database columns and types](#)
- [Database Interface Designer settings](#)
- [DB2 \(z/OS ODBC\) Adapter commands](#)
- [Adapter commands for a source](#)
- [Adapter commands for a target](#)
- [Using tdfmaker to define update keys](#)
- [Using mtsmaker on z/OS](#)
- [DB2 \(z/OS ODBC\) Adapter limitations](#)
- [Return codes and error messages](#)

System requirements

The minimum system requirements and operating system requirements for the DB2® (z/OS® ODBC) Adapter are detailed in the [system requirements](#). See also the [release notes](#). In addition to these requirements, to install and run the DB2 (z/OS ODBC) Adapter:

- Verify that you have installed the Language Environment/370 runtime library. The adapter is compatible with all supported releases of LE/370.
- Verify that the Command Servers for both Batch and CICS® have been installed prior to using the adapters. For more information, see the Command Server documentation.
- Verify that the DB2 (z/OS ODBC) interface has been installed and the IBM-supplied plans for ODBC have been bound.

Database columns and types

The Database Interface Designer and **mtsmaker** with the Type Tree Maker generate type trees for queries, tables, and views in a DB2-compliant Database Management System (DBMS). Item types are created in a type tree that represents the data types of the columns of a query, table, or view.

The Database Interface Designer and **mtsmaker** get information about columns by calling DB2® to describe the columns associated with a query, table, or view. DB2 returns the data type, length, and other information to the Database Interface Designer and **mtsmaker**. The Database Interface Designer and **mtsmaker** then map the DB2 data types to types in a type tree.

- [How DB2 data types convert to item types in a type tree](#)
- [Date and time formats](#)

How DB2 data types convert to item types in a type tree

The following table lists the DB2® (z/OS® ODBC) data types and the values of the item type properties to which they correspond when a type tree is generated by the Database Interface Designer and mtsmaker. An asterisk (*) in the Length column indicates that the DBMS dictates the length of this type.

DB2 (z/OS ODBC) Data Type	Interpret as	Item Subclass, Presentation	Length
SQL_VARCHAR	Character	Text	*
SQL_CHAR	Character	Text	*
SQL_DECIMAL	Character	Number, Decimal	17
SQL_NUMERIC	Character	Number, Decimal	17
SQL_SMALLINT	Character	Number, Integer	6
SQL_INTEGER	Character	Number, Integer	11
SQL_REAL	Binary	Number, Float	4
SQL_FLOAT	Binary	Number, Float	8
SQL_DOUBLE	Binary	Number, Float	8
SQL_LONGVARCHAR	Character	Text	*
SQL_BIT	Character	Number, Integer	*
SQL_TINYINT	Character	Number, Integer	4
SQL_BIGINT	Character	Number, Integer	20
SQL_BINARY	Binary	Text	*
SQL_VARBINARY	Binary	Text	*
SQL_LONGVARBINARY	Binary	Text	*
SQL_DATE	Character	Date & Time	10
SQL_TIME	Character	Date & Time	8
SQL_TIMESTAMP	Character	Date & Time	19-26
SQL_INTERVAL_HOUR	Character	Text	*
SQL_INTERVAL_MINUTE	Character	Text	*
SQL_INTERVAL_SECOND	Character	Text	*
SQL_INTERVAL_YEAR_TO_MONTH	Character	Text	*
SQL_INTERVAL_DAY_TO_HOUR	Character	Text	*
SQL_INTERVAL_DAY_TO_MINUTE	Character	Text	*
SQL_INTERVAL_DAY_TO_SECOND	Character	Text	*
SQL_INTERVAL_HOUR_TO_MINUTE	Character	Text	*
SQL_INTERVAL_HOUR_TO_SECOND	Character	Text	*
SQL_INTERVAL_MINUTE_TO_SECOND	Character	Text	*
SQL_BLOB	Binary	Text	*
SQL_CLOB	Character	Text	*
SQL_DBCLOB	Character	Text	*

Note: The SQL_BLOB, SQL_CLOB and SQL_DBCLOB data types are not supported as parameters for stored procedures.

Date and time formats

When using columns defined as dates and times in DB2® (z/OS® ODBC) databases, use these formats:

Column	Format
Date	ccyy-mm-dd
Time	hh:mm:ss
Timestamp	ccyy-mm-dd hh:mm:ss[.fff...]

Formats

Format	Description
cc	two-digit century
yy	two-digit year
mm	two-digit month
dd	two-digit day
hh	two-digit hour
mi	two-digit minute
ss	two-digit second
.fff...	optional fractional seconds
ffffff	six-digit fraction of a second
hh24	two-digit hour using a 24-hour day

If the group format is fixed, the field is padded to 26 with trailing spaces.

Example:

2001-08-27 00:00:00 specifies August 27, 2001.

In the Generate Type Tree from dialog, use the Represent date/time columns as text items check box to define whether to automatically format this information as Date & Time (which is with this check box disabled, the result of which is shown in the Item Subclass, Presentation column) as a text string. When generated as a text string, it might be necessary to use either the TEXTTODATE or TEXTTOTIME function in a map rule to convert the text string to the date and time format required by the database. If you are generating new type trees, it is recommended that you disable this check box.

This Represent date/time columns as text items check box is modal. After it has been disabled, it will remain disabled for all subsequent type tree generations, regardless of source. Therefore, you must be careful in determining this setting.

Database Interface Designer settings

When defining a DB2® (z/OS® ODBC) database in the Database Interface Designer, in addition to the common settings available for all of the adapters in the Database Definition dialog, you need to enter information specific to DB2 (z/OS ODBC).

DB2 (z/OS ODBC) Adapter-specific settings include:

- **Data Source > Database Interface Designer**
The data source you defined in your development PC that is used by the Database Interface Designer to access the database information for design-time purposes.
 - **Data Source > Runtime**
The data source you defined in your ODBC database host platform to be used for access to the database for runtime (map execution) purposes both from the Map Designer and a Command Server.
 - **Security**
 - The user ID used to connect to the database.
 - The authorization password to connect to the DB2 subsystem.
- The IBM® DB2 implementation ignores the Security setting values. For more information, see the appropriate IBM documentation for the call-level interface.

Note: To see all common settings, see the Database Interface Designer documentation.

DB2 (z/OS® ODBC) Adapter commands

Use the adapter commands when specifying data sources and targets. The applicability of many of the commands depends upon whether you are specifying a source or target, whether a database/query file (.mdq) is used, and the situations in which the usage of the command applies.

Adapter commands can be used in GET > Source > Command or PUT > Target > Command settings in the Map Designer and Integration Flow Designer, using GET, PUT, DBLOOKUP, or DBQUERY function calls, or overriding a data source or target using execution commands in a RUN function or on the command line.

If connectivity to the mainframe-based DB2® is available, you can:

- View DB2 host database table names directly from Database Interface Designer
- Build type trees for DB2 host tables without using MTSMAKER
- Test maps on your workstation
- [**DB2 \(z/OS ODBC\) Adapter-specific commands**](#)
- [**Database Adapter Type \(-DBTYPE\) adapter command**](#)
- [**Row count \(-ROWCNT\) adapter command for DB2 \(z/OS ODBC\)**](#)
- [**Data Source \(-SOURCE\) adapter command**](#)

DB2® (z/OS® ODBC) Adapter-specific commands

This documentation lists adapter commands that are DB2-specific database parameters. For a complete listing of all adapter commands, see the Resource Adapters documentation.

Database Adapter Type (-DBTYPE) adapter command

Use the Database Adapter Type adapter command (-DBTYPE) to specify the database adapter type.

-DBTYPE ODBC

Option

Description

ODBC

The database adapter type is ODBC. In order to use the ODBC database adapter type, your systems programmer (DBA) must bind the ODBC default package and plans. For more information, see the topic regarding default plans and packages in the *IBM® DB2 Universal Database for z/OS®:ODBC Guide and Reference*.

Row count (-ROWCNT) adapter command for DB2 (z/OS ODBC)

Use the Row Count adapter command (-ROWCNT) to specify the number of rows to be retrieved per fetch.

-ROWCNT row_count

Option

Description

row_count

The number of rows to be retrieved per fetch. The time to fetch is optimized in accordance with the increase in the number of rows to retrieve. Therefore, a significant performance improvement may be gained by retrieving as many rows as possible in a single fetch.

The default for the number of rows to be retrieved per fetch for the DB2® (z/OS® ODBC) Adapter is one row.

Data Source (-SOURCE) adapter command

Use the Data Source adapter command (-SOURCE) to specify the ODBC data source or the data source for an .mdq file. Use this command only when using a non-default (alternate) plan. For more information, see the topic regarding default plans and packages in the *IBM® DB2 Universal Database for z/OS®:ODBC Guide and Reference*.

-SOURCE datasource

Option

Description

datasource

The ODBC data source.

Adapter commands for a source

This summary shows the syntax of the adapter commands that can be used when defining a data source using an .mdq file or without using one, including both the required and optional adapter commands in the following situations:

- Using the GET...Source...Command setting in the Map Designer and Integration Flow Designer.
- Overriding a data source using the Input Source Override - Database execution command (-ID) using a RUN function or on the command line.
- Using a DBLOOKUP, DBQUERY, or GET function in map or component rules.
- [GET > Source > Command setting](#)
- [Database execution command: input source override \(-ID\)](#)
- [DBLOOKUP and DBQUERY functions](#)
- [GET function](#)

GET > Source > Command setting

In an input card, when you specify Database as the value for the GET...Source setting, the Command field is displayed where you can enter adapter commands.

Database command options when there is a database/query file

[-DBTYPE ODBC]
[-SOURCE datasource]
[-STMT SQL_statement]
[-FILE [directory]]

```

[-VAR name=value...]
[-USER user_ID]
[-PASSWORD password]
[-CSTMT [number]]
[-ROWCNT row_count]
[-AUDIT[G] [+| [full_path]]
[{-TRACE|-TRACEERR} [+| [full_path]]]

```

Database command options when there is not a database/query file

```

-DBTYPE ODBC
-STMT SQL_statement
[-SOURCE datasource]
[-FILE [directory]]
[-VAR name=value...]
[-USER user_ID]
[-PASSWORD password]
[-CSTMT [number]]
[-ROWCNT row_count]
[-AUDIT[G] [+| [full_path]]
[{-TRACE|-TRACEERR} [+| [full_path]]]

```

Database execution command: input source override (-ID)

Use the Input Source Override - Database execution command (-ID) to designate a database as the source or you can override one or more of the adapter command settings or database definitions in a RUN function or on the command line.

When you are not using a database/query file (.mdq), the -SOURCE and -STMT commands are mandatory unless there is a default data source specified for DB2® ODBC. For detailed information about DB2 ODBC, see the DB2 Universal Database for z/OS®: ODBC documentation.

The adapter commands are shown using a single quotation mark, which is the Windows syntax. For non-Windows platforms, use two single quotation marks followed by one single quotation mark and end with one single quotation mark followed by two single quotation marks.

Scenario: compiled map source is a database

Database/query file (.mdq)	No database/query file (.mdq)
'[-MDQ mdq_file -DBNAME database_name] [-QUERY query_name -STMT SQL_stmt] [-FILE [directory]] [-VAR name=value..] [-USER username] [-PASSWORD password] [-CCARD -CSTMT [number]] [-ROWCNT row_count] [-AUDIT[G] [+ [full_path]]] [{-TRACE -TRACEERR} [+ [full_path]]]'	'-DBTYPE ODBC -STMT SQL_statement [-SOURCE datasource] [-FILE [directory]] [-VAR name=value...] [-USER user_ID] [-PASSWORD password] [-CCARD -CSTMT [number]] [-ROWCNT row_count] [-AUDIT[G] [+ [full_path]]] [{-TRACE -TRACEERR} [+ [full_path]]]'

Scenario: compiled map source is not a database

Database/query file (.mdq)	No database/query file (.mdq)
'-MDQ mdq_file -DBNAME database_name -QUERY query_name -STMT SQL_stmt [-FILE [directory]] [-VAR name=value..] [-USER username] [-PASSWORD password] [-CCARD -CSTMT [number]] [-ROWCNT row_count] [-AUDIT[G] [+ [full_path]]] [{-TRACE -TRACEERR} [+ [full_path]]]'	'-DBTYPE ODBC -STMT SQL_statement [-SOURCE datasource] [-FILE [directory]] [-USER user_ID] [-PASSWORD password] [-CCARD -CSTMT [number]] [-ROWCNT row_count] [-AUDIT[G] [+ [full_path]]] [{-TRACE -TRACEERR} [+ [full_path]]]'

DBLOOKUP and DBQUERY functions

The DBLOOKUP and DBQUERY functions can be used in component rules in the Type Designer and map rules in the Map Designer when creating a map that can be used with a database.

Using DBLOOKUP with a database/query file

```

DBLOOKUP ("SQL_statement",
"-MDQ mdq_file
-DBNAME database_name
[-USER username]
[-PASSWORD password]
[-CCARD|-CSTMT [number]]
[-ROWCNT row_count]
[-AUDIT[G] [+| [full_path]]]
[{-TRACE|-TRACEERR} [+| [full_path]]]")

```

Using DBQUERY without a database/query file

```
DBQUERY ("SQL_statement",
"-DBTYPE ODBC
[-SOURCE datasource]
[-USER username]
[-PASSWORD password]
[-CCARD|-CSTMT [number]]
[-ROWCNT row_count]
[-AUDIT[G] [+| [full_path]]
[{-TRACE|-TRACEERR} [+| [full_path]]]"
```

GET function

The GET function returns the data from the source adapter.

When you are not using a database/query file (.mdq), the -SOURCE and -STMT commands are mandatory unless there is a default data source specified for DB2® ODBC. For detailed information about DB2 ODBC, see the DB2 Universal Database for z/OS®: ODBC documentation.

Using GET with a database/query file

```
GET ("DB",
"-MDQ mdq_file
-DBNAME database_name
-QUERY query_name|-STMT SQL_stmt
[-FILE [directory]]
[-VAR name=value...]
[-USER username]
[-PASSWORD password]
[-CCARD|-CSTMT [number]]
[-ROWCNT row_count]
[-AUDIT[G] [+| [full_path]]
[{-TRACE|-TRACEERR} [+| [full_path]]]"
```

Using GET without a database/query file

```
GET "DB",
"-DBTYPE ODBC
-STMT SQL_stmt
[-SOURCE datasource]
[-FILE [directory]]
[-USER username]
[-PASSWORD password]
[-ROWCNT row_count]
[-CCARD|-CSTMT [number]]
[-AUDIT[G] [+| [full_path]]
[{-TRACE|-TRACEERR} [+| [full_path]]]"
```

Adapter commands for a target

This summary shows the syntax of the adapter commands that can be used when defining a data target using an .mdq file or without using one, including both the required and optional adapter commands in the following situations:

- Using the [PUT > Target > Command setting](#) in the Map Designer and Integration Flow Designer.
- Overriding a data source using the Output Source Override - Database execution command (-OD) using a RUN function or on the command line
- Using the PUT function in map or component rules.
- [PUT > Target > Command setting](#)
- [Database execution command: output source override \(-OD\)](#)
- [PUT function](#)

PUT > Target > Command setting

In an output card, when you specify Database as the value for the PUT > Target setting, the Command field is displayed so that you can enter adapter commands.

Database command options when there is a database/query file

```
[-DBTYPE ODBC]
[-SOURCE datasource]
[-PROC procedure_name|-TABLE table_name]
[-USER user_ID]
[-PASSWORD password]
[-CSTMT [number]]
[-DELETE]
[-UPDATE [OFF|ONLY]]
[-BADDATA[+| full_path]
```

```

[-AUDIT[G] [+]
 [full_path]
[{-TRACE|-TRACEERR} [+]
 [full_path]]]
```

Database command options when there is no database/query file

```

-DBTYPE ODBC
-PROC procedure_name|-TABLE table_name
[-SOURCE datasource]
[-USER user_ID]
[-PASSWORD password]
[-CSTMT [number]]
[-DELETE]
[-BADDATA[+] full_path]
[-AUDIT[G] [+]
 [full_path]]
[{-TRACE|-TRACEERR} [+]
 [full_path]]]
```

Database execution command: output source override (-OD)

Use the Output Source Override - Database execution command (-OD) to designate a database as a target or you can override one or more of the adapter command settings or database definitions in a RUN function or on the command line.

The adapter commands are shown using a single quotation mark, which is the Windows syntax. For non-Windows platforms, use two single quotation marks followed by one single quotation mark and end with one single quotation mark followed by two single quotation marks.

Scenario: compiled map target is a database

Database/query file (.mdq)	No database/query file (.mdq)
' [-MDQ mdq_file -DBNAME database_name] [-PROC procedure_name -TABLE table_name] [-USER username] [-PASSWORD password] [-CCARD -CSTMT [number]] [-DELETE] [-UPDATE [OFF ONLY]] [-BADDATA[+] full_path] [-AUDIT[G] [+] [full_path]] [{-TRACE -TRACEERR} [+] [full_path]]'	' -DBTYPE ODBC -PROC procedure_name -TABLE table_name [-SOURCE datasource] [-USER user_ID] [-PASSWORD password] [-CCARD -CSTMT [number]] [-DELETE] [-UPDATE [OFF ONLY]] [-BADDATA[+] full_path] [-AUDIT[G] [+] [full_path]] [{-TRACE -TRACEERR} [+] [full_path]]'

Scenario: compiled map target is not a database

Database/query file (.mdq)	No database/query file (.mdq)
' -MDQ mdq_file -DBNAME database_name -PROC procedure_name -TABLE table_name [-USER username] [-PASSWORD password] [-CCARD -CSTMT [number]] [-DELETE] [-UPDATE [OFF ONLY]] [-BADDATA[+] full_path] [-AUDIT[G] [+] [full_path]] [{-TRACE -TRACEERR} [+] [full_path]]'	' -DBTYPE ODBC -PROC procedure_name -TABLE table_name [-SOURCE datasource] [-USER user_ID] [-PASSWORD password] [-CCARD -CSTMT [number]] [-DELETE] [-BADDATA[+] full_path] [-AUDIT[G] [+] [full_path]] [{-TRACE -TRACEERR} [+] [full_path]]'

PUT function

Use the PUT function to pass data to the target adapter.

The -SOURCE and -PROC|-TABLE commands when used **Without Database/Query File** are mandatory unless there is a default data source specified for DB2® ODBC. For detailed information on DB2 ODBC, see the DB2 Universal Database for z/OS®: ODBC documentation.

Using PUT with a database/query file

```

PUT ("DB",
"-MDQ mdq_file
-DBNAME database_name
-PROC procedure_name|-TABLE table_name
[-USER username]
[-PASSWORD password]
[-CCARD|-CSTMT [number]]
[-DELETE]
[-UPDATE [OFF|ONLY]]
[-BADDATA[+] full_path]
[-AUDIT[G] [+]
 [full_path]]
[{-TRACE|-TRACEERR} [+]
 [full_path]]")
```

Using PUT without a database/query file

```

PUT "DB",
"-DBTYPE ODBC
-PROC procedure_name|-TABLE table_name
[-SOURCE datasource]
[-USER username]
[-PASSWORD password]
[-CCARD|-CSTMT [number]]
[-DELETE]
[-BADDATA[+] full_path]
[-AUDIT[G][+] [full_path]]
[{-TRACE|-TRACEERR}[+] [full_path]]")

```

Using tdfmaker to define update keys

To insert the data produced by a mapping operation as new rows into a database table or to update only specific columns, you need to define the key columns and the columns to update. If you have access to your DB2® database through ODBC, use the Database Interface Designer to designate certain columns in a database as key columns in a table. For information about defining update key columns using the Database Interface Designer, see the Database Interface Designer documentation. If you do not have ODBC connectivity to your DB2 database, use the tdfmaker installed on your z/OS® system during the installation of the adapters for DB2.

When you run tdfmaker on your z/OS system, tdfmaker interrogates all of the tables in the library or each specified table name. It derives a listing of the column names defined in the tables and then outputs this information to a table definition file (.tdf).

After tdfmaker completes, transfer the table definition file in TEXT mode to your development PC, specifying the file name extension as .tdf. Then use the Database Interface Designer to process the .tdf so that you can specify update keys. Because you do not have ODBC connectivity when using the Database Interface Designer, when you select Set Update Keys from the Database menu, you are prompted for the name of the .tdf.

- [Using tdfmaker on z/OS](#)

Using tdfmaker on z/OS

When executing tdfmaker on z/OS®, the following adapter commands are required in your JCL (Job Control Language):

- The first adapter command is the DDNAME that identifies the output file.
- The -D adapter command identifies the DB2® subsystem.
- [Sample tdfmaker.JCL file](#)

Sample tdfmaker.JCL file

The following is sample JCL for using tdfmaker:

```

//TDFMAKER EXEC PGM=TDFMAKER,PARM='`-@PARMS'
//-----*
//** Load library containing TDFMAKER and DBUTIL
//**-----*
//STEPLIB DD DSN=<load_library>,DISP=SHR
//-----*
//** Print datasets
//**-----*
//SYSPRINT DD SYSOUT=<sysout_class>
//SYSOUT DD SYSOUT=<sysout_class>
//SYSUDUMP DD DUMMY
//-----*
//** The output TDF file. This must be a variable format file. *
//** Maximum record length is optional; 255 is sufficiently   *
//** large while not being too large to allow viewing in ISPF. *
//**-----*
//OUTPUT DD DSN=<tdf_file>,
//          DISP=(NEW,CATLG,DELETE),
//          DCB=(LRECL=255,RECFM=VB),
//          UNIT=<unit_adapter command>,
//          SPACE=<space_allocation>
//**-----*
//** The -@ option allows TDFMAKER parms optionally to reside  *
//** in a file                                              *
//**-----*
//PARMS DD *
OUTPUT
-D <DB2_subsystem_name>
<table_name>
<table_name>
...
/*
```

Using mtsmaker on z/OS

When using the adapters for DB2® on z/OS®, use mtsmaker to assist with generating a type tree for a query, table, or view when you do not have connectivity between the Database Interface Designer and your database. Note the following when using mtsmaker on z/OS:

- When you use `mtsmaker` and do not specify all required adapter commands, you are not prompted for this information. If an adapter command is missing or incorrect, the program reports this fact and stops running.
 - Typing `-H` or `?` on the command line does not access help for the parameters.
 - You can use a database/query file (`.mdq`) that was created with the Database Interface Designer in your z/OS environment by transferring it, in TEXT mode, to the system hosting your database. Transfer all `.mdq` files and tree script files (`.mts`) in TEXT mode. For example, the following settings are used: ASCII/EBCDIC translation and carriage-return/line-feed delimiters. Use the ASCII and CRLF options of the IND\$FILE transfer program or the ASCII option of FTP.
 - Wherever a file name is coded, it must refer to a DDNAME, never to a dataset name.

• [**mtsmaker parameters for z/OS**](#)

- mtsmaker parameters for z/OS

mtsmaker parameters for z/OS

The following are the additional mtsmaker parameters that are applicable only when used on z/OS®.

Adapter command

Description

-D subsystem_name
Required command when using mtsmaker without a database/query file (.mdq). Used to identify a z/OS DB2® data source. This name should refer to the name of the DB2 subsystem you intend to use.

the DB2 subsystem
Application plan name

The `-i` command is obsolete and is no longer required.

The -L command

The **PRNAME** of a file containing commands that is used (that is, the file) instead of the PARM statement.

- Sample JCL files for mtssmaker on z/OS

Sample JCL files for mtssmaker on z/OS®

The following is sample JCL for using mtsmaker with an .mdq file:

```
//MTSMAKER EXEC PGM=MTSMAKER,
// PARM=-Q <query_name> -M MDQFILE -F <query_name>.MTT -O OUTPUT
//*
//** Load library containing MTSMAKER and DBUTIL *
//*
//STEPLIB DD DSN=<load_library>,DISP=SHR
//*
//** Print datasets *
//*
//SYSPRINT DD SYSOUT=<sysout_class>
//SYSOUT DD SYSOUT=<sysout_class>
//SYSUDUMP DD DUMMY
//*
//** The MDQ file
//*
//MDQFILE DD DSN=<mdq_file>,DISP=SHR
//*
//** The output MTS file. This must be a variable format file.
//** Maximum record length is optional; 255 is sufficiently *
//** large while not being too large to allow viewing in ISPF. *
//*
//OUTPUT DD DSN=<mts_file>,
//          DISP=(NEW,CATLG,DELETE),
//          DCB=(LRECL=255,RECFM=VB),
//          UNIT=<unit_adapter_command>,
//          SPACE=<space_allocation>
```

The following is sample JCL for using mtmaker without an .mdq file:

```
/*MTSMAKER EXEC PGM=MTSMAKER,PARM='`-@PARMS'  
/*  
/*  Load library containing MTSMAKER and DBUTIL *  
/*  
//STEPLIB DD DSN=<load_library>,DISP=SHR  
/*  
/*  Print datasets *  
/*  
//SYSPRINT DD SYSOUT=<sysout_class>  
//SYSOUT DD SYSOUT=<sysout_class>  
//SYSUDUMP DD DUMMY  
/*  
/* The output MTS file. Can be a member of a PDS as long as  
/* the format is variable.  
/*  
//OUTPUT DD DSN=<mts_library>(<member_name>),DISP=SHR  
/*  
/* The -@ option allows MTSMAKER parms to reside in a file  
/*  
//PARMS DD *  
-T <table_name>  
-D <DB2 subsystem name>
```

```
-F <table>.MTT -O OUTPUT  
/*
```

DB2® (z/OS® ODBC) Adapter limitations

The Database Interface Designer and adapters offer options and functions for accessing and manipulating data contained within a database. However, there are some restrictions and limitations of certain functions:

- **Stored Procedures**

Using stored procedures to access adapter commands and return values from stored functions is not supported.

- **Bind Facility**

Using bind values in database functions is not supported.

- **Database Triggers**

Using a data source as an input event trigger for the Launcher is not supported.

The SQL_BLOB, SQL_CLOB and SQL_DBCLOB data types are not supported as parameters for stored procedures.

Return codes and error messages

Return codes and messages are returned when the particular activity completes. Return codes and messages can also be recorded as specified in the audit logs, trace files, execution summary files, and so forth.

For information about error codes and messages returned by database-specific adapters, see the Resource Adapters documentation.

Various troubleshooting tools are available in case you encounter problems while using the database adapters. For example, if you attempt to run a map that uses the adapter and encounter problems or do not receive the expected results, use the following adapter troubleshooting tools:

- adapter audit log (.log)
- adapter trace file (.mtr)

DB2 (z/OS) Adapter overview

Use the DB2® (z/OS®) Adapter to access and manipulate data contained in databases that are DB2 data sources in the CICS® and IMS environment. For all other z/OS environments, use the DB2 (z/OS ODBC) adapter. For more information about the DB2 (z/OS ODBC) adapter, see the DB2 (z/OS ODBC) documentation. You can install adapters on additional systems for remote database connectivity.

You can use the DB2 (z/OS) Adapter with a Command Server, the Software Development Kit, or with a map in a map rule.

If you plan to only use **mtsmaker** without an .mdq file to create each type tree, you do not need to use the Database Interface Designer. However, you will still need to use the Database Interface Designer to create maps on the workstation. For information about using **mtsmaker**, see the Resource Adapters documentation.

- [System requirements](#)
- [Binding the application plan](#)
- [Database columns and types](#)
- [Database Interface Designer settings](#)
- [DB2 \(z/OS\) Adapter commands](#)
- [DB2 \(z/OS\) Adapter limitations](#)
- [Return codes and error messages](#)

System requirements

Before installing and running the DB2® (z/OS®) Adapter, verify that you have installed the Language Environment® (LE) runtime library. The DB2 (z/OS) Adapter is compatible with all supported releases of LE.

For more information, see the Command Server documentation.

Binding the application plan

Ensure that the application plan for the adapter you intend to use has been bound. For DB2® (z/OS® ODBC), these plans are supplied by IBM®. If you want to use the native DB2 adapter, the DBUTILE plan must be bound as described in the IBM Transformation Extender z/OS Configuration documentation.

Database columns and types

The Database Interface Designer and **mtsmaker** with the Type Tree Maker generate type trees for queries, tables, and views in a DB2-compliant Database Management System (DBMS). Item types are created in a type tree that represents the data types of the columns of a query, table, or view.

The Database Interface Designer and **mtsmaker** get information about columns by calling DB2® to describe the columns associated with a query, table, or view. DB2 returns the data type, length, and other information to the Database Interface Designer and **mtsmaker**. The Database Interface Designer and **mtsmaker** then map the DB2 data types to types in a type tree.

- [Item type properties](#)
- [Date and time formats](#)

Database Interface Designer settings

When you define a DB2® (z/OS®) database in the Database Interface Designer, in addition to the common settings available for all of the database-specific adapters in the Database Definition dialog, you need to enter information specific to DB2 (z/OS).

DB2 (z/OS) Adapter-specific settings include:

- **DB2 > Plan name**
This is the name used to bind the plan when the IBM® Transformation Extender products were installed.
- **DB2 > Subsystem name**
This is the name used to specify the DB2 subsystem.
- **Have access to DB2 through ODBC**
This check box determines whether you will access your DB2 database through ODBC and Client Access for Windows.
If you do have ODBC access, the remaining fields must be assigned values.
- **ODBC Definition > Data source**
This is the data source you defined in your development computer that is used by the Database Interface Designer to access the database information for design-time purposes.
- **Security > User ID**
This is the user ID to connect to the DB2 subsystem.
This field is used by the Database Interface Designer only. It is ignored when the map is run on z/OS.
- **Security > Password**
This is the authorization password to connect to the DB2 subsystem.
This field is used by the Database Interface Designer only. It is ignored when the map is run on z/OS.

DB2 (z/OS®) Adapter commands

Use the adapter commands when specifying data sources and targets. The applicability of many of the commands depends upon whether you are specifying a source or target, whether a database/query file (.mdq) is used, and the situations in which the usage of the command applies.

Adapter commands can be used in GET > Source > Command or PUT > Target > Command settings in the Map Designer and Integration Flow Designer, using GET, PUT, DBLOOKUP, or DBQUERY function calls, or overriding a data source or target using execution commands in a RUN function or on the command line.

If connectivity to the mainframe-based DB2® is available, you can:

- View DB2 host database table names directly from Database Interface Designer
- Build type trees for DB2 host tables without using MTSMAKER
- Test maps on your workstation
- [DB2 \(z/OS\) Adapter-specific commands](#)
- [Database adapter type adapter command for DB2 \(z/OS\)](#)
- [Database/query file adapter command for DB2 \(z/OS\)](#)
- [DB2 \(z/OS\) Adapter source commands](#)
- [DB2 \(z/OS\) Adapter commands for a target](#)

DB2 (z/OS) Adapter-specific commands

The following section lists and describes those adapter commands that are DB2-specific database parameters. For a complete listing of all database-specific adapter commands, see the DB2® (z/OS®) Adapter.

- [Database Adapter Type \(-DBTYPE\)](#)
- [Database/Query File \(-MDQ\)](#)

Database adapter type adapter command for DB2® (z/OS)

Use the Database Adapter Type adapter command (-DBTYPE) to specify the database adapter type.

-DBTYPE {DB2MVS | ODBC}

Option

Description

DB2MVS

The database adapter type is z/OS® and you must specify an **.mdq** file.

This must be specified if the original card is not a database and no database is specified using the Database/Query adapter command (**-MDQ**) and the Database Name adapter command (**-DBNAME**).

ODBC

The database adapter type is ODBC. In order to use the ODBC database adapter type, your systems programmer (DBA) must bind the ODBC default package and plans. For more information, see the topic regarding default plans and packages in the *IBM® ODBC* documentation.

Database/query file adapter command for DB2® (z/OS®)

Use the **-MDQ** command to specify the DDNAME that identifies the .mdq file.

-MDQ DDNAME

Option**Description****DDNAME**

Specify the DDNAME that identifies the .mdq file.

DB2® (z/OS®) Adapter source commands

This summary shows the syntax of the adapter commands that can be used when defining a data source using an .mdq file or without using one, including both the required and optional adapter commands in the following situations:

- Using the **GET > Source > Command** setting in the Map Designer and Integration Flow Designer.
- Overriding a data source using the Input Source Override - Database execution command (**-ID**) using a RUN function or on the command line.
- Using a DBLOOKUP, DBQUERY, or GET function in map or component rules.
- [GET > source > command for DB2 \(z/OS\) Adapter](#)
- [DB2 \(z/OS\) Adapter database execution command \(input source override\)](#)
- [DBLOOKUP or DBQUERY functions](#)
- [GET function](#)

GET > source > command for DB2® (z/OS®) Adapter

Use the Map Designer or Integration Flow Designer to specify Database as the value for the **GET > Source** setting and enter the database-specific adapter commands as desired for the **Command** setting.

Database command options when there is a database/query file:

```
'' [-DBTYPE DB2MVS]
[-SOURCE datasource]
[-STMT SQL_statement]
[-VAR name=value...]
[-CSTMT [number]]
[{-TRACE [file|ERROR]}]'''
```

Database command options when there is no database/query file:

```
'' -DBTYPE ODBC
-STMT SQL_statement
[-SOURCE datasource]
[-VAR name=value...]
[-CSTMT [number]]
[{-TRACE [file|ERROR]}]'''
```

DB2® (z/OS®) Adapter database execution command (input source override)

Use the Input Source Override - Database execution command (**-ID**) to designate a database as the source or you can override one or more of the adapter command settings or database definitions in a RUN function or on the command line.

The adapter commands are shown using a single quotation mark, which is the Windows syntax. For non-Windows platforms, use two single quotation marks followed by one single quotation mark and end with one single quotation mark followed by two single quotation marks.

Scenario: compiled map source is a database

Database/query file (.mdq)

No database/query file (.mdq)

Database/query file (.mdq)	No database/query file (.mdq)
'-MDQ DDNAME -DBNAME database_name [-QUERY query_name -STMT SQL_stmt] [-VAR name=value...] [-CCARD -CSTM'T [number]] [{-TRACE [file ERROR]}]'	'-DBTYPE ODBC -STMT SQL_statement [-SOURCE datasource] [-VAR name=value...] [-CCARD -CSTM'T [number]] [{-TRACE [file ERROR]}]'

Scenario: compiled map source is not a database

Database/query file (.mdq)	No database/query file (.mdq)
'-MDQ DDNAME -DBNAME database_name [-QUERY query_name -STMT SQL_stmt] [-VAR name=value...] [-CCARD -CSTM'T [number]] [{-TRACE [file ERROR]}]'	'-DBTYPE ODBC -STMT SQL_statement [-SOURCE datasource] [-CCARD -CSTM'T [number]] [{-TRACE [file ERROR]}]'

DBLOOKUP or DBQUERY functions

The DBLOOKUP and DBQUERY functions can be used in component rules in the Type Designer and map rules in the Map Designer when creating a map that can be used with a database without specifying a database/query file name (.mdq) or a database name.

Using DBLOOKUP with a database/query file:

```
DBLOOKUP
(''SQL_statement'',
'-MDQ DDNAME
-DBNAME database_name
[-CCARD|-CSTM'T [number]]
[{-TRACE [file|ERROR]}]''')
```

Using DBQUERY without a database/query file:

```
DBQUERY
(''SQL_statement'',
'-DBTYPE ODBC
[-SOURCE datasource]
[-CCARD|-CSTM'T [number]]
[{-TRACE [file|ERROR]}]''')
```

GET function

The GET function returns the data from the source adapter.

Using GET with a database/query file:

```
GET (''DB'', ''
-MDQ DDNAME
-DBNAME database_name
[-QUERY query_name|-STMT SQL_stmt
[-VAR name=value...]
[-CCARD|-CSTM'T [number]]
[{-TRACE [file|ERROR]}]''')
```

Using GET without a database/query file:

```
GET ''DB'', ''
-DBTYPE ODBC
-STMT SQL_stmt
[-SOURCE datasource]
[-CCARD|-CSTM'T [number]]
[{-TRACE [file|ERROR]}]''')
```

DB2® (z/OS®) Adapter commands for a target

This summary shows the syntax of the adapter commands that can be used when defining a data target using an .mdq file or without using one, including both the required and optional adapter commands in the following situations:

- Using the PUT > Target > Command setting in the Map Designer and Integration Flow Designer.
- Overriding a data source using the Output Source Override - Database execution command (-OD) using a RUN function or on the command line
- Using the PUT function in map or component rules.
- [PUT > target > command for DB2 \(z/OS\) Adapter](#)
- [DB2 \(z/OS\) Adapter database execution command \(output source override\)](#)
- [PUT function](#)

PUT > target > command for DB2® (z/OS®) Adapter

Use the Map Designer or Integration Flow Designer to specify **Database** as the value for the PUT > Target setting and enter the adapter commands as desired for the **Command** setting.

The adapter commands are shown using a single quotation mark, which is the Windows syntax. For non-Windows platforms, use two single quotation marks followed by one single quotation mark and end with one single quotation mark followed by two single quotation marks.

Database command options when there is a database/query file	Database command options when there is no database/query file
<pre>''' [-DBTYPE DB2MVS] [-SOURCE datasource] [-TABLE table_name] [-CSTMT [number]] [-DELETE] [-UPDATE [OFF ONLY]] [{-TRACE [file ERROR]}]'''</pre>	<pre>''' -DBTYPE ODBC -TABLE table_name [-SOURCE datasource] [-CSTMT [number]] [-DELETE] [{-TRACE [file ERROR]}]'''</pre>

DB2® (z/OS®) Adapter database execution command (output source override)

Use the Output Source Override - Database execution command (-OD) to designate a database as a target or you can override one or more of the adapter command settings or database definitions in a RUN function or on the command line.

The adapter commands are shown using a single quotation mark, which is the Windows syntax. For non-Windows platforms, use two single quotation marks followed by one single quotation mark and end with one single quotation mark followed by two single quotation marks.

Scenario: compiled map target is a database

Database/query file (.mdq)	No database/query file (.mdq)
<pre>'[-MDQ DDNAME -DBNAME database_name] [-TABLE table_name] [-CCARD -CSTMT [number]] [-DELETE] [-UPDATE [OFF ONLY]] [{-TRACE [file ERROR]}]'</pre>	<pre>'-DBTYPE ODBC -TABLE table_name [-SOURCE datasource] [-CCARD -CSTMT [number]] [-DELETE] [-UPDATE [OFF ONLY]] [{-TRACE [file ERROR]}]'</pre>

Scenario: compiled map target is not a database

Database/query file (.mdq)	No database/query file (.mdq)
<pre>'-MDQ DDNAME -DBNAME database_name -TABLE table_name [-CCARD -CSTMT [number]] [-DELETE] [-UPDATE [OFF ONLY]] [{-TRACE [file ERROR]}]'</pre>	<pre>'-DBTYPE ODBC -TABLE table_name [-SOURCE datasource] [-CCARD -CSTMT [number]] [-DELETE] [-UPDATE [OFF ONLY]] [{-TRACE [file ERROR]}]'</pre>

PUT function

Use the PUT function to pass data to the target adapter.

Using PUT with a database/query file	Using PUT without a database/query file
<pre>PUT (''DB'', ''-MDQ DDNAME -DBNAME database_name -TABLE table_name [-CCARD -CSTMT [number]] [-DELETE] [-UPDATE [OFF ONLY]] [{-TRACE [file ERROR]}]''')</pre>	<pre>PUT (''DB'', ''-DBTYPE ODBC -TABLE table_name [-SOURCE datasource] [-CCARD -CSTMT [number]] [-DELETE] [{-TRACE [file ERROR]}]'''')</pre>

DB2® (z/OS®) Adapter limitations

The Database Interface Designer and adapters offer options and functions for accessing and manipulating data contained within a database. However, there are some restrictions and limitations of certain functions:

- **Stored Procedures**

Using stored procedures to access adapter commands and return values from stored functions is not supported.

- **Bind Facility**

Using bind values in database functions is not supported.

- **Database Triggers**

Using a data source as an input event trigger for the Launcher is not supported.

Return codes and error messages

Return codes and messages are returned when the particular activity completes. Return codes and messages can also be recorded as specified in the audit logs, trace files, execution summary files, and so forth.

For information about error codes and messages returned by database-specific adapters, see the Resource Adapters documentation.

Various troubleshooting tools are available in case you encounter problems while using the database adapters. For example, if you attempt to run a map that uses the adapter and encounter problems or do not receive the expected results, use the following adapter troubleshooting tools:

- adapter audit log (.log)
- adapter trace file (.mtr)

Extreme Scale adapter overview

IBM® Transformation Extender uses the Extreme Scale adapter to connect to an Extreme Scale server to access data or set data, or to run SQL queries to retrieve data. Extreme Scale can improve application performance by scaling processing to serve more users in less time.

Extreme Scale customers can access values from databases that are supported by Extreme Scale and can GET and PUT values that can be shared across multiple instances of IBM Transformation Extender.

- [System requirements](#)
- [Extreme Scale adapter command overview](#)

This table summarizes the adapter commands, syntax, and whether you can use the command with data sources, data targets, or both. You can specify the commands on the adapter command line.

System requirements

See the [system requirements](#) for details about minimum system requirements for IBM® Transformation Extender. See the Extreme Scale documentation for the minimum system requirements for Extreme Scale.

Extreme Scale adapter command overview

This table summarizes the adapter commands, syntax, and whether you can use the command with data sources, data targets, or both. You can specify the commands on the adapter command line.

The commands are not case-sensitive, but the command values are case-sensitive. Choices are separated by the OR symbol /. Optional parameters are enclosed in brackets []. The topics that follow provide the command syntax in railroad track format.

Table 1. Extreme Scale adapter command summary

Command	Syntax	Use with data source (GET)	Use with data target (PUT)
Connection string to connect to the Extreme Scale server	<code>{-CONNSTRING -C} connection_string</code>	Yes	Yes
Grid name	<code>{-GRID -G} grid_name</code>	Yes	Yes
Map that is called as the hash map (located within the grid)	<code>{-HASHMAP -M} map_name</code>	Yes	Yes
Key to look for	<code>{-KEY -K} key</code>	Yes	Yes
Reference of the complex object that is present in the IBM® Transformation Extender Object pool	<code>{-REFERENCE -R} object_reference</code>	No	Yes
Command to apply	<code>{-COMMAND -D} {UPDATE PUT INSERT REMOVE}</code>	No	Yes
Whether to apply a PUT command immediately, instead of waiting until the map completes	<code>{-NOW -N}{YES NO}</code>	No	Yes
SQL statement to retrieve values	<code>{-SQL -S} SQL_statement</code>	Yes	No
Trace	<code>-T [+][file_name]</code>	Yes	Yes
Trace errors	<code>-TE [+][file_name]</code>	Yes	Yes

- [Command aliases](#)

The WES adapter alias for the Extreme Scale adapter can be used on both input and output cards and in GET and PUT map rules.

- [Connection string \(-CONNSTRING\) command](#)

The **-CONNSTRING** adapter command specifies the connection string that is used to connect to the Extreme Scale server.

- [Grid \(-GRID\) command](#)

The **-GRID** adapter command specifies the name of the Extreme Scale Object Grid that the adapter connects to on the Extreme Scale server.

- [Hash map \(-HASHMAP\) command](#)

The **-HASHMAP** adapter command specifies the name of the Extreme Scale hash map to be used by the Extreme Scale server.

- [Key \(-KEY\) command](#)

The **-KEY** adapter command specifies the key to be sought for a GET or PUT to retrieve or set the value for the Extreme Scale hash map key.

- [Reference \(-REFERENCE\) command](#)

The **-REFERENCE** adapter command specifies the reference ID of the complex Java™ object that is in the IBM Transformation Extender Object Pool. When a map uses the JEXIT function to build a Java object, you can use this command to add the object to the Extreme Scale database.

- **Command (-COMMAND) adapter command**
The **-COMMAND** adapter command specifies the command to be run. If the command specified is a PUT, this command can be used with the **-NOW** command. The **-NOW** command indicates that the command specified is to be run immediately, instead of when the map completes.
 - **Now (-NOW) command**
Use the **-NOW** adapter command with the PUT function of the **-COMMAND** adapter command to indicate that the command specified is to be run immediately, instead of when the map completes.
 - **SQL statement (-SQL) command**
The **-SQL** adapter command specifies the SQL statement to be used to retrieve values with a GET function.
 - **Create trace file (-T) command**
The **-T** command produces an adapter trace log file. When you use the **-T** command at the command property card level or within a PUT or GET map rule, it produces the file `m4exscale.log`. The information within the file contains connection information and information on how the map performed.
 - **Trace adapter errors (-TE) command**
The **-TE** command produces a trace file that contains only the adapter errors that occurred during map execution.
-

Command aliases

The `WES` adapter alias for the Extreme Scale adapter can be used on both input and output cards and in GET and PUT map rules.

Specify adapter overrides by using a command string on the command line or in a map rule (either a PUT or a GET). The execution command syntax is as follows:

```
-IA[alias]card_num
-OA[alias]card_num
```

where:

-IA
The Input Source Override execution command.

-OA
The Output Target Override execution command.

alias
The adapter alias `WES`.

card_num
The number of the map card.

To override an input card 1:

```
-IAWES1 'adapter_commands'
```

To override an output card 1:

```
-OAWES1 'adapter_commands'
```

Connection string (-CONNSTRING) command

The **-CONNSTRING** adapter command specifies the connection string that is used to connect to the Extreme Scale server.

This command is required on input cards or GET commands when the Extreme Scale adapter connects to an Extreme Scale server to access data. This command is required on output cards or PUT commands when the Extreme Scale adapter connects to an Extreme Scale server to set data. The connection string argument is used to connect to the Extreme Scale Server. *hostname* is either the loop-back host name or the host name of the operating system that the user is logged in to.

-CONNSTRING command syntax



hostname:port
The connection string for the Extreme Scale server.

Grid (-GRID) command

The **-GRID** adapter command specifies the name of the Extreme Scale Object Grid that the adapter connects to on the Extreme Scale server.

This command is required on input cards or GET commands when the Extreme Scale adapter connects to an Extreme Scale server to access data. This command is required on output cards or PUT commands when the Extreme Scale adapter connects to an Extreme Scale server to set data.

-GRID command syntax



grid_name
The name of the Extreme Scale Object Grid that the adapter connects to on the Extreme Scale server.

Hash map (-HASHMAP) command

The **-HASHMAP** adapter command specifies the name of the Extreme Scale hash map to be used by the Extreme Scale server.

This command is required on input cards or GET commands when the Extreme Scale adapter connects to an Extreme Scale server to access data. This command is required on output cards or PUT commands when the Extreme Scale adapter connects to an Extreme Scale server to set data.

-HASHMAP command syntax

```
▶ [-HASHMAP [hashmap_name] -M]
```

hashmap_name

The name of the Extreme Scale hash map to be used.

Key (-KEY) command

The **-KEY** adapter command specifies the key to be sought for a GET or PUT to retrieve or set the value for the Extreme Scale hash map key.

This command is required on input cards or GET commands when the Extreme Scale adapter connects to an Extreme Scale server to access data. This command is required on output cards or PUT commands when the Extreme Scale adapter connects to an Extreme Scale server to set data.

-KEY command syntax

```
▶ [-KEY [key] -K]
```

key

The key to be sought for a GET or PUT to retrieve or set the value for the Extreme Scale hash map key. This key is a text string.

Reference (-REFERENCE) command

The **-REFERENCE** adapter command specifies the reference ID of the complex Java™ object that is in the IBM® Transformation Extender Object Pool. When a map uses the JEXIT function to build a Java object, you can use this command to add the object to the Extreme Scale database.

This command is not applicable on input cards or GET commands. This command is optional on output cards (PUT commands) when the Extreme Scale adapter connects to an Extreme Scale server to set data.

-REFERENCE command syntax

```
▶ [-REFERENCE [reference_ID] -R]
```

reference_ID

The reference ID of the complex object in the IBM Transformation Extender Object Pool. When a map uses the JEXIT function to build a Java object, you can use this command to add the object to the Extreme Scale database.

Command (-COMMAND) adapter command

The **-COMMAND** adapter command specifies the command to be run. If the command specified is a PUT, this command can be used with the **-NOW** command. The **-NOW** command indicates that the command specified is to be run immediately, instead of when the map completes.

This command is not applicable on input cards. This command is optional on output cards or PUT commands when the Extreme Scale adapter connects to an Extreme Scale server to set data.

-COMMAND command syntax

```
▶ [-COMMAND [UPDATE | PUT | INSERT | REMOVE] -D]
```

UPDATE

Update the referenced data on the Extreme Scale server.

PUT

Put (set) the referenced data to the Extreme Scale server. Optionally, specify the **-NOW** command to indicate that the PUT function is to process immediately rather than when the map completes.

INSERT

Insert the referenced data to the Extreme Scale server.

REMOVE

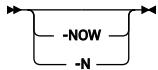
Remove the referenced data from the Extreme Scale server.

Now (-NOW) command

Use the **-NOW** adapter command with the PUT function of the **-COMMAND** adapter command to indicate that the command specified is to be run immediately, instead of when the map completes.

This command is not applicable on input cards or GET commands when the Extreme Scale adapter connects to an Extreme Scale server. This command is optional on output cards or PUT commands when the Extreme Scale adapter connects to an Extreme Scale server to set data.

-NOW command syntax

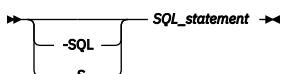


SQL statement (-SQL) command

The **-SQL** adapter command specifies the SQL statement to be used to retrieve values with a GET function.

This command is optional on input cards or GET commands when the Extreme Scale adapter connects to an Extreme Scale server to GET data. This command is not applicable on output cards or PUT commands when the Extreme Scale adapter connects to an Extreme Scale server.

-SQL command syntax



SQL_statement

The SQL statement that is used with a GET command to retrieve values from the Extreme Scale server.

Create trace file (-T) command

The **-T** command produces an adapter trace log file. When you use the **-T** command at the command property card level or within a PUT or GET map rule, it produces the file `m4exscale.log`. The information within the file contains connection information and information on how the map performed.

This command is optional.

-T command syntax



+

Appends trace information to the existing trace file.

`m4exscale.log`

The default trace file that is created.

`file_name`

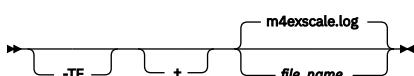
The name and path of the trace file. This parameter is optional. If you omit it, the adapter creates a trace file that is named `m4exscale.log` in the map source directory.

Trace adapter errors (-TE) command

The **-TE** command produces a trace file that contains only the adapter errors that occurred during map execution.

This command is optional.

-TE command syntax



+

Appends trace adapter error information to the `m4exscale.log` adapter trace file.

`file_name`

The name and path of the trace file. This parameter is optional. If you omit it, the adapter appends the error information to a trace file that is named m4exscale.log in the map source directory.

Overview of email adapters

An email adapter automatically receives email messages as sources of a map or sends email messages as targets of a map.

Using email adapters you can:

- Send sales orders to a remote manufacturing facility
- Retrieve notifications of error messages for technical support activities
- Pass notification of a contract price to a component

IBM Transformation Extender supports the following email transport protocols:

- Notes® API (Lotus Notes®)
- Internet email (SMTP/POP3)
- [System requirements](#)
- [Command alias](#)
- [Using adapter commands](#)
- [Commands for email adapters](#)
- [Syntax summary](#)
- [Troubleshooting email adapters](#)
- [Return codes and error messages](#)

System requirements

The computer where the email adapter is installed must be able to access the email servers over a network. It is assumed that a Command Server has already been installed on the computer where the adapter is to be installed for runtime purposes. See the [system requirements](#) and the [release notes](#) for additional details.

Command alias

Specify adapter commands by using a command string on the command line or by creating a command file that contains adapter commands. The command syntax is:

`-IA[alias]card_num`
`-OA[alias]card_num`

In the command syntax, `-IA` is the Input Source Override execution command and `-OA` is the Output Target Override execution command, `alias` is the adapter alias, and `card_num` is the number of the map card. The email adapter alias and corresponding execution commands are listed below.

Adapter	Alias	As Input	As Output
email	EMAIL	<code>-IAEMAILcard_num</code>	<code>-OEMAILcard_num</code>

Using adapter commands

Use an email adapter either as the data source of an input map card, as the data target for an output map card, or both. You can also use an email adapter on the command line or in a map rule using the GET, PUT, or RUN functions.

The IBM Transformation Extender email adapters support email transport protocols for several popular email applications. Use the required Protocol command (`-PROTO`) in the `GET > Source > Command` or `PUT > Target > Command` settings to specify the email transport protocol.

See [Syntax Summary](#) for adapter command syntax.

Commands for email adapters

The following table lists valid commands for the email adapters, the command syntax, and whether the command is supported (✓) for use with data sources, targets, or both.

Name	Syntax	Source	Target
Attachment (-ATT or -ATTACH)	<code>-ATTACH filename</code>	✓	✓
Attachment Name (-AN or -ATTNAME)	<code>-ATTNAME fieldname</code>	✓	✓
Audit (-A or -AUDIT)	<code>-AUDIT[+]S][full_path]</code>	✓	✓
BCC Recipient (-BCC)	<code>-BCC recipient</code>		✓
CC Recipient (-CC)	<code>-CC recipient</code>		✓
From (-FROM) Note: Supported by the internet email transport protocol only	<code>-FROM sender</code>		✓
Header (-HDR)	<code>-HDR[+]</code>	✓	✓

Name	Syntax	Source	Target
Login (-L or -LOGIN) Note: Supported by the internet email transport protocol only	-LOGIN {APOP USERPASS}	✓	
Listen (-LSN)	-LSN seconds	✓	
Markread (-MARKREAD) Note: Supported by the Lotus Notes® email transport protocol only	-MARKREAD		✓
Password (-P or -PASS)	-PASS <i>password</i>	✓	✓
Port (-PORT)	-PORT <i>port_number</i>	✓	✓
Post Office (-PO) Note: Supported by the Lotus Notes email transport protocol only	-PO <i>post_office</i>	✓	✓
Priority (-PRI or -PRIORITY)	-PRIORITY <i>level</i>	✓	✓
Protocol (-PR or -PROTO)	-PROTO {INET LNOTES}	✓	✓
Raw (-RAW)	-RAW	✓	✓
Note: Supported by the internet email transport protocol only			
Receipt (-RCPT or -RECEIPT)	-RECEIPT		✓
Server (-SVR or -SERVER) Note: Supported by the internet email transport protocol only	-SERVER <i>host_name</i>	✓	✓
SSL Protocol (-SPROTO)	-SPROTO {SSLv2 SSLv3 SSLv23 TLSv1 TLSv11 TLSv12}	✓	✓
SSL Encryption Strength (-STR)	-STR {WEAK STRONG ANY}	✓	✓
Subject (-SUB or -SUBJECT)	-SUBJECT <i>subject</i>		✓
Text (-TXT or -TEXT)	-TEXT <i>string</i>	✓	✓
To Recipient (-TO)	-TO <i>recipient</i>		✓
Trace (-T or -TRACE)	-T[E][+][S V] [<i>full_path</i>]	✓	✓
Type (-TYPE) Note: Supported by the internet email transport protocol only	-TYPE <i>content_type</i>		✓
Userid (-UID or -USERID) Note: Supported by the Lotus Notes email transport protocol only	-USERID <i>user_id</i>	✓	✓
Username (-U or -USER)	-USER <i>user_name</i>	✓	✓

- [Attachment \(-ATT or -ATTACH\)](#)
- [Attachment name \(-AN or -ATTNAME\)](#)
- [Audit \(-A or -AUDIT\)](#)
- [BCC recipient \(-BCC\)](#)
- [CC recipient \(-CC\)](#)
- [From \(-FROM\)](#)
- [Header \(-HDR\)](#)
- [Login \(-L or -LOGIN\)](#)
- [Listen \(-LSN\)](#)
- [Markread \(-MARKREAD\)](#)
- [Password \(-P or -PASS\)](#)
- [Port \(-PORT\)](#)
- [Post office \(-PO\)](#)
- [Priority \(-PRI or -PRIORITY\)](#)
- [Protocol \(-PR or -PROTO\)](#)
- [Raw \(-RAW\)](#)
- [Receipt \(-RCPT or -RECEIPT\)](#)
- [Server \(-SVR or -SERVER\)](#)
- [SSL protocol \(-SPROTO\)](#)
- [SSL encryption strength \(-STR\)](#)
- [Subject \(-SUB or -SUBJECT\)](#)
- [Text \(-TXT or -TEXT\)](#)
- [To recipient \(-TO\)](#)
- [Trace \(-T or -TRACE\)](#)
- [Type \(-TYPE\)](#)
- [Userid \(-UID or -USERID\)](#)
- [Username \(-U or -USER\)](#)

Attachment (-ATT or -ATTACH)

The attachment adapter command (-ATT or -ATTACH) processes message attachments.

When used for a data source, you can read attachments (instead of the note text) by omitting the filename.

When used for a data target, this command changes the location of data supplied through the map from the body of the email to a file attachment when you specify the filename. For example, if the map rule is the text literal Hello World, the body of the email contains "Hello World". If you then enter -att sample.txt for the Target, PUT...>Target... Command in the output card, the text literal Hello World will appear in the attachment as a text file with the name sample.txt; the body of the email will be empty.

-ATTACH *filename*

Option

Description

filename

For data targets only, supply the name of the attachment file. For data sources, omit this option.

To generate the name portion of the attachment's filename using the adapter, supply a filename in extension-only format (for example, .ext).

- [Multiple attachments](#)

Multiple attachments

The **-ATTACH** command only supports a single attachment. If you are using the internet email transfer protocol (**-PROTO INET**) and multiple attachments need to be sent or received, you can accomplish this by chaining the MIME adapter with the email adapter, and using the **-RAW** option of the email adapter to map the MIME header fields.

An example of a map that sends and receives multiple attachments is provided in the map source file *emailatt.mms* in the *install_dir\examples\adapters\mime* directory.

Attachment name (-AN or -ATTNAME)

The Attachment Name adapter command (**-AN** or **-ATTNAME**) specifies that the name of the attachment should be returned together with the attachment itself. This command can only be used on input. It results in the name of the attachment being returned as a header to the data.

-ATTNAME *fieldname*

Option	Description
<i>fieldname</i>	Specify the name of the header field that should be returned in the header.

For example, to specify **myfield** as the name of the header field that should be returned in the header:

-ATTNAME *myfield*

When used with the **-HDR** option, the attachment name becomes another header field. For example:

```
From: Mr. X <mrX@mail.net>
Subject: Attachment test
Date: Wed, 3 Apr 2002 10:13:36 -0500
To: mrsX@mail.net
X-Attachment-Name: myday.txt
Breakfast
Kids to school
Go to work
```

The option takes an optional parameter that is the name of the attachment name field. This is provided for two reasons:

- To offer a solution in the event that **X-Attachment-Name** appears in the data, which would cause distinguish ability problems during validation
- To provide a means to offer a user-defined field name that might be more suited to the context of its use than the default name

Audit (-A or -AUDIT)

Use the Audit adapter command (**-A** or **-AUDIT**) to produce a log file in the map directory where the adapters are installed with the default name *m4email.log*. Audit information is recorded detailing files sent or received from each host, elapsed time, file size, and retry count for each file.

-AUDIT[+] [S] [full_path]

Option	Description
+	Append audit information to the existing log file. If a log file does not exist, it is created.
S	Summary mode. Record only minimal information in the log file.
full_path	Create audit file with the specified name in the specified directory. (By default, the directory is where the map is located and the file name is <i>m4email.log</i> .)

BCC recipient (-BCC)

The BCC Recipient adapter command (**-BCC**) identifies a blind courtesy copy (BCC) recipient, but is otherwise identical to the To Recipient adapter command (**-TO**).

-BCC *recipient*

Option	Description
<i>recipient</i>	Specify the BCC recipient.

CC recipient (-CC)

The CC Recipient adapter command (**-CC**) identifies a courtesy copy (CC) recipient, but is otherwise identical to the To Recipient adapter command (**-TO**).

-CC recipient

Option

Description

recipient

Specify the CC recipient.

From (-FROM)

The From adapter command (**-FROM**) uses the specified email address as the sender of the generated message.

-FROM sender

Option

Description

sender

Supply the email address of the sender for outbound messages.

Note: This command is only supported by the internet email transport protocol and is required. You must also specify at least one of the following commands:

- [To recipient \(-TO\)](#)
- [CC recipient \(-CC\)](#)
- [BCC recipient \(-BCC\)](#)

Header (-HDR)

When used on input, the Header command option (**-HDR**) of the email adapter retrieves the message header in the following format:

```
From: originator[originator_address]
Subject: subject text
Date: yyyy/mm/dd hh: mm
To: recipient[recipient_address]; ....;
recipient[recipient_address]
cc: recipient[recipient_address]; ....;
recipient[recipient_address]
bcc: recipient[recipient_address]; ....;
recipient[recipient_address]
```

-HDR [+]

Option

Description

+

Append user-supplied headers

On output, the (**-HDR+**) option can be used to append the user-supplied header fields to the message header. Each header field should be terminated by a CR/LF and an empty line (CR/LF only) delimits the headers from the data.

Note: The (**-HDR+**) option is only compatible with internet email transport protocol (**-PROTO INET**) on output.

For example, use the **-HDR+** command to append the user-supplied header fields to include:

```
X-MyfirstHeader:John
X-MysecondHeader:Paul
X-MythirdHeader:Michael
This is my message.
```

Login (-L or -LOGIN)

The Login adapter command (**-L** or **-LOGIN**) is used for data sources to supply the method(s) used to gain access to the mailbox. When a Login command is not provided, the default setting is **USERPASS**. Consult your System Administrator if you are unsure of the method to use. Generally, the method used is **USERPASS**.

Note: This command is only supported by the internet email transport protocol.

-LOGIN {APOP|USERPASS}

Option

Description

APOP

Sends the APOP command as described in RFC 1939 (<http://www.ietf.org/rfc/rfc1939.txt>).

USERPASS

Sends a plain-text user name and password combination

Listen (-LSN)

Use the listen (**-LSN**) command to specify a time (in seconds) to wait for messages to arrive. If no email is available in the specified listen time, you will receive an error message of **Source not available**.

The command `-LSN 0` means do not wait at all. If no message is immediately available, the adapter returns a warning indicating that no messages were found.

-LSN seconds

Option

Description

seconds

The number of seconds to wait for messages to arrive

Markread (-MARKREAD)

Use the Markread adapter command (`-MARKREAD`) to specify that the messages the adapter is sending have been read. They will be marked as read in the sender's **Sent** folder. This command is not required.

Note: This command is only supported by the Lotus Notes® email transport protocol.

Note: The `-MARKREAD` command can only be used when the `-PROTO` command is specified on the adapter command line as well (used with the `LNOTES` command option of the Protocol adapter command).

-MARKREAD

Password (-P or -PASS)

Use the Password adapter command (`-P` or `-PASS`) to gain authorized access to mailboxes that use security passwords. This command is required unless you are using a mailbox that does not require you to specify the user ID and password to connect to the mailbox.

-PASS password

Option

Description

password

When used for a data source, supply the case-sensitive password required for entry to the source mailbox.

When used for a data target, supply the case-sensitive password required for the target mailbox.

Port (-PORT)

Use the Port adapter command (`-PORT`) to specify the port number used for SMTP and POP3 on exchange server. The `-PORT` accepts integer value.

-PORT port_number

Option

Description

port_number

If `-PORT` is used for SMTP or POP3 communication. Default for SMTP is 465 and 995 for POP3 if not specified. SMTP protocol is used to send a mail and POP3 is used to receive a mail.

Post office (-PO)

Use the Post Office adapter command (`-PO`) to gain authorized access to mailboxes. The Post Office adapter command is required.

-PO post_office

Option

Description

post_office

Required value. When used for a data source, supply the post office (case sensitive) required for entry to source mailbox.

When used for a data target, supply the post office (case sensitive) required for the target mailbox.

Note: The Post Office adapter command (`-PO`) is supported by the Lotus Notes® (`LNOTES`) email transport protocol only.

See [Userid \(-UID or -USERID\)](#) for its use with `LNOTES`.

Priority (-PRI or -PRIORITY)

Use the Priority adapter command (`-PRI` or `-PRIORITY`) to set the priority of the message.

-PRIORITY level

Option

Description

level

Required value. Specify the level of priority:

1. -low or -lo
2. -normal or -nrm
3. -high or -hi

Note: This command is only supported by Lotus Notes® and internet email transport protocols.

Protocol (-PR or -PROTO)

Use the Protocol adapter command (-PR or -PROTO) to specify the email transport protocol.

-PROTO {INET|LNOTES}

Option

Description
INET
Specifies the SMTP email transport protocol for data targets and POP3 for data sources
LNOTES
Specifies the Lotus Notes® email transport protocol

Raw (-RAW)

Use the Raw (-RAW) command in an input card to dump out the received message without interpreting or processing the header fields. Use it on an output card if the data specified contains the header data.

The type tree RawInet.mtt in the examples directory (*install_dir\examples\adapters\email*) can be used to create the header data.

-RAW

The -RAW option is only compatible with internet email transport protocol (-PROTO INET). If used with another protocol, it will be flagged as an error.

Similarly, the -RAW command is incompatible with all other header options:

- -ATTACH
- -BCC
- -CC
- -HDR
- -PRIORITY
- -RECEIPT
- -SUBJECT
- -TEXT
- -TYPE

An error will be flagged if any of these options are used when -RAW is specified.

Note: This command is only supported by the internet email transport protocol.

Receipt (-RCPT or -RECEIPT)

Use the Receipt adapter command (-RCPT or -RECEIPT) to request a return receipt. Valid with data targets only.

-RECEIPT

Server (-SVR or -SERVER)

Use the Server adapter command (-SVR or -SERVER) to identify the host server where the mailbox can be accessed or addressed.

-SERVER host_name

Option

Description
<i>host_name</i> For data sources, supply the internet host name where the mailbox can be accessed. For data targets, supply the internet host name that will forward the outgoing message.

Note: This command is only supported by the internet email transport protocol.

SSL protocol (-SPROTO)

Use the Secure Socket Layer (SSL) protocol adapter command (-SPROTO) to set the SSL protocol level.

This command decides whether to use Secure channel to communicate with the Email server. If `-SPROTO` is not specified, it will be an unsecured communication channel between the map and the Email server.

This command is applicable only when the SSL `secure_mode` setting in the config.yaml configuration file (/runtime/secure_mode) is set to 0 and RSA-based SSL encryption is in effect. This command is superseded by an SSL `secure_mode` setting of 1 or 2.

```
-SPROTO {SSLv2|SSLv3|SSLv23|TLSv1|TLSv11|TLSv12}
```

Option

Option	Description
SSLv2	Specify SSL protocol version 2
SSLv3	Specify SSL protocol version 3
SSLv23	Specify SSL protocol version 2, version 3, TLSv1, TLSv1.1 or TLSv1.2 Whenever the protocol method of <code>SSLv23</code> is specified, the protocol of first choice is <code>TLSv1.2</code> , with fallback to <code>TLSv1.1</code> , then to <code>TLSv1</code> , then to <code>SSLv3</code> , and finally to <code>SSLv2</code> , depending on the loaded GSKit module.
TLSv1	Specify Transport Layer Security (TLS) protocol version 1
TLSv11	Specify Transport Layer Security (TLS) protocol version 1.1
TLSv12	Specify Transport Layer Security (TLS) protocol version 1.2

SSL encryption strength (-STR)

Use the Secure Socket Layer (SSL) encryption strength adapter command (`-STR`) to specify the encryption strength.

The SSL encryption strength adapter command (`-STR`) can be either `WEAK`, only using weak encryption algorithms, or `STRONG`, only using strong encryption algorithms. If not specified, any available encryption algorithm is used.

If `STRONG` encryption algorithm is not available, then `WEAK` encryption algorithm is used. If the export version of the libraries is used, only weak encryption algorithms are available, and this command is ignored.

This command is applicable only when the SSL `secure_mode` setting in the config.yaml configuration file (/runtime/secure_mode) is set to 0 and RSA-based SSL encryption is in effect.

```
-STR {WEAK|STRONG|ANY}
```

Option

Option	Description
WEAK	Use exportable encryption algorithm only
STRONG	Use strong (non-exportable) encryption algorithm only
ANY	Use any available encryption algorithm

Subject (-SUB or -SUBJECT)

The Subject adapter command (`-SUB` or `-SUBJECT`) uses the specified subject text as the subject of the generated message. If no text is specified, the subject field contains `Message Produced by IBM`.

```
-SUBJECT "subject"
```

Option

Option	Description
<code>subject</code>	Supply the text to be placed in the subject field for outbound messages.

Text (-TXT or -TEXT)

Use the Text adapter command (`-TXT` or `-TEXT`) to process the text part(s) of the message.

When used for a data source, it specifies that text parts should be retrieved in addition to attachments if the Attachment adapter command (`-ATTACH`) is specified. In this case, omit `string` and the flag is ignored if `-ATTACH` has not been specified.

When used for a data target, the Text adapter command (`-TEXT`) causes the specified `string` to be sent as the text part of the message. In this case, `string` is required, and if `-ATTACH` has not been specified, the text is prefixed to the data.

```
-TEXT string
```

Option

Description
<i>string</i>
Required value when used for a data target. Specify the string to be sent as the text part of the outbound message. When used for a data source, this option is not required.

To recipient (-TO)

Use the To Recipient adapter command (-TO) to identify a recipient of the message.

-TO *recipient*

Option

Description
<i>recipient</i>

Required value. Supply the name of the recipient.

Note: If you use the INET email transport protocol, only the SMTP type of addressing is applicable for a *recipient* name.

Trace (-T or -TRACE)

Use the Trace adapter command (-T or -TRACE) to produce a diagnostics file that contains detailed information about email adapter activities.

The default filename is m4email.mtr and it is created in the map directory unless otherwise specified.

-T [E] [+|S|V] [*full_path*]

Option

Description
E
Produce a trace file containing only the adapter errors that occurred during map execution.
+
Append trace information to the existing log file.
S
Summary mode. Record only minimal information in the log file. Default value.
V
Verbose mode. Record in the log file all activity occurring while the adapter is retrieving data. If not specified, summary mode is assumed.
<i>full_path</i>
Create a trace file with the specified name in the specified directory. The default directory is the map directory and the default filename is m4email.mtr.

Note: You can override the adapter command line trace options dynamically using the Management Console. See "Dynamic Adapter Tracing" in the *Launcher* documentation for detailed information.

Note: Because summary mode is the default, using -TRACES provides the same output as -TRACE.

Type (-TYPE)

Use the Type adapter command (-TYPE) to specify the MIME content-type for the attachment used in the adapter request. The default content-type is **application/octet-stream**.

-TYPE *content_type*

Option

Description
<i>content_type</i>

Specify the MIME content-type for the attachment used in the adapter request. You can use other types, but do not use compound types such as multipart.

Note: This command can only be used with the -ATTACH command in an output card. In addition, this command is only supported by the internet email transport protocol.

Userid (-UID or -USERID)

Use the Userid adapter command (-UID or -USERID) to specify the user ID file required when you use the LNOTES command option.

-UID *user_id*

Option

Description
<i>user_id</i>

Specify a valid user id file of the specified post office.

Note: This adapter command is required only when the Post Office adapter command (-PO) is used and the Protocol adapter command (-PROTO) is specified with the LNOTES command option.

Username (-U or -USER)

Use the Username adapter command (-U or -USER) to identify the user name associated with the account being accessed. You can omit this command if mailbox session is currently active.

-USER user_name

Option

Description

user_name

Specify a valid user name for the email account being accessed.

When used for a data target and this command is omitted, the default profile is used to log on to the email system. You must be logged-on for the following scenarios:

- A default profile has not been defined.
- The messaging system does not support default profiles.
- -U is not used to specify a profile.

The profile requires a password and the -P command is not used to specify the password.

Syntax summary

This documentation provides syntax summaries for each of the email adapter protocol options that support Lotus Notes® (LNOTES) and internet email (INET) transport protocols.

- [Lotus Notes adapter commands](#)
- [Internet email adapter commands](#)
- [Email adapters example using data sources and targets](#)
- [Example: send mail using the Lotus Notes protocol](#)
- [Example: send mail using internet email protocol](#)
- [Command line example: Lotus Notes override](#)
- [Command line example: internet email override](#)

Lotus Notes adapter commands

The following command syntax summaries are for the email adapter protocol option that supports the Lotus Notes® (LNOTES) email transport protocol.

- [Data sources for LN NOTES](#)
- [Data targets for LN NOTES](#)

Data sources for LN NOTES

The following command syntax summary is for the email adapter protocol LN NOTES option used for data sources.

```
-PROTO LN NOTES
[-PASS password]
[-PO post_office]
[-ATTACH filename]
[-TEXT string]
[-PRIORITY level]
[-HDR]
[-LSN seconds]
[-T[E] [+][S|V] [full_path]]
[-AUDIT[+][S] [full_path]]
[-USER user_name]
[-USERID user_id]
```

Data targets for LN NOTES

The following command syntax summary is for the email adapter protocol LN NOTES option used for data targets:

```
-PROTO LN NOTES
-TO recipient|-CC recipient|-BCC recipient
[-PASS password]
[-PO post_office]
[-SUBJECT subject]
[-PRIORITY level]
[-RECEIPT]
[-ATTACH filename]
[-TEXT string]
[-T[E] [+][S|V] [full_path]]
[-AUDIT[+][S] [full_path]]
[-MARKREAD]
```

```
[-USER user_name]
[-USERID user_id]
```

You must specify at least one of the following commands, in addition to the Protocol (-PROTO) command:

- To Recipient (-TO *recipient*)
- CC Recipient (-CC *recipient*)
- BCC Recipient (-BCC *recipient*)

Internet email adapter commands

The following command syntax summaries are for the email adapter protocol option that supports the internet email transport protocol.

- [Data sources for INET](#)
- [Data targets for INET](#)

Data sources for INET

The following command syntax summary is for the email adapter protocol INET option used for data sources:

```
-PROTO INET
-SERVER host_name [-LOGIN {APOP|USERPASS}]
[-USER user_name]
[-PASS password]
[-ATTACH]
[-PRIORITY level]
[-TEXT string]
[-RAW]
[-HDR]
[-LSN seconds]
[-T[E] [+][S|V] [full_path]]
[-AUDIT[+][S] [full_path]]
```

Data targets for INET

The following command syntax summary is for the email adapter protocol INET option used for data targets:

```
-PROTO INET
-FROM sender
-TO recipient|-CC recipient|-BCC recipient
[-SERVER host_name]
[-PASS password]
[-SUBJECT subject]
[-PRIORITY level]
[-RECEIPT]
[-ATTACH filename]
[-RAW]
[-TEXT string]
[-T[E] [+][S|V] [full_path]]
[-AUDIT[+][S] [full_path]]
```

You must specify at least one of the following commands, in addition to the Protocol (-PROTO) and From (-From) commands:

- To Recipient (-TO *recipient*)
- CC Recipient (-CC *recipient*)
- BCC Recipient (-BCC *recipient*)

Email adapters example using data sources and targets

```
= RUN("XExLogEM.mmc", " -WD -AE=" + GETDIRECTORY( AuditLog ) +
"XExLogEM.log" + " -IFI " + GETFILENAME(AuditLog) + " -
OAEMAIL1 -T -PR "+ Protocol Field:EmailOptionRec + " -U "" +
User Field:EmailOptionRec + " " +
IF(PRESENT(#01 Destination Field:EmailOptionRec), "-TO "" + #01
Destination Field:EmailOptionRec + "", NONE) +
IF(PRESENT(#02 Destination Field:EmailOptionRec), " -TO "" +
#02 Destination Field:EmailOptionRec + "", NONE) +
IF(PRESENT(#03 Destination Field:EmailOptionRec), " -TO "" +
#03 Destination Field:EmailOptionRec + "", NONE) +
" -CC "squentin" -SUB "" + Subject Field:EmailOptionRec + " " +
" -TEXT ""+ Text Field:EmailOptionRec + " " +
" -ATTACH "" + Attachment Field:EmailOptionRec + " " ` " )
```

Example: send mail using the Lotus Notes® protocol

```
-PR LNOTES -MARKREAD -TO name -PASS password  
-CC recipient -PASS password -SUBJECT subject  
-TEXT string -RECEIPT
```

Example: send mail using internet email protocol

```
-PR inet -TO name -FROM name -ATT filename -T
```

Command line example: Lotus Notes override

To override output map card 3 using the email adapter for a Lotus Notes® email system on a Windows platform, send an attachment to a user, and generate a new trace file:

```
-OAEMAIL3 ` -PR lnotes -TO name -P lymdear4 -ATT filename -T`
```

Command line example: internet email override

To override output map card 4 using an adapter for internet email systems on a Windows operating system, send an attachment, and generate a trace file:

```
-OAEMAIL4 ` -PR inet -TO name -FROM name -ATT filename -T`
```

Troubleshooting email adapters

For information about error codes and messages returned by the adapters, see [Return Codes and Error Messages](#).

Various troubleshooting tools are available when you encounter problems using email adapters as data sources or targets for a map. If you attempt to run a map that uses an email adapter and receive a runtime error or do not get the expected output, use any or all of the following troubleshooting tools:

- View the Trace log file (.mtr).
 - View source and target data.
 - [Trace log file](#)
 - [Internet email](#)
 - [Lotus Notes](#)
-

Trace log file

The adapter trace log file contains detailed information provided by the adapter, recording the actions that are taking place such as connections that are established and statements that are executed. The trace log file is produced during the adapter execution and can be used as a debugging aid. To produce a trace file for the adapter, use the **-TRACE** command. The default name for the log is m4email.mtr which is located in the same directory as the compiled map file. You can specify an alternate filename and/or location using the **-TRACE** command.

Internet email

- The 32-bit version of internet email (wsock32.dll, normally installed as part of the Windows operating system) must be installed on the same computer where the adapters are to be installed.)
- Windows or BSD (UNIX) Sockets must be installed on the client computer. The Sockets installation must support TCP/IP.
- The client computer must be able to access an appropriate server. Test this as follows:
 - telnet target-ip 25 (to test a SMTP email target)
 - telnet target-ip 110 (to test a POP3 email source)

where **target-ip** is the IP address of the server. If the test is successful, you will receive a welcome message from the server. To close the session, enter **QUIT** (all caps) and press **Enter**.

Note: You might not be able to see what you are typing on the screen.

Lotus Notes

This section presents troubleshooting information for the email adapter protocol options that support the Lotus Notes® (LNOTES) email transport protocols.)

- The specific version of the Lotus Notes transfer protocol APIs must be installed on the same computer where the adapters are to be installed.
 - For **LNOTES** protocol command option, it is the 32-bit version of the Notes® API, which is nnotes.dll and its associated software.
- Notes must be in the PATH. When using the Launcher, only the System PATH is available. The User PATH is never accessible if the server runs under the appropriate user ID. You might have to manually modify those settings. You might have to copy the Notes PATH settings from the user ID PATH variable to the system PATH

variable. Do not change the user ID for the Launcher; even if running as a specified user ID, user environment variables are not available. You must restart the system so that changes to system environment variables are available to a service.

Note: If there is an active Notes session running on the same computer, errors can occur in the adapter and display the message: Could not open ID file. Close the Notes session to resolve this issue.

Return codes and error messages

Return codes and messages are returned when the particular activity completes. Return codes and messages can also be recorded as specified in the audit logs, trace files, execution summary files, and so on.

- [Messages](#)

Messages

The following is a listing of all the codes and messages that can be returned as a result of using the email adapter for sources or targets.

Note: Adapter return codes with positive numbers are warning codes that indicate a successful operation. Adapter return codes with negative numbers are error codes that indicate a failed operation.

Return Code	Message
0	OK
-1	Insufficient memory to continue
-1	Library Initialization Failed
-2	Invalid Entry Point. (AIX® adapters only)
-3LNOTES	Could not load adapter
-3INET	Could not load adapter
-3	Error Sending Data
-4	Error Receiving Data
-5	LNOTES: Insufficient memory to continue
-5	INET: Insufficient memory to continue
0	No Messages
1	No data provided. Create on content specified: no data sent.
-1	Unsupported protocol
-1	Insufficient memory to continue
-1	Invalid protocol selection. (SMTP for Get/POP3 for Put)
-5	Memory Error
-600	Internal Error: Resource Manager Error
-?	Mail Error where -? is an API-specific error code, made negative if >0
-???	???? (SMTP only) where -??? is an error code returned from the server, made negative and where ??? is a server-supplied error message

Excel Adapter

The Excel Adapter reads data from and maps data to a sheet in a Microsoft Excel workbook. In Design Studio, use the Excel importer (File > Import > Excel) to create a type tree based on a specified worksheet in the Excel workbook.

Excel Adapter command aliases

Use **EXCEL** as the adapter command alias on input and output card overrides and in **GET** and **PUT** rules. For example:

Input source override execution command	-IAEXCEL card_num
Output target override execution command	-OAEXCEL card_num

Inbound adapter example

Command line: -W Sheet1 -T

GET rule: =PARSE(GET("EXCEL", "-W Sheet1 -T", Input))

Inbound mapping rules are likely to be based on row number and the **EXTRACT** function:

Figure 1. Inbound mapping rules based on row number and **EXTRACT** function

Figure 1. Inbound mapping rules based on row number and EXTRACT function	
Excel	
sequence	
◆ ID	=GETID(EXTRACT(Rows:ExcelIn, RowNumber Columns:Rows:ExcelIn = 2))
◆ NAME	=GETNAME(EXTRACT(Rows:ExcelIn, RowNumber Columns:Rows:ExcelIn = 2))
▶ Detail (1:s)	=GETDETAILS(EXTRACT(Rows:ExcelIn, RowNumber Columns:Rows:ExcelIn = 2))
	NAME:sequence:Excel:global:XMLOut

Outbound mapping rules can be index-based. RowNumber indicates where to insert the data.

Figure 2. Index-based outbound mapping rules using RowNumber

Figure 2. Index-based outbound mapping rules using RowNumber	
ExcelOut	
Rows [1]	
◆ RowNumber Columns	=2
◆ COL0 Columns (0:1)	="ID"
◆ COL1 Columns (0:1)	=ID:sequence:Excel:global:XMLIn
◆ COL2 Columns (0:1)	=NONE
◆ COL3 Columns (0:1)	="NAME"
◆ COL4 Columns (0:1)	=NAME:sequence:Excel:global:XMLIn
Rows [2]	
◆ RowNumber Columns	=4
◆ COL0 Columns (0:1)	="FULLNAME"
◆ COL1 Columns (0:1)	="TITLE"
◆ COL2 Columns (0:1)	="BIRTH"
◆ COL3 Columns (0:1)	="COUNTRY"
◆ COL4 Columns (0:1)	="ID"
Rows (s)	=FOREACHDETAIL(Detail:sequence:Excel:global:XMLIn, COL4 Columns:Rows[2]:ExcelOut)

- **System requirements**

See the [release notes](#) for the Excel Adapter requirements.

- **Excel Adapter commands**

This documentation describes the functions and use of the Excel Adapter commands.

System requirements

See the [release notes](#) for the Excel Adapter requirements.

Excel Adapter commands

This documentation describes the functions and use of the Excel Adapter commands.

- **Transaction scope and document persistence in memory**

When a document is added using the **-ADOC** command, the Excel Adapter stores the document in memory. Documents stored in memory are identified by a document identifier that is either provided by the user, or generated by the Excel Adapter. The document persists in memory as specified by the transaction scope in the card settings.

- **data_from_map option**

Some of the Excel Adapter commands that are described in this documentation include the **data_from_map** option. The **data_from_map** option represents the data that is passed as the third parameter of a **GET** function. It is not a parameter on the command line.

- **Add document (-ADOC) command**

The **-ADOC** command adds a file on the local file system, or a remote file accessible through **HTTP** protocol, as a document in memory, and returns the document ID to the map. Alternatively, the **-ADOC** command can add dynamic data from a map as a document in memory and return the document ID to the map.

- **Remove document (-RDOC) command**

The **-RDOC** command removes a document from memory. By default, documents are removed from memory when map, or card processing completes. Transaction scope card setting determines the document persistence in memory. Use this command if too many intermediate documents accumulate in memory during map processing.

- **Replace document (-UDOC) command**

The **-UDOC** command replaces an existing document in memory with a new one and returns the new document ID.

- **Get document data (-GDOC) command**

The **-GDOC** command returns data from a document in memory for the given document identifier.

- **Excel workbook (-FILE) command**

The Excel workbook name. Use this command for data sources or data targets.

- **[Worksheet \(-WRKSHEET\) command](#)**
The Excel sheet name in the workbook. This command is required. Use this command for data sources or data targets.
- **[Read Mode \(-READMODE\) command](#)**
The **-READMODE** command specifies the type of read operation to be performed. Use this command for data sources.
- **[Worksheet Identifier \(-IDENTIFIER\) command](#)**
The **-IDENTIFIER** command specifies how to identify a worksheet. Use this command for data sources.
- **[Worksheet Index \(-WRKSHEETINDEX\) command](#)**
The **-WRKSHEETINDEX** command specifies the index of the worksheet. Use this command for data sources.
- **[Template \(-TEMPLATE\) command](#)**
The Excel template workbook name. This command is optional. Use this command for data targets.
- **[Excel transformation \(-XLS\) command](#)**
The **-XLS** command allows the map to transform map data in delimited format to Excel formatted data using the GET map rule. PUT receives map data in delimited format and transforms it to Excel formatted data. This command allows the same with GET, so that it can return Excel formatted data back to the map by performing in-memory transformation. The delimited format data has to conform to the Excel importer generated type tree for the corresponding worksheet in the Excel workbook.

Transaction scope and document persistence in memory

When a document is added using the **-ADOC** command, the Excel Adapter stores the document in memory. Documents stored in memory are identified by a document identifier that is either provided by the user, or generated by the Excel Adapter. The document persists in memory as specified by the transaction scope in the card settings.

- When the transaction scope is MAP, the document is accessible in memory to later map rules in the same map. The document is removed from memory when the map completes processing.
- When the transaction scope is CARD, the document is accessible in memory to later map rules on the same card. The document is removed from memory when the card completes processing, and is not accessible to map rules in later cards.

data_from_map option

Some of the Excel Adapter commands that are described in this documentation include the *data_from_map* option. The *data_from_map* option represents the data that is passed as the third parameter of a **GET** function. It is not a parameter on the command line.

If both the **-FILE** and *data_from_map* options are provided, then *data_from_map* has precedence over the **-FILE** option.

Add document (-ADOC) command

The **-ADOC** command adds a file on the local file system, or a remote file accessible through **HTTP** protocol, as a document in memory, and returns the document ID to the map. Alternatively, the **-ADOC** command can add dynamic data from a map as a document in memory and return the document ID to the map.

-ADOC command syntax



-FILE

Adds a single file, specified by a local file URL (**file:///**), or remote file URL (**http://**), as a document in memory, and returns a document ID to the map. The **-FILE** keyword is optional, but you must specify either the **-FILE**, or the *data_from_map* option on the **-ADOC** command. The **doc_ID** parameter is optional. If specified, the **-ADOC** command associates the user-specified identifier for the document in memory and returns the same. If not specified, the **-ADOC** command generates a unique document identifier for the document in memory.

data_from_map

Adds dynamic data from a map as a document in memory and returns the document ID to the map. The *data_from_map* is optional, but you must specify either the **-FILE**, or the *data_from_map* option on the **-ADOC** command.

Examples

The following example adds the `parsewds.txt` file as a document in memory and returns the adapter generated, unique document ID to the map:

```
=GET("EXCEL", "-FILE file:///C:\parsewds.txt -ADOC")
```

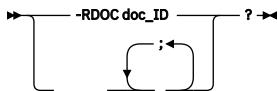
The following example adds the `Hello World!!` text string as a document in memory and returns `mydocid` to the map:

```
=GET("EXCEL", "-ADOC mydocid", "Hello World!!")
```

Remove document (-RDOC) command

The **-RDOC** command removes a document from memory. By default, documents are removed from memory when map, or card processing completes. Transaction scope card setting determines the document persistence in memory. Use this command if too many intermediate documents accumulate in memory during map processing.

-RDOC command syntax



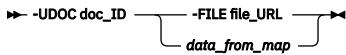
`doc_ID`

doc_ID indicates the document identifier of a document to be deleted from memory, if needed.

Replace document (-UDOC) command

The **-UDOC** command replaces an existing document in memory with a new one and returns the new document ID.

-UDOC command syntax



-FILE

Specifies the local file URL (`file:///`), or remote URL (`http://`), of the document that is to replace the document in memory. Either the **-FILE** or the *data_from_map* option is required.

data_from_map

Replaces the document in memory with dynamic data from a map. Either the *data_from_map*, or the **-FILE** option is required.

Get document data (-GDOC) command

The **-GDOC** command returns data from a document in memory for the given document identifier.

-GDOC command syntax



`doc_ID`

doc_ID indicates the document identifier of a document to be fetched from memory.

Excel workbook (-FILE) command

The Excel workbook name. Use this command for data sources or data targets.

-F *workbook.xls[x]*
-FILE *workbook.xls[x]*

The **-FILE** command is optional for **GET** rules but required on input cards. For example, the workbook name is not required if a map uses the File Adapter to read an Excel workbook and passes the data in memory to the Excel Adapter. The adapter can read from either the .xls or the .xlsx workbook format.

The **-FILE** command specifies a URL to a local file on the file system (`file:///`), or a remote file on a host server (`http://`), or a document in memory (`memory://`). If the workbook file name alone is provided for the **-FILE** command, it is located relative to the **map** directory.

Worksheet (-WRKSHEET) command

The Excel sheet name in the workbook. This command is required. Use this command for data sources or data targets.

-W *sheetname*
-WRKSHEET *sheetname*

Read Mode (-READMODE) command

The **-READMODE** command specifies the type of read operation to be performed. Use this command for data sources.

-RM { *wsl* | *wsd* }
-READMODE { *wsl* | *wsd* }

The **-READMODE** command is optional. Valid values for this command are: *wsl* (worksheet list) and *wsd* (worksheet data). Default is *wsd*.

Worksheet list specification for **-READMODE** command returns the delimited list of worksheet names defined in the Excel workbook. The delimiter between worksheet names is Line feed (<LF>) character. Ignores **-WORKSHEET** command if specified in the command line.

Worksheet Identifier (-IDENTIFIER) command

The **-IDENTIFIER** command specifies how to identify a worksheet. Use this command for data sources.

-I { wsi / wsn }
-IDENTIFIER { wsi / wsn }

The **-IDENTIFIER** command is optional. Valid values for this command are: *wsi* (worksheet by index) and *wsn* (worksheet by name). Default is *wsn*.

Worksheet by index specification for **-IDENTIFIER** command returns worksheet data by index number of the worksheet specified with **-WRKSHEETINDEX** command.

Worksheet Index (-WRKSHEETINDEX) command

The **-WRKSHEETINDEX** command specifies the index of the worksheet. Use this command for data sources.

-WI worksheet_index
-WRKSHEETINDEX worksheet_index

This **-WRKSHEETINDEX** command is required when **-IDENTIFIER wsi** has been specified. Worksheet index number specified must be greater than or equal to 1.

Template (-TEMPLATE) command

The Excel template workbook name. This command is optional. Use this command for data targets.

-P template.xls[x]
-TEMPLATE template.xls[x]

The **-TEMPLATE** command specifies the URL to a local file on the file system (**file:///**), or to a remote file on a host server (**http://**), or to a document in memory (**memory://**). If the workbook file name alone is provided for the **-TEMPLATE** command, it is located relative to the **map** directory. The **-TEMPLATE** command can also be used to add a new worksheet to an existing Excel workbook.

Excel transformation (-XLS) command

The **-XLS** command allows the map to transform map data in delimited format to Excel formatted data using the GET map rule. PUT receives map data in delimited format and transforms it to Excel formatted data. This command allows the same with GET, so that it can return Excel formatted data back to the map by performing in-memory transformation. The delimited format data has to conform to the Excel importer generated type tree for the corresponding worksheet in the Excel workbook.

This command is generally useful in environments where file system access is restricted to the users, and Excel transformations must be performed in memory and then transformed output to be returned to the map for further processing. A stream of Excel formatted data is passed back to the map with GET:

=GET("EXCEL","-XLS -TEMPLATE memory://<template-document-id>,<delimited-map-data>)

A template should also be added to the memory, if it is not a remote file, using the **-ADOC** command when performing in-memory transformation of delimited formatted data to Excel formatted data. The **-ADOC** command returns a user specified document ID, or a system generated ID for the template.

This two step process is shown here:

- **template-document-id=GET("EXCEL",-ADOC <my-template-documentid>,<template-file-data-as-stream>) or template-document-id=GET("EXCEL",-ADOC",<template-file-data-as-stream>)**
- **excel-data-out=GET("EXCEL","-XLS -TEMPLATE memory://<template-document-id>,<delimited-map-data>)**

Store output document in memory

Without the **-XLS** command, instead of streaming to a map, output operation can keep the output document in memory until it is fetched with **-GDOC** command. This is a two step process:

- **=PUT("EXCEL",-FILE memory://<my-output-doc-id> -TEMPLATE memory://<template-document-id>,<delimited-map-data>)** or the adapter command line can be with the output card
- **excel-data-out=GET("EXCEL",-GDOC <my-output-doc-id>")**

Default File URL

It is required to specify **memory://** for memory documents, or **http://** for remote documents, or **file:///** for local file system documents using the **-FILE (-F)** and **-TEMPLATE (-P)** commands. If **file:///** is not specified in the path to the local file system documents, such paths are resolved relative to the map directory.

Reference delimited data

-FILE memory://<delimited-data-document-id> can be specified. Delimited data is in memory and can be referenced instead of passing map data as a third parameter of GET, third parameter has precedence over **-FILE** option.

Google Cloud Storage Adapter

The Google Cloud Storage Adapter provides access to Google Cloud Storage buckets and objects in the Google Cloud Platform.

The adapter supports the following features:

- The Google Cloud Storage adapter provides access to Google Cloud Storage buckets and objects in the Google Cloud Platform.
 - Discovering available buckets and objects with the option to filter objects based on the provided delimiter and prefix values.
 - Authenticating the user using the specified service account credentials file, or automatically by the environment.
 - Reading the specified object in a single chunk or in multiple chunks of the specified size.
 - Keeping or deleting the specified object after reading and processing it successfully, as part of committing the map transaction.
 - Reading the list of objects in the specified bucket, with optional filtering, for further processing the list in a map.
 - Writing an object, with the option to fail or overwrite the target object if it exists already.
 - Writing an object in a single chunk or in multiple chunks by configuring the map inputs to run in burst mode.
 - Writing data to an intermediate (temporary) object and moving it to the target object as part of committing the map transaction.
- **Authenticating Connections**
The Google Cloud Storage adapter supports service account based authentication and authorization. Refer to the **Credentials File** adapter property documentation for more information.
- **Adapter properties and commands**
This section lists the properties supported by the Google Cloud Storage adapter.
- **Examples**

Authenticating Connections

The Google Cloud Storage adapter supports service account based authentication and authorization. Refer to the **Credentials File** adapter property documentation for more information.

Adapter properties and commands

This section lists the properties supported by the Google Cloud Storage adapter.

Credentials File

Property identifier: credentials_file

Adapter command (short syntax): -CF**credentials_file**

Adapter command (long syntax): -CREDENTIALSFILE**credentials_file**

Specifies JSON file with credentials of the service account used to access the Cloud Storage service. When not specified, the authentication is done using the credentials file pointed to by the GOOGLE_APPLICATION_CREDENTIALS environment variable. When running in the Google Cloud environment, where applicable the authentication may be performed automatically by the environment using the default service account for the environment in which the adapter is running. Note that the adapter only supports authentication for Google service accounts. Authentication for Google end-user accounts and API keys is not supported.

Bucket

Property identifier: bucket

Adapter command (short syntax): -B **bucket_name**

Adapter command (long syntax): -BUCKET **bucket_name**

Specifies the name of the bucket to access. The bucket must exist already. When configuring the adapter in Design Server, the list of existing buckets can be fetched, and a bucket can be selected from the list.

Object

Property identifier: object

Adapter command (short syntax): -O **object_name**

Adapter command (long syntax): -OBJECT **object_name**

Specifies the name of the object (blob) to access. When configuring the adapter in the Design Server web UI, the list of existing objects in the specified bucket can be fetched and an object can be selected from the list.

Read mode

Property identifier: read_mode

Adapter command (short syntax): -RM object_data|object_list

Adapter command (long syntax): -READMODE object_data|object_list

The read operation mode. Two modes are supported:

- Object Data - in this mode the adapter retrieves the contents of the specified object. This is the default read mode. The corresponding adapter command value is object_data (case-insensitive).
- Object List - in this mode the adapter retrieves a list of objects in the specified bucket. Each object name in the list is terminated with a linefeed character. If Prefix and Delimiter options are specified, they are used as filters when assembling the list of objects. The corresponding adapter command value is object_list (case-insensitive).

Prefix

Adapter command (short syntax): -PFX **prefix**

Adapter command (long syntax): -PREFIX **prefix**

Specifies object name prefix to use for filtering object names for the object discovery operation, and for listing objects for the Object List mode read operation. When specified, only the object names that start with it are included in the result.

Delimiter

Adapter command (short syntax): -DEL **delimiter**

Adapter command (long syntax): -DELMITTER **delimiter**

Specifies object name delimiter to use for filtering object names for the object discovery operation, and for listing objects for the Object List mode read operation. When specified, it is treated as path delimiter, and for object names that include the delimiter in their name, only the portion of the object name up to and including the specified delimiter is included in the result. Note that if the prefix filter is specified as well, any delimiters present in the prefix position are ignored, and only the delimiters present after the prefix are considered for the delimiter filtering.

Overwrite

Adapter command (short syntax): -OW

Adapter command (long syntax): -OVERWRITE

When selected, it instructs the adapter to overwrite the object is writing data to in case the object with the same name already exists. The default is not to overwrite the object but to report an error if attempt it made to write data to an object that already exists.

Read Success Action

Adapter command (short syntax): -RSA keep|delete

Adapter command (long syntax): -READSUCCESSACTION keep|delete

Specifies the action to perform on the object after successfully reading it. The adapter performs this action when requested to commit the source transaction under which it read the object. The action is performed as part of the transaction commit operation perform. The supported actions are:

- Keep - the adapter does not perform any action and leaves the source object intact. This is the default action. The corresponding adapter command value is keep (case-insensitive).
- Delete - the adapter deletes the object. The corresponding adapter command value is delete (case-insensitive).

Read Chunk Size (Bytes)

Adapter command (short syntax): -RCS **size**

Adapter command (long syntax): -READCHUNKSIZE **size**

Size of chunks, in bytes, in which to retrieve data from the source object. When the property is not specified, or is set to a non-positive value, the entire object data is retrieved in a single chunk.

Intermediate Object Mode

Adapter command (short syntax): -IOM

Adapter command (long syntax): -INTERMEDIATEOBJECTMODE

Instructs the adapter to write data to a uniquely-named intermediate object, and to copy that object to the specified target object when committing the target transaction. The adapter automatically creates a unique name to use for the intermediate object, by generating a universally unique identifier (UUID) value consisting of 68 dash and hex digit characters, to which it appends an underscore character and the specified target object name.

Intermediate Object Prefix

Adapter command (short syntax): -IOP **prefix**

Adapter command (long syntax): -INTERMEDIATEOBJECTPREFIX **prefix**

In addition to the unique name for the intermediate object that the adapter constructs automatically, this property provide option to include additional user-defined prefix in the intermediate object names. This can be used to help identify or filter intermediate objects in the bucket created by the adapter.

Intermediate Object Failure Action

Adapter command (short syntax): -IOFA keep|delete

Adapter command (long syntax): -INTERMEDIATEOBJECTFAILUREACTION keep|delete

Specifies action to perform on the intermediate object in case of a failed write operation. The adapter performs this action on the intermediate object as part of the target transaction rollback operation. Note that this action applies only to the intermediate object, not to the specified target object. The supported actions are:

- Keep - Preserve the intermediate object. This can help with troubleshooting problems and can also be used to force-commit the transaction, by manually copying the object to the target object. The corresponding adapter command value is keep (case-insensitive).
- Delete- Delete the intermediate object. This is the default action. The corresponding adapter command value is delete (case-insensitive).

Logging

Adapter command (short syntax): -T[V|E][+] [log_file]

Adapter command (long syntax): -TRACE[VERBOSE|ERROR][+] [log_file]

This property, along with its two sub-properties, Append Log and Log File Name is used to enable logging at the adapter level. When enabled, the adapter logs messages with details about the operations it is performing on the Cloud Storage service. The default logging level is Info and is used to log informational level messages. The level Error can be specified to enable logging of error level messages only, and the level Verbose can be used to enable logging of all messages. The property Append Log can be set to instruct the adapter to append messages to the log file if the log file exists already, instead of overwriting it, which is the default option. The sub-property Log File Name can be used to specify the file name for the log file. By default the log file name is m4gcstorage.mtr and the file is created in the current map directory.

Convert Data

Specifies to convert data to parsed form. If enabled, support for converting CSV, Avro, Excel, Parquet and JSON documents into sets of data records in the CSV format. This means that the data produced by the converters can be loaded directly into relational databases.

Data Format

Specifies the data format. This is a mandatory property. Select one of the following as per the data format type. This property field is applicable when Convert Data property toggle is enabled.

- CSV
- Excel
- Avro
- Parquet
- JSON

Header

If enabled, specifies whether the CSV data contains a header row.

Quote

Specifies the quote character. Default value is ".

Delimiter

Specifies the field delimiter character. Default value is ,.

Escape

Specifies the escape character.

Separator

Specifies the row separator characters. Default value is \r\n.

Character Set

Specifies the character set of the data. Default value is UTF-8.

Limit

Specifies the maximum number of records to return.

Sample Size

Specifies the number of records to analyze when determining the data structure. Default value is 100.

Worksheet Index

Specifies the index of the worksheet. The index of the first worksheet is 1 (default).

Select Source Tables

Select which arrays of objects in the AVRO data you wish to map to tables in the target.

Find Array

If enabled, specifies that the output will iterate on the first array found in the JSON document.

Array Path

Specifies the path of the JSON array within the document. Omit if the document is an array. For example,"/resources".

Examples

GET function example for fetching message from Cloud Storage subscription

The following is an example of a **GET** map rule used to retrieve data from an object in the Cloud Storage bucket and performing authentication explicitly based on the specified credentials file. Logging is enabled, with verbose level and default log file m4gcstorage.mtr created in the map directory.

Replace *creds_file*, *bucket_name* and *object_name* with the credentials file path, bucket name and object name applicable to your environment.

```
GET("GCSTORAGE", "-CF creds_file -B bucket_name -O object_name -RM OBJECT_DATA -TV")
```

PUT function example for inserting message to a Cloud Storage topic

The following is an example of a **PUT** map rule used to write data to an object in the Cloud Storage bucket and overwrite the object if it exists already. Authentication is performed automatically, such as using the GOOGLE_APPLICATION_CREDENTIALS environment variable. Logging is enabled, with error level and appending messages to the specified log file. The last argument, *input_data* represents the data to be written to the object, and may be a hard-coded data, or, more typically, a reference to a type from another map card.

Replace *log_file*, *bucket_name* and *object_name* with the log file name, bucket name and object name applicable to your environment:

```
PUT("GCSTORAGE", "-B bucket_name -O object_name -OW -TE+ log_file", input_data)
```

Google Pub / Sub Adapter

The Google Cloud Pub/Sub adapter provides access to Google Cloud Pub/Sub topics and subscriptions in the Google Cloud Platform.

The adapter supports the following features:

- Testing the configured connection by authenticating the user and attempting to list the available topics.
- Discovering available topics and subscriptions of the pull kind.
- Authenticating the user using the specified service account credentials file, or automatically by the environment.
- Reading messages from the specified pull subscription, with the option to create the subscription if it does not exist already.

- Option to specify message deadline when creating a new subscription, and separately for the messages retrieved from a subscription.
- Option to acknowledge retrieved messages immediately, never, or when committing the source transaction.
- Option to specify number of messages to poll per request.
- Logical message mode, where multiple Pub/Sub messages are treated as a single logical message.
- Lets you write messages to the specified topic with the option to create the topic automatically if it does not exist.
- Option to batch messages locally before publishing them to the Pub/Sub service.
- Option to specify the publish timeout.
- Configuring the adapter as a listener for triggering flows based on the arrival of messages on the specified pull subscription.

- **[Authenticating Connections](#)**

The Google Cloud Pub/Sub adapter supports username/password-based authentication method.

- **[Adapter properties and commands](#)**

This section lists the properties supported by the Google Cloud Pub/Sub adapter.

Credentials File

Specifies the JSON file with credentials of the service account used to access the Cloud Pub/Sub service. When not specified, the authentication is done using the credentials file pointed to by the GOOGLE_APPLICATION_CREDENTIALS environment variable. When running in the Google Cloud environment, where applicable the authentication may be performed automatically by the environment using the default service account for the environment in which the adapter is running. The adapter only supports authentication for Google service accounts. Authentication for Google end-user accounts is not supported.

The corresponding adapter command is -CF (or -CREDENTIALSFILE) *credentials_file*.

Username

Specifies the username for the connection. This is optional property. For many drivers, you can specify the username as part of the connection string. Some drivers might use authentication other than username and password, and some drivers might not require authentication at all. The corresponding adapter command is -USER *user_name*.

Project

This property identifies the project that hosts the topics and subscriptions accessed by the adapter. The corresponding adapter command is -P (or -PROJECT) *project_id*.

Topic

Specifies the topic to which messages are written. When reading messages and creating the subscription automatically. This is the topic for which the subscription is created. The corresponding adapter command is -TP (or -TOPIC) *topic_id*.

Create Topic

Instructs the adapter to create the specified target topic if the topic does not exist already. This property is applicable only when the adapter is used in target scenario. The corresponding adapter command is -CT (or -CREATETOPIC).

Subscription

This property identifies the subscription from which to read messages. The adapter supports only pull type subscriptions, and it does not support push type subscriptions. When discovering subscriptions, only the subscriptions of pull type are included in the result. The corresponding adapter command is -S *subscription_id*.

Create Subscription

Instructs the adapter to create the specified subscription if the subscription does not exist already. This property is applicable only when the adapter is used in source scenario. The new subscription is created for the specified topic, and it is created as subscription of pull type.

The corresponding adapter command is -CS (or -CREATESUBSCRIPTION).

Batch Count

Specifies the number of messages to store locally in the Pub/Sub client before flushing the buffer. The default value is 1 which implies immediate flushing of the buffer. The flushing of the batch is controlled using a combination of conditions specified in Batch Count, Batch Size (Bytes) and Batch Delay (Milliseconds) properties. When any of those conditions is met, the buffer is flushed.

The corresponding adapter command is -BC (or -BATCHCOUNT) *count*.

Batch Size (Bytes)

Specifies the size in bytes of the buffer in which to store messages locally before flushing the buffer. The default value is 1 which implies immediate flushing of the buffer. The flushing of the batch is controlled using a combination of conditions specified in Batch Count, Batch Size (Bytes) and Batch Delay(Milliseconds) properties. When any of these conditions is met, the buffer is flushed.

The corresponding adapter command is -BS (or -BATCHSIZE) *size*.

Batch Delay (Milliseconds)

Specifies the duration in milliseconds for storing messages locally in a buffer before flushing the buffer. The default value is 1 which implies immediate flushing of the buffer. The flushing of the batch is controlled using a combination of conditions specified in Batch Count, Batch Size (Bytes) and Batch Delay(Milliseconds) properties. When any of those conditions is met, the buffer is flushed.

The corresponding adapter command is -BD (or -BATCHDELAY) *delay*.

Subscription Deadline (Seconds)

Specifies the deadline in milliseconds to set for the messages on the subscription when the subscription is created by the adapter. This is the interval each subscriber is given to acknowledge the retrieval of a message from the subscription. If the message is not acknowledged during this interval and its deadline is not extended, it is automatically made available for redelivery. The default value is 10 seconds.

The corresponding adapter command is -SD (or -SUBSCRIPTIONDELAY) *deadline*.

Acknowledgments

Specifies when the adapter should acknowledge retrieved messages. The supported options are:

- **Never** - the adapter never sends acknowledgments for the messages. After the deadline for acknowledgment expires, the messages are made available for redelivery. This is the default option. It is case-insensitive. The corresponding adapter command value is never.
- **Immediate** - the adapter sends acknowledgment for messages immediately after retrieving them before the messages are processed by the map. It is case-insensitive. The corresponding adapter command value is immediate.
- **Transactional** - the adapter acknowledges the messages as part of committing the source transaction. It is case-insensitive. The corresponding adapter command value is transactional.

The corresponding adapter command is -ACK (or -ACKNOWLEDGMENTS)never|immediate|transactional.

Message Deadline (Seconds)

Specifies the deadline in milliseconds to set for the messages retrieved from a subscription. This becomes the new time interval. The adapter has to acknowledge the messages before they are made available for redelivery. When the value is not specified, the default deadline value is -1, which represents the deadline set at the subscription level. The corresponding adapter command is -MD (-MESSAGEDEADLINE) deadline.

Poll Message Count

Specifies the maximum number of messages to poll from the Pub/Sub service per single request. The default value is 1 which effectively disables retrieving messages in batches. When more than one message is retrieved per request, the messages are stored locally, and the adapter polls the subscription for new messages only after it has exhausted all locally stored messages.

The corresponding adapter command is -PMC (-POLLMESSAGECOUNT)*count*.

Publish Timeout (Seconds)

Specifies the time in milliseconds. The adapter waits for the publish operation to complete, before timing out and reporting error. The default value is -1 which indicates no timeout. The corresponding adapter command is -PT (-PUBLISHTIME) *timeout*.

Logical Message Mode

Specifies that the adapter will run in a mode in which the payload it exchanges with the framework is assumed to consist of one or more Pub/Sub physical messages. When this mode is not used, each payload the adapter exchanges with the framework corresponds to exactly one Pub/Sub message.

In logical message mode, the payload the adapter exchanges with the framework is in the following format:

32-bit integer representing the total remaining payload size (4 bytes) 32-bit integer representing the size of message 1 (4 bytes) message 1 data 32-bit integer representing size of message 2 (4 bytes) message 2 data.

In target context, a single payload (logical message) provided to the adapter may result in multiple Pub/Sub messages published to the target topic. In source context, multiple Pub/Sub messages retrieved from the subscription may be combined and provided to the framework as a single payload (logical message).

The corresponding adapter command is -LMM (or -LOGICALMESSAGEMODE).

Logical Message Count

Specifies the number of Pub/Sub messages to include in a single logical message when reading messages from the subscription in logical message mode. The default value is 1. The size of the logical message provided by the adapter is controlled by the combination of conditions defined by two properties:

- Logical Message Count and
- Logical Message Size (Bytes).

Whichever condition is met first determines the actual size of the logical message.

The corresponding adapter command is -LMC (or -LOGICALMESSAGECOUNT) *count*.

Logical Message Size (Bytes)

Specifies the size in bytes of a single logical message when reading messages from the subscription in logical message mode. When the total size of messages retrieved from the subscription reaches or exceeds this value, the logical message is considered complete. The default value is 0. The size of the logical message provided by the adapter is controlled by the combination of conditions defined by two properties:

- Logical Message Count and
- Logical Message Size (Bytes).

Whichever condition is met first determines the actual size of the logical message.

The corresponding adapter command is -LMS (or -LOGICALMESSAGESIZE) *size*.

Limit

Specifies the number of messages to retrieve from the subscription. The default value is 1. The special value "S" indicates all available messages and the value 0 means no wait. The corresponding adapter command is -QTY *limit*.

Timeout

Specifies the timeout in seconds to wait for a new message to arrive. The default value is "S". This is a special value that indicates unlimited (infinite) wait. The corresponding adapter command is -LSN *timeout*.

Logging

This property specifies the level of logging to use for the log (trace) file produced by the adapter. The default is Off. The value Information means log informational, the value Errors Only means log error messages only, and the value Verbose means log debug and trace level messages along with the informational and error messages.

The corresponding adapter command is:

-T [E|V] [+|] [file_path]

-T -> Log adapter informational messages.
-TE -> Log only adapter errors.
-TV -> Use verbose (debug) logging. The log file records all activity that occurs while the adapter is producing or consuming messages.
+ -> Appends the trace information to the existing log file. Omit this argument to create a new log file.
file_path -> The full path to the adapter trace log. If you omit this keyword, the adapter creates the m4gcpubsub.mtr log file in the map directory.

Append Log

Flag indicating what to do if the specified log file already exists. When set to true, the log messages are appended to the file. When set to false, the file is truncated, and the messages are written to the empty file. The default value is true.

Examples

GET function example for fetching message from Cloud Pub/Sub subscription

The following is an example of a **GET** map rule used to retrieve messages from a Cloud Pub/Sub subscription and performing authentication explicitly based on the specified credentials file. Messages are acknowledged as part of committing the source transaction.

```
GET("GCPUBSUB", "-CF creds_file -S subscription_name -ACK TRANSACTIONAL -TV")
```

PUT function example for inserting message to a Cloud Pub/Sub topic

The following is an example of a **PUT** map rule used to write messages to a Cloud Pub/Sub topic and create the topic if it does not exist already. Publish timeout is set to 10 seconds. Authentication is performed automatically, such as using the GOOGLE_APPLICATION_CREDENTIALS environment variable.

```
PUT("GCPUBSUB", "-T topic_name -CT -PT 10 -TE+ log_file", input_data)
```

FTP adapter overview

Use the File Transfer Protocol (FTP) Adapter to transport data. When source data arrives at its target, it often must be transformed into a different format for a receiving application to use it successfully.

If you have the Command Server, Launcher, or Software Development Kit on one platform, you can use the FTP adapter to retrieve or send data to another platform.

The adapter can connect to any FTP server that follow RFC 959. For optimal performance, the server should also follows RFC 1123. Noncompliant servers may cause errors in the adapter.

- [FTP adapter introduction](#)
This documentation introduces the File Transfer Protocol (FTP) adapter. You can use this adapter with a Command Server, Launcher, Software Development Kit, or map in a map rule.
- [Adapter Properties and Commands](#)
This section lists the properties supported by the adapter.
- [FTP Adapter command examples](#)
- [Syntax summary](#)
- [Troubleshooting](#)
The FTP adapter can reference up to 1024 temporary files during process execution.
- [Return codes and error messages](#)

FTP adapter introduction

This documentation introduces the File Transfer Protocol (FTP) adapter. You can use this adapter with a Command Server, Launcher, Software Development Kit, or map in a map rule.

System requirements

The minimum system requirements and operating system requirements for the FTP adapter are detailed in the system requirements and the release notes. For runtime purposes, a Command Server is assumed to have already been installed on the computer where the adapter will be installed.

In addition, the following are requirements for installing and running the FTP adapter:

- The FTP adapter must be installed in the same directory as the Command Server for your specific platform.
- TCP/IP must be installed on the machine where the adapter is installed.
- The machine where the FTP adapter is installed must be able to access the FTP servers over a TCP/IP network.

About sockets

Windows or BSD (UNIX) sockets must be installed on the client computer.

- The socket installation must support TCP/IP.
- The client computer must be able to access an appropriate server. To test this, type:
 - telnet target-ip 25 (to test an SMTP e-mail target)
 - telnet target-ip 110 (to test a POP3 e-mail source)

Where *target-ip* is the server IP address, if the test is successful, you will receive a welcome message from the server. To shut down the session, type **QUIT** and press **Enter**.

You might not see what you are typing on the screen.

Access to international VANs

The FTP adapter can access international VANs and communicate with a Trading Manager post office. For detailed configuration information, see the Partner Manager documentation.

Command alias

Specify adapter commands by using a command string on the command line or by creating a command file that contains adapter commands. The command syntax is:

```
-IA[alias] card_num
-OA[alias] card_num
```

In the command syntax, **-IA** is the Input Source Override execution command, and **-OA** is the Output Target Override execution command, *alias* is the adapter alias, and *card_num* is the number of the map card. The FTP adapter alias and corresponding execution commands are listed in the following table.

Adapter	Alias	As Input	As Output
FTP	FTP	-IAFTP <i>card_num</i>	-OAFTP <i>card_num</i>

Adapter Properties and Commands

This section lists the properties supported by the adapter.

FTP adapter commands

The following table lists valid commands for the FTP adapter, the command syntax, and whether the command is supported (✓) for data sources, targets, or both.

Table 1. Adapter properties and commands

Name	Syntax	Source	Target
Account	-ACCT <i>account_id</i>	✓	✓
Audit	-AUDIT [+][S] [full_path]	✓	✓
Client Certificate	-CERT {label}	✓	✓
Code Page Local (z/OS only)	-CPL <i>code_page_name</i>	✓	✓
Code Page Remote (z/OS only)	-CPR <i>code_page_name</i>	✓	✓
Fail	-FAIL <i>error_code[:error_code...]</i>	✓	✓
Firewall Connection Method URL	-FIREWALL <i>FTP://firewall_URL</i>	✓	✓
FTP URL	-URL <i>FTP://ftp_URL</i>	✓	✓
GXS	-GXS	✓	✓
Ignore	-IGNORE <i>error_code[:error_code...]</i>	✓	✓
Implicit	-IMPLICIT	✓	✓
Keep Server	-KS	✓	✓
List Option	-LS <i>sort_type</i>	✓	✓
Make Directory	-MKD	✓	✓
Move	-MOVE [<i>file_name directory_location</i>]	✓	
Passive Mode	-PASV	✓	✓
Port Mode	-PORT	✓	✓
Quantity	-QTY <i>number</i>	✓	
Site	-SITE <i>command_text</i>	✓	✓
SSL Encryption Strength	-STR {WEAK STRONG ANY}	✓	✓
SSL Protocol	-SPROTO {SSLv2 SSLv3 SSlv23 TLSv1}	✓	✓
Stage	-STAGE [<i>file_name</i>]		✓
Success	-SUCCESS <i>error_code[:error_code...]</i>	✓	✓
System	-SYS {WIN32 UNIX UNKNOWN}	✓	✓
Trace	-T[E][+][S V] [full_path]	✓	✓
Timeout	-TIMEOUT (<i>nnn</i>)	✓	✓

Account

Specifies the account required to connect to an FTP server. If the account is required to log on, *account_id* must be specified. The corresponding adapter command is **-AC** or (**-ACCT** *account_id*).

account_id: Specifies the account for connecting to an FTP server.

Audit

Generates a log file in the map directory where the adapter is installed with the default name **m4ftp.log**. The log file records audit information with details such as files sent or received from each host, elapsed time, file size, and retry count for each file.

The corresponding adapter command is: **-AUDIT** or **-A**

-AUDIT[+][S] [full_path]

For example:

+> Appends audit information to the existing log file. If no log file exists, a new one will be created.

S-> Summary mode, recording only minimal information in the log file.

full_path-> Creates an audit file with the specified name in the specified directory. By default, the directory is where the map is located, and the file name is **m4ftp.log**.

Client Certificate

Specifies the label of the client certificate in the key store. This command is applicable only when the **SSL_CLIENT secure_mode** setting in the config.yaml file is set, and SSL/TLS encryption is in effect.

The corresponding adapter command is **-CERT {label}**.

label: The name of the entry in the certificate store, which is specified by the **key_store** field in the config.yaml file.

Note: The certificate store contains all personal and trusted certificates in a PKCS12 format file. To see the examples on how to import private keys and certificate chains into this encrypted file, refer to the readme.txt file located in the <Design Studio installation folder>|examples|secureadapt|ssl folder.

Code Page Local

Specifies the code page of the data on the local system. The local system is the system on which the IBM Sterling Transformation Extender is running.

Use this command with the TYPE=ASCII feature of the **-URL** command. This command must be used if the Code Page Remote adapter command (**-CPR**) is specified.

This command is only valid on z/OS systems. The default value is the default **EBCDIC** code page for your system.

The corresponding adapter command is **-CPL code_page_name**.

code_page_name: Specifies the code page name of the data on the local system.

For more information, see the description of the **atoe_l** function in the *IBM z/OS C/C++ Run-Time Library Reference*.

Code Page Remote

Specifies the code page of the data on the remote system, from which or to which the data is being transferred. This command is only valid on z/OS systems; the default value is ISO **8859-1**.

Use this command with the TYPE=ASCII feature of the **-URL** command. This command must be used if the Code Page Local adapter command (**-CPL**) is specified.

The corresponding adapter command is **-CPR code_page_name**.

code_page_name: Specifies the code page name of the data on the local system.

Fail

Specifies the error codes that cause the adapter to fail and disconnect from the FTP server. This is an optional command.

The default behavior of the FTP adapter is to pass 100 and 200-level error codes and to fail only on 300, 400, and 500-level error codes as defined in RFC 959. Any other error codes returned are ignored. Using the Fail adapter command, you can specify the error codes, such as from third-party data, for which the adapter fails.

The Fail adapter command requires a numerical integer for the error code. If necessary, you can specify multiple error codes.

The corresponding adapter command is **-FAIL**.

-FAIL error_code[:error_code...]

error_code: Any numerical integer return code except 100, 200, 300, 400, and 500 level FTP error codes as defined in RFC 959 that you want the adapter to fail on.

Use a colon (:) to separate multiple error codes. For example, the following command indicates that the adapter will fail on return codes 13 and 24:

-FAIL 13:24

Firewall Connection Method

Specifies the URL for the proxy server (firewall). One of the following connection methods must be specified:

- OPEN
- SITE
- TRANSPARENT
- USER
- USERFW
- USERFWPASS
- USERLOGON or USER
- USERLOGONFW or USERFW

The corresponding adapter command is **-FW** or **(-FIREWALL)**.

Generally, the command format is written as:

-FIREWALL FTP://[fw_user[:fw_pass]@[fw_host[:fw_port]]/method

fw_user: Specifies the username required for connecting to the proxy server. Unless you specify a password, this parameter must be followed by @ if you specify fw_host.

:fw_pass: Specifies the password required to authenticate the username. This parameter must be followed by @ if you are also specifying fw_host.

fw_host: Specifies the name (or address) of the proxy server to which to connect.

:fw_port: Specifies the name (or number) of the port to use for connection.

/method: Required value. The method dictates how the firewall connection is established based on a combination of the connection information provided using - **FIREWALL** (for proxy server URL) and - **URL** (for FTP URL). Select one of the following methods that specifies the corresponding Connection Information.

Method Connection Information

OPEN ftp_host, USER ftp_user, PASS ftp_pass

TRANSPARENT is the only method directly connecting to the FTP server; all other methods connect to the firewall. For more information on the FTP variables shown in the Connect Information column, see the section for the FTP URL (-URL) adapter command.

Adapter behavior when the -FIREWALL command Is present

A firewall isolates and restricts some computers from reaching other computers.

Only a small part of the FTP adapter knows its presence if a firewall is mentioned.

Without a firewall, the start of a file transfer begins with:

- Establish a Socket connection to the remote host.
- Issue the **USER** command over the connection.
- Issue the **PASS** command over the connection.

When a firewall is used, more steps are performed at the beginning of the file transfer.

These steps depend on the *method* specified in the last part of the -FIREWALL command.

OPEN - Establish a Socket connection to the firewall host and issue the following command over the connection in this order:

OPEN ftp_host USER ftp_user PASS ftp_pass

SITE - Establish a Socket connection to the firewall host and issue the following command over the connection in this order:

USER fw_user PASS fw_pass SITE ftp_host USER ftp_user PASS ftp_pass

TRANSPARENT - Establish a Socket connection to the firewall host and issue the following command over the connection in this order:

USER fw_user PASS fw_pass USER ftp_user PASS ftp_pass

USERFW - Establish a Socket connection to the firewall host and issue the following command over the connection in this order:

USER ftp_user@ftp_host fw_user PASS ftp_pass USER ftp_user PASS ftp_pass

USERFWPASS - Establish a Socket connection to the firewall host and issue the following command over the connection in this order:

USER ftp_user@fw_user@ftp_host PASS ftp_pass@fw_pass

USERLOGON or USER - Establish a Socket connection to the firewall host and issue the following command over the connection in this order:

USER fw_user PASS fw_pass USER ftp_user@ftp_host PASS ftp_pass USER

USERLOGONFW or USERFW - Establish a Socket connection to the firewall host and issue the following command over the connection in this order:

USER fw_user@ftp_host PASS fw_pass USER ftp_user PASS ftp_pass

URL syntax

As the syntax for the rest of the URL varies depending on the scheme selected, URL schemes use direct IP-based protocol for a specified host on the Internet follow a common syntax for the scheme-specific data:

//user:password@host:port/url_path

The scheme-specific data may exclude some or all parts of the following components: *user:password@*, *:password*, *:port*, and */url_path*

The scheme-specific data starts with a double slash (//), indicating compliance with the common Internet scheme syntax. The *user:password* component conforms to the following rules:

user: An optional username. Some schemes allow the specification of a username.

password: An optional password. When present, it follows the username and is separated by a colon. The username and password are followed by an at sign (@). Any :, @, or / characters must be encoded. See URL encoding.

The *url_path* of an FTP URL has the following syntax:

cwd1/cwd2/.../ cwdN/name; type = xfer_type

Where *cwd1* through *cwdN* and *name* are possibly encoded strings and *xfer_type* is one of the characters a, i, or d. The part; *type=xfer_type* be omitted. The *cwdx* and *name* parts are empty. The whole *url_path* be omitted, including the /, delimiting it from the prefix containing *user*, *password*, *host*, and *port*.

The *url_path* is interpreted as a series of FTP commands as follows:

- Each *cwd* element will be sequentially supplied as the argument to a **CWD**, change working directory, command.
- If the type code is "d," perform an **NLST**, name list, or command with *name* as the argument and interpret the results as a file directory listing.
- Otherwise, perform a **TYPE** command with *xfer_type* as the argument and then access the file named *name*. For example, using the **RETR** command.
The *xfer_type* argument has been extended beyond what the RFC supports, specifically the D1, D2, D3, and D4 sections. Also, *host:port* be excluded, although it is rarely useful to do so; the *host* defaults to *localhost*.

FTP URL

Specifies the file names to be retrieved (adapter source) or created and appended (adapter target). This command is required to establish a connection to an FTP server.

The corresponding adapter command is **-URL**

Generally, the command format is:

-URL [FTP://][ftp_user[:ftp_pass]@][ftp_host[:ftp_port]]
`[/[...][dir/[dir/...]] [filename] ;type=xfer_type[+]]`

Option	Description
FTP:	Specifies a connection to an FTP server. If not specified, FTP: is assumed, along with the <code>//</code> . If specified, another <code>/</code> must separate the user and host parameters from the file and transfer type.
<code>:ftp_user</code>	Specifies the username to connect to the FTP server. Unless you specify a password, this parameter must be followed by <code>@</code> if you specify <code>ftp_host</code> .
<code>:ftp_pass</code>	Specifies the password that authenticates the username. If you also specify <code>ftp_host</code> , this parameter must be followed by <code>@</code> .
<code>ftp_host</code>	Specifies the name or address of the FTP server to which to connect. If not specified, the default of localhost is assumed.
<code>:ftp_port</code>	Specifies the port name or number to use for connection.
<code>dir/...</code>	Specifies the file's directory path. The ellipsis (...) indicates that this field can repeat, depending on your directory structure.
<code>filename</code>	Specifies the name of the target file. This is required for an adapter target but optional for an adapter source. If not specified, all files in the path are retrieved.
<code>;type=xfer_type</code>	<p>Specifies the transfer type or mode. The supported transfer types vary from server to server. Depending on the server, there might be additional or omitted transfer types. Valid transfer types are:</p> <p><code>A[len][!] ASCII</code></p> <p>Where <code>len</code> specifies the optional record length to pad or truncate the record, and <code>!</code> indicates no line terminator for the record(s).</p> <p>When sending an FTP request to the AS/400, the file must first be created on the AS/400 for the <code>len</code> parameter to pad the record correctly.</p> <p>Description of Transfer type</p> <ul style="list-style-type: none"> • I: Directory (file names only) • LB: Local 8-bit types (usually equivalent to I) • D: Directory (file names only) • D1: Directory (file names only) • D2: Directory (server-defined format) • D3: Directory (easily parsed line format, as defined in Internet draft draft-bernstein-eplf-02.txt) • D4: Directory (machine-readable list format as defined in Internet draft draft-ietf-ftpext-m1st-02.txt)
<code>+</code>	Appends the transferred data to the existing file.

If specified, an additional forward slash (`/`) is required to separate the user and host parameters from the file and transfer type.

GXS

Allows proper connectivity with the GXS VAN using an FTPS connection. The corresponding adapter command is **-GXS**.

When this command is in use, the **-IMPLICIT** adapter command is not necessary.

Generally, the command format is:

-TV -GXS -URL FTPS://john_doe:1234@gxsics.com/send/sendfile;type=A

This command performs several functions:

-TV-> Creates a trace verbose file in the directory where the map is located.

-GXS-> Specifies that a GXS (**-GXS**) secure connection to the server is used.

_URL FTPS://-> Specifies sending the file using **FTPS** to the **URL** address `secure.com`.

User id and Password-> Specifies the **user id** name `John Doe` and a **password** of `1234` is used to login to this account.

/Send/-> Specifies **send** will be the directory where the send file is stored.

sendfile;type=A-> specifies that **sendfile** will be transferred in ASCII mode (**type=A**).

Ignore

Allows the adapter to ignore the specific FTP error codes. The corresponding adapter command is **-I** or (**-IGNORE**).

-IGNORE error_code[:error_code...]

`error_code`: Specifies the FTP error codes that the adapter should ignore. A colon (`:`) should be used to separate multiple error codes.

For example, if you specified the command: **-I450:550**. The adapter ignores errors of type 450 (file busy) and errors of type 550 (file not available) and returns a zero-length file to the map.

Implicit

Specifies implicit FTPS secure connections to a server for sending or receiving data. This command implies the use of port number 990 as the default. If an alternate port number is required, it must be included on the command line after the specified URL. For example, `ftps://secure.com:6377`.

The corresponding adapter command is **-IMPLICIT**.

For example:

-TV -IMPLICIT -URL FTPS://john_doe:1234@secure.com/upload/mail_list;type=A

This command performs several functions:

-TV: Creates a trace verbose file in the directory where the map is located.

-IMPLICIT: Specifies that an implicit (**-IMPLICIT**) secure connection to the server is used.

-URL FTPS://: Specifies sending the file using **FTPS** to the **URL** address *secure.com*.

user id and password: Specifies the **user id** name *John Doe* and a **password** of 1234 are used to login to this account.

upload/mail_list;type=A: Specifies that the send file will be named **mail_list** after being transferred in ASCII mode (**type=A**).

Keep Server

Specifies in a map rule that when retrieving a file from an FTP server, it should not be deleted after transfer. This is an optional command. This option should be used if this behavior is required when using the **GET** mapping function. Alternatively, when running the adapter from an input card, the same behavior can be accomplished using the card property, **OnSuccess**, by setting it to **Keep**.

The corresponding adapter command is **-KS**.

For example:

-KS -URL ftp://user:pass@server/file.txt

List Option

Allows files to be retrieved based on a user supplied sorting preference. This command is used with the existing FTP URL (**-URL**) adapter command and supports a subset of FTP options that control the sequence of file transfers. This is an optional command.

The corresponding adapter command is **-LS**

-LS sort_type

The adapter passes the **-LS** command value to a remote FTP server as an argument when it lists the files in the directory. One of three events happens:

- The FTP server accepts the option and lists the files in the specified order.
- The FTP server ignores the option and lists the files in the default order.
- The FTP server rejects the option, and the adapter returns an error.

For example:

-URL ftp://testftp:test@host/mydir -ls tr

The **-LS** command is issued by the adapter before any file transfer, so that its results are used to control the order of the file transfer. In the example above, all files from *mydir* would be transferred, the oldest file being transferred first.

The following are the command options that are allowed:

t-> Sorts by time stamp.

r-> Reverses the order of sort.

tr-> Reverses the order of sort by time stamp.

rt-> Same as **-LS tr**.

All other list options result in the error message: `invalid ls option logged` in the adapter trace file.

Some FTP servers can ignore the List Option (**-LS**) command options (*t, r, tr, and rt*).

Make Directory

Creates a directory. For instance, if a file specified in a **PUT** operation belongs to a directory that does not exist, this command creates a new directory. If the command is not specified and the file specified on a **PUT** operation points to a directory that does not exist, the adapter will fail. This is an optional command.

The corresponding adapter command is **-MKD**.

For example:

PUT("FTP", "-URL ftp://jdoe:secret@server1/some/folder/ data.txt -MKD", "Some data")

This will force **jdoe/some** and create the **jdoe/some/folder**. A new file called **data.txt** will be added to this new directory.

Move

Moves files from their original location on the source FTP server to a different location on the source FTP server after the files have been transferred to the target server. On the Move adapter command, you can specify a new name for the original source file as a different file name, path, or both.

The FTP Adapter typically provides options for deleting or keeping files on the source FTP server after they are transferred to the target FTP server. The Move command provides additional functionality that permits you to move the files to a different location on the source server, as opposed to, for example, deleting them.

The corresponding adapter command is **-MOVE**.

For example:

-MOVE [file_name | directory_location]

file_name: Specifies the new name on the source FTP server for the transferred original file. This can be specified as the source file's absolute (fully qualified) or relative path.

The following commands demonstrate examples of how you can specify new names for the original file on the source FTP server as absolute and relative path names when the original file name in the FTP command to be downloaded is specified as *myoriginalfilename.txt*:

- **-MOVE /mynewfilename.txt**: This Move command, specified with an absolute path and new file name, moves the *myoriginalfilename.txt* file after it is downloaded from its original location on the source FTP server to your root directory for that FTP client and renames it *mynewfilename.txt*.

- **-MOVE mynewfilename.txt**: This Move command, specified with a new file name, renames the myoriginalfilename.txt file after it is downloaded to mynewfilename.txt on the source FTP server.

directory_location: Specifies the different directory or subdirectory names on the source FTP server for the original or transferred files. The directory name must be terminated with a path separator. Examples include backward slash (\) for Windows and forward slash (/) for UNIX environments.

If the directory does not exist and the **MKD** option is also specified on the command line, the directory will be created relative to the current location of the file or files on the source FTP server that have already been downloaded.

The adapter will fail if the directory does not exist, and the **MKD** option is not specified on the command line.

If you are unsure that the directory exists, use the **MKD** option. When you specify this command option whether the directory exists, the adapter command will avoid a failure result. If the directory does exist, the **MKD** option will not attempt to create the directory and the adapter command will continue.

The following commands demonstrate examples of how you can specify an existing or new subdirectory name for the original file on the source FTP server when the original subdirectory name in the FTP command is specified as myoriginalsubdir:

- **-MOVE /mydiffexistingsubdir/**: This Move command, specified with a different subdirectory that already exists, moves the file after it has been downloaded from its original location on the source FTP server to the mydiffexistingsubdir subdirectory with its original file name.
- **-MOVE /mynewsubdir/ -MKD**: This Move command, specified with a new subdirectory, makes the new mynewsubdir subdirectory in a location that is absolute to the current location and then moves the file after it has been downloaded from its original location on the source FTP server to the new subdirectory with its original file name.

Platform-specific FTP Server limitations

The Move adapter command functionality is subject to platform-specific FTP Server limitations. For example, on z/OS environments, the **-MOVE** option cannot be executed across different file systems. The Hierarchical File System (HFS) on your UNIX System Services (USS) environment cannot move a file to a data set.

Note: The FTP Adapter does not recognize z/OS Partitioned Data Set (PDS) file names.

Passive Mode

Specifies that only passive mode should be used for the FTP data connection. This command is primarily useful for firewalls and is considered a preferred connection method for security reasons.

By default, the adapter attempts to connect using passive mode. If unsuccessful, the adapter uses Port Mode. See the Port Mode adapter command (-PT). During a passive mode connection, the server listens for the connection and dictates where the adapter is to connect. Older servers might not support passive mode.

The corresponding adapter command is **-PV** or (**-PASV**).

Port Mode

Specifies that only port mode should be used for the FTP data connection. This older connection method might be the only way to connect on older servers. The corresponding adapter command is **-PT** or **-PORT**.

By default, the adapter attempts to establish a connection using passive mode. If the adapter is unsuccessful, it uses port mode. See the Passive Mode adapter command **-PV**.

Note: The IBM Sterling Transformation Extender FTP Adapter only supports the FTP secure socket layer (SSL) protocol in passive mode (-PASV). If you specify the **-PORT** option on the command line and the FTP URL specifies the SSL protocol (for example, `ftps://`), the **-PORT** option is ignored, and passive mode is used instead.

Submit command after file transfer

Submits a specified command to the FTP server after each file transfer occurs. The corresponding adapter command is **-PQUOTE**.

-PQUOTE*command*.

command: The command or command string will be submitted to the FTP server.

For example:

The following command records the size of the test.dat file in the adapter trace file after the file transfers:

-PQUOTE 'size test.dat'

Quantity

Specifies the number of files retrieved by the adapter.

In burst mode, each burst might return fewer messages than specified, but the total number of messages returned by the adapter will not exceed the value set by the **-QTY** command. If not specified, the default quantity is the **FetchUnit** size of the burst.

The corresponding adapter command is **-QTY**

-QTY*number*.

number: The number of files to be retrieved by the adapter.

If you want the maximum number of files retrieved in burst mode infinite, you must set the **-QTY** command option number to 's' and specify "`-LSN 0`" on the command line. The `-LSN` command is currently unsupported; however, it must only be used with the `-QTY` command in this scenario.

Submit command before file transfer

Submits a specified command to the FTP server when the connection is established and immediately before each file transfer occurs. The corresponding adapter command is **-QUOTE**.

-QUOTE*command*.

command: The command or command string is to be submitted to the FTP server.

The following command submits the **TYPE B 6** command to a System i® server:

-QUOTE 'TYPE B 6'

Site

Sends an FTP server-specific site command to the FTP server software. The corresponding adapter command is **-SITE**.

-SITE'command_text':

'command_text': Specify the FTP-server-specific site command, enclosed in single quotation marks, to send to the FTP server software.

The following command sends the *namefmt* command to an AS/400 FTP server before files are transferred.

-SITE 'namefmt=1'

To see what commands are accepted by the server, use a BSD-based command line client, typically the FTP command installed with your Windows or UNIX operating system and log in to the server and type the following from the `ftp` prompt: `quote HELP SITE` or `quote SITE HELP`.

The command specified by **-SITE** is executed before the files are transferred.

SSL Encryption Strength

Specifies the encryption strength for FTP connections. It can be set to either WEAK, only using weak algorithms, or STRONG, only using strong algorithms. If not specified, any available algorithm is used. If STRONG is unavailable, then WEAK is used. However, if the export version of the libraries is used, only weak algorithms are available, and this command is ignored.

This command is applicable only when the SSL **secure_mode** setting is in the config.yaml configuration file (/runtime/secure_mode) is set to 0, and RSA-based SSL encryption is in effect.

The corresponding adapter command is **-STR {WEAK/STRONG/ANY}**.

WEAK-> Use strong (non-exportable) encryption only.

STRONG-> Use exportable encryption only.

ANY-> Use any available encryption.

SSL Protocol

Specifies setting the Secure Socket Layer (SSL) protocol level. SSL enables the adapter to process FTPS URLs. The corresponding adapter command is **-SPROTO{SSLv2/SSLv3/SSLv23/TLSv1}**.

SSLv2: Specify SSL protocol version 2.

SSLv3: Specify SSL protocol version 3.

SSLv23: Specify SSL protocol version 2, version 3, TLSv1, TLSv1.1 or TLSv1.2. Whenever the protocol method of **SSLv23** is specified, the protocol of first choice is **TLSv1.2**, with a fallback to **TLSv1.1**, then to **TLSv1**, then to **SSLv3**, and finally to **SSLv2**, depending on the GSKit module that is loaded.

TLSv1: Specify Transport Layer Security (TLS) protocol version 1.

If **-SPROTO** is not specified, the SSL Protocol defaults to **SSLv23**.

This command is applicable only when the SSL **secure_mode** setting is in the config.yaml configuration file (/runtime/secure_mode) is set to 0, and RSA-based SSL encryption is in effect. This command is superseded by an SSL **secure_mode** setting of 1 or 2.

Stage

Use the Stage adapter command to:

- Specify that uploaded data should be staged in a temporary or shadow file.
- Prevent partial updates of a file from being transferred. This is especially important if the FTP server processes uploads in an automated fashion.

During an upload, the FTP server generates a uniquely named temporary file. When the transfer has been completed successfully, the original file is deleted, and the temporary file is renamed. For this process to succeed, you must have access rights associated with your user ID to delete and rename files, and if the optional file name (*file_name*) is omitted, the server must provide a correct Store Unique (STOU) response. The adapter issues a warning if the server response cannot be interpreted. It is recommended that you run a preliminary test to ensure that the process is completed successfully.

The corresponding adapter command is **-STAGE** or **-STG**.

-STAGE[file_name]

file_name: Specifies a unique file name for the temporary file. This option is especially useful if a server does not supply unique file names.

Success

Allows a map to succeed when 300-level error codes, as defined in RFC 959, are returned. If this command option is not specified, the adapter's default behavior of the adapter is to fail on 300-level error codes. This is an optional command.

The corresponding adapter command is **-S** or **(-SUCCESS)**

-SUCCESS error_code[:error_code...]

When you specify **-SUCCESS** on the adapter command line, a numerical integer value between 300 and 399 is required. When this value is returned, the adapter does not fail.

error_code- An FTP error code, defined in RFC 959, between 300 and 399. Separate multiple error codes with a colon (:). For example:

-SUCCESS 301:302:303

System

Specifies the system type when the adapter cannot accurately determine the system type of the FTP server.

The FTP server system type is not necessarily the same as the operating system of the machine it runs on. Some Windows FTP servers use UNIX conventions. If in doubt, consult your server documentation or use the `dir` command to view the results.

The corresponding adapter command line is:

-SYS {WIN32|UNIX|UNKNOWN}

WIN32: The server follows Windows conventions. For example, directories in pathnames are separated by a backward slash (\).

UNIX: The server follows UNIX conventions. For example, directories in pathnames are separated by a forward slash (/).

UNKNOWN: Because the platform is not known, exercise generic code. Do not perform anything specific to a platform.

Trace

Produces a diagnostic file that contains detailed information about FTP adapter activity. The default filename is **m4ftp.mtr**, created in the map directory unless otherwise specified.

The corresponding adapter command is `-T` or `(-TRACE)`

For example:

-T[E][+][S|V] [full_path]

E-> Produces a trace file containing only the adapter errors during map execution.

+-> Appends trace information to the existing log file.

S-> Summary mode. Record only minimal information in the log file. This is the default value.

V-> Verbose mode. Record all activity occurring while the adapter retrieves data in the log file. If not specified, summary mode is assumed.

full_path-> Creates a trace file with the specified name in the specified directory. The default directory is the map directory. The default file name is **m4ftp.mtr**.

You can dynamically override the adapter command line trace options using the Management Console. See "Dynamic Adapter Tracing" in the *Launcher* documentation for detailed information.

Because summary mode is the default, using `-TRACES` provides the same output as `-TRACE`.

Timeout

Specifies the waiting time for the client to connect to the server before the request times out. If this option is not specified, the maximum timeout is 300 seconds, and the connection time for a client varies for different operating systems.

When this option is specified, the connection attempt is made for `-TIMEOUT`, time in seconds, and fails if it is not connected within the specified period.

The corresponding adapter command is `-TIMEOUT`.

-TIMEOUT(nn)

nnn: The number of seconds to wait for activity on the FTP connection before the session ends in error. The default value is 300.

FTP Adapter command examples

Example of the Firewall connection method URL using Userlogon

The following command uses the `USERLOGON` (or `USER`) method to connect to the proxy server `fire` with no port specified, with a username of `jbond` and password of `007`:

```
-FW FTP://jbond:007@fire/userlogon
```

See URL Syntax and URL Encoding for more information.

Example of the input card command in the Map Designer

The following is an example of the input card command in the Map Designer:

```
-T -URL FTP://sales@host/c:/forms/myfile.txt
```

The command causes the `myfile.txt` file to be retrieved from a remote host named `sales`. The `-T` adapter command specifies that the adapter creates a trace file to report adapter activity information during the FTP process.

Syntax summary

Data sources

The following is the command syntax of the FTP adapter commands used for data sources:

```
-URL [FTP://  
  [ftp_user[:ftp_pass]@] [ftp_host[:ftp_port]]  
  [/[/]] [dir/[dir/...]] [filename] ;type=xfer_type[+]
```

```

[-ACCT account_id]
[-AUDIT[+] [S] [full_path]]
[-CERT {label}]
[-CPL code_page_name]
[-CPR code_page_name]
[-FIREWALL FTP://[fw_user[:fw_pass]@][fw_host[:fw_port]]]
[method]
[-GXS]
[-IGNORE error_code[:error_code...]]
[-IMPLICIT]
[-KS]
[-LS sort_type]
[-MKD ]
[-MOVE [file_name|directory_location]]
[-PASV]
[-PORT]
[-QTY number]
[-QUOTE command]
[-SITE command_text]
[-SPROTO {SSLv2|SSLv3|SSLv23|TLSv1}]
[-STR {WEAK|STRONG|ANY}]
[-SYS {WIN32|UNIX|UNKNOWN}]
[-T[E] [+][S|V] [full_path]]

```

Data targets

The following is the command syntax of the FTP adapter commands used for data targets:

```

-URL [FTP://][ftp_user[:ftp_pass]@][ftp_host[:ftp_port]]
    [/[/][dir/[dir/...]] [filename][;type=xfer_type[+]]
[-ACCT account_id]
[-AUDIT[+] [S] [full_path]]
[-CERT {label}]
[-CPL code_page_name]
[-CPR code_page_name]
[-FIREWALL FTP://[fw_user[:fw_pass]@][fw_host[:fw_port]]]
[method]
[-GXS]
[-IGNORE error_code[:error_code...]]
[-IMPLICIT]
[-KS]
[-LS sort_type]
[-MKD ]
[-PASV]
[-PORT]
[-PQUOTE command]
[-QUOTE command]
[-SITE command_text]
[-SPROTO {SSLv2|SSLv3|SSLv23|TLSv1}]
[-STAGE [file_name]
[-STR {WEAK|STRONG|ANY}]
[-SYS {WIN32|UNIX|UNKNOWN}]
[-T[E] [+][S|V] [full_path]]

```

Using FTP adapter commands

Use the adapter commands from the Map Designer or Integration Flow Designer as the data source of an input map card and as the data target for an output map card. You can also use the adapter on the command line or in a map rule using the functions >GET, PUT, or RUN.

The FTP adapter requires the -URL adapter command for the GET>Source>Command and PUT>Target>Command settings.

See "[FTP Commands](#)" for more information on FTP adapter commands and their syntax. For general information about commands, see the *Resource Adapters documentation*.

Examples of using FTP adapter commands

To retrieve the mine.txt file, located on the host computer, with no port specified, under c:\forms with no username or password specified, enter the following command:

```
-URL FTP://host/c:/forms/mine.txt
```

The following command example would retrieve the entire contents of the **forms** directory because the file name is omitted from the **host** computer:

```
-URL FTP://host/c:/forms/
```

To retrieve the payments.txt file, located on the source FTP server, with no port specified, with a username and password specified, and then move the file from its original location on the source FTP server to your root directory for that FTP client and rename it paymentsbackup.txt, enter the following command:

```
-URL FTP://ftpuser:mypassword@sourceftpserver/payments.txt -MOVE /paymentsbackup.txt
```

To retrieve the invoices.txt file, located on the source FTP server in the billing subdirectory, no port specified, with a user name and password specified, make the new billingbackup subdirectory in a location that is absolute to the current location of the original file, and then move the file from its original location on the source FTP server to the new subdirectory on the source FTP server with its original file name, enter the following command:

```
-URL FTP://ftpuser:mypassword@sourceftpserver/billing/invoices.txt
      -MOVE /billingbackup/ -MKD
```

URL syntax

The syntax for the rest of the URL varies depending on the scheme selected. URL schemes that involve the direct use of an IP-based protocol for a specified host on the Internet use the following common syntax for scheme-specific data:

```
//user:password@host:port/url_path
```

Some or all parts of *user:password@*, *:password*, *:port*, and */url_path* can be excluded. The scheme-specific data starts with a double slash (*//*) to indicate that it complies with the common Internet scheme syntax. The *user:password* component conforms to the following rules:

user: An optional username. Some schemes allow the specification of a user name.

password: An optional password. When present, it follows the user name and is separated by a colon. The user name and password are followed by an at sign (@). Any :, @, or / characters must be encoded, within the user and password field. See "URL Encoding".

The *url_path* of an FTP URL has the following syntax:

cwd1/cwd2/.../ cwdN/ name; type= xfer_type

Where *cwd1* through *cwdN* and *name* are possibly encoded strings and *xfer_type* is one of the characters a, i, or d. The part *; type= xfer_type* be omitted. The *cwdx* and *name* parts are empty. The whole *url_path* be omitted, including the /, delimiting it from the prefix containing *user*, *password*, *host*, and *port*.

The *url_path* is interpreted as a series of FTP commands as follows:

- Each *cwd* element will be sequentially supplied as the argument to a CWD, change working directory, command.
- If the type code is "d," perform an NLST, name list, command with *name* as the argument, and interpret the results as a file directory listing.
- Otherwise, perform a TYPE command with *xfer_type* as the argument and then access the file named *name*. For example, using the RETR command.
The *xfer_type* argument has been extended beyond what the RFC supports, specifically the D1, D2, D3, and D4 sections. Also, *host:port* should be excluded.
Although it is rarely useful to do so, the *host* defaults to localhost.

URL encoding

The URL does not need to be completely 'URL-encoded'. As shown in the following table, you only need to use escape characters for tokens specifically used in an FTP URL. Any FTP URL that contains spaces must be enclosed in quotation marks; for example:

-URL "FTP://Home Machine/FTP Files"

Table 1. The following tokens are replaced with escape characters, respectively.

Tokens	Escape characters
% (percent)	%25
@ (at)	%40
/ (forward slash)	%2F
: (colon)	%3A
; (semicolon)	%3B
& (ampersand)	Escape is not needed; use it as is.
# (pound)	Escape is not needed; use it as is.
8-bit or control characters	Escape is not needed; use it as is.

Troubleshooting

The FTP adapter can reference up to 1024 temporary files during process execution.

For error codes and messages the adapter returns, see Return Codes and Error Messages.

Various troubleshooting tools are available in case of problems using the FTP adapter for data sources or targets for a map. For example, if you attempt to run a map that uses the FTP adapter and encounter problems or do not receive the expected results, use the following adapter troubleshooting tools:

- adapter audit log (.log)
- adapter trace file (.mtr)

Adapter audit log

The adapter audit log is a text file created during the adapter execution that records information about the events that occurred during the adapter activity. Use the adapter command (-AUDIT) to produce adapter audit log. The default log name is **m4ftp.log**, located in the map directory. See Trace (-T or -TRACE) for more information about the Trace command.

The adapter audit log contains information including the following:

- host identifier
- files sent or received
- size of each file
- elapsed time
- retry count for each file

Sample FTP adapter audit log

```
BEGIN FTP AUDIT
Receiving from Host: localhost
Filename      Filesize   Time   Retry
-----
rollback.gif    183 KBytes  000s    0
END FTP AUDIT
```

Adapter trace file

The adapter trace file contains detailed information the adapter provides and records the actions taking place, such as connections established and statements executed. The trace file is produced during the adapter execution and can be used as a debugging aid. To produce a trace file for the adapter, use the (-TRACE) adapter command. The default adapter trace file name is **m4ftp.mtr**, located in the map directory. See Audit (-A or -AUDIT) for more information about the Audit command.

Sample FTP adapter trace file

```
Retry Count is 0. Retry Interval is 0. Rollback is OFF.
Fetch Unit is 0. Burst Mode is OFF
Run Started at 08: 49: 38.911 on 03/28/00.
>220- (S)
>220-This is a demo version of the Vermillion FTP Daemon, licensed only for (S)
>220-a 30 day evaluation period. For continued use, you must purchase a (S)
>220-license. For more information or to download the latest version, please (S)
>220-visit Arcane Software on the web at `http://www.arcanesoft.com'. (S)
>220- (S)
>220-Note: This session is limited to 15 minutes. If you are transferring a (S)
>220-    file at that time, the system will wait until it is complete (S)
>220-    before automatically disconnecting you. (S)
>220- (S)
<USER sls

>331 Password required for sls. (S)
<PASS sls>

230 User sls logged in. (S)
<SYST

>215 UNIX Type: L8 (S)
Setting AIX/UNIX-specific parameters.
<CWD /d:

>250 CWD command successful. (S)
<TYPE I

>200 Type set to I. (S)
<PASV

>227 Entering passive mode (127,0,0,1,8,230) (S)
Using port `2278' for the connection.
Connecting to `127.0.0.1'.
Socket Opened.
Connected.
<RETR rollback.gif

>150 Opening BINARY mode data connection for rollback.gif (187714 bytes). (S)
(187714 bytes received)
Socket Closed.
Connection closed.
>226 Transfer Complete. (S)
<QUIT

>221 Goodbye. (S)
Socket Closed.
Connection closed.
Winsock Client ID 1 disabled.
Read 187714 bytes from /d:/rollback.gif.
Data in E: \167.tmp kept.
Run Completed at 08: 49: 39.742 on 03/28/00.
```

Temporary file limit

During a PUT or output card operation, the FTP adapter transfers files at the end of a successful map execution. When a run map invokes the FTP adapter, and the FTP operation is set to Card scope, the adapter transfers the files after the run map completes.

The default FTP operation is Map scope. When a run map invokes the FTP adapter, and the FTP operation is set to Map scope, the FTP adapter stores the contents in a temporary file. After the top-level map completes successfully, the FTP adapter sends all the temporary files to the target FTP server. If the number of FTP temporary files in all executing maps exceeds the limit of 1024, the adapter writes a message to the trace file and returns an adapter error message.

Return codes and error messages

Return codes and error messages are returned when the particular activity is completed. Return codes and error messages can also be recorded as specified in the audit logs, trace files, and execution summary files,

Messages

The following is a listing of all the codes and messages that can be returned when using the FTP adapter for sources or targets.

Adapter return codes with positive numbers are warning codes that indicate a successful operation. Adapter return codes with negative numbers are error codes that indicate a failed operation.

Table 1. Return Code with Messages

Return Code	Message
0	OK
-1	Insufficient memory to continue
-1	Library Initialization Failed
-2	Invalid Entry Point. (AIX adapters only)

Return Code	Message
-3FTP	Could not load adapter
-3	Error Sending Data
-4	Error Receiving Data
-5	FTP: Insufficient memory to continue
1	No data was provided. Create on content specified: no data sent.
2	Could not interpret the STOU response! The target file will not be renamed.
-1	FTP Setup failed
-1	FTPRRecv() failed
-1	FTPSend() failed
-101	No URL was specified
-102	Invalid SSL protocol
-103	Invalid SSL algorithm strength
-104	Code page values are invalid for this system
-105	Unable to initialize code page description
-106	Specify both -CPR and -CPL or neither
-107	Code page translation results in unequal character counts
-108	Iconv() failed to convert all characters
-109	The move option is missing a value. The move option requires a target directory name or file name.
-110	Move option value exceeds its size limit.
-600	Internal Error: Resource Manager Error
-???	An error code returned from the server made a negative
????	Server-supplied error message

GZIP/ZLIB Adapter

Use the GZIP/ZLIB Adapter to retrieve data in file or memory formats and export data for GZIP, ZLIB and Plain file formats.

- [GZIP/ZLIB Adapter overview](#)
- [System requirements](#)
- [Command alias](#)
- [GZIP/ZLIB Adapter commands](#)
- [Syntax summary](#)
- [Command behavior](#)

GZIP/ZLIB Adapter overview

The GZIP/ZLIB Adapter commands enable you to define the type of input and output streams and the compression actions.

The following data streams can be used for input for the GZIP/ZLIB Adapter:

- GZIP
- ZLIB
- Plain

The following file formats are output as data for the GZIP/ZLIB Adapter:

- GZIP
- ZLIB
- Plain

System requirements

See the system requirements and the release notes for the GZIP/ZLIB Adapter requirements. It is assumed that a Command Server has already been installed on the computer where the adapter is to be installed for runtime purposes.

Command alias

Adapter commands can be specified by using a command string on the command line or creating a command file that contains adapter commands. The execution command syntax is:

```
-IA[alias] card_num
-OA[alias] card_num
```

where **-IA** is the Input Source Override execution command and **-OA** is the Output Target Override execution command, **alias** is the adapter alias, and **card_num** is the number of the input or output card. The following table shows the adapter alias and its execution command.

Adapter	Alias	As Input	As Output
GZIP/ZLIB	GZIP	-IAGZIPcard_num	-OAGZIPcard_num

GZIP/ZLIB Adapter commands

The following table lists valid commands for the GZIP/ZLIB Adapter, the command syntax, and whether the command is supported (✓) for use with data sources, targets, or both.

Command	Syntax	Source	Target
Action (-ACTION)	-ACTION <i>actiontype</i>	✓	✓
File (-FILE)	-FILE <i>filename</i>	✓	✓
Stream (-STREAM)	-STREAM <i>datatype</i>	✓	✓
Trace (-T)	-T[+] [<i>full_path</i>]	✓	✓

- [Action adapter command for GZIP/ZLIB](#)
- [File adapter command for GZIP/ZLIB](#)
- [Stream adapter command for GZIP/ZLIB](#)
- [Trace adapter command for GZIP/ZLIB](#)

Action adapter command for GZIP/ZLIB

Use the Action adapter command (-ACTION) to decompress or compress the data stream before processing it. If an action is not specified as an option to the -ACTION command, the default action is AUTO.

-ACTION [DECOMPRESS | COMPRESSTOGZIP | COMPRESSTOZLIB | AUTO]

Option

Description

DECOMPRESS

Specifies decompression of the data stream.

COMPRESSTOGZIP

Specifies compression of the data stream using the GZIP file format.

COMPRESSTOZLIB

Specifies compression of the data stream using the ZLIB file format.

AUTO

Automatic compression (DECOMPRESS, COMPRESSTOGZIP, or COMPRESSTOZLIB) occurs depending on the type of data stream.

For a complete list of behaviors associated with this command when used with the -STREAM command, see ["Command behavior"](#).

File adapter command for GZIP/ZLIB

Use the File adapter command (-FILE) for data sources and data targets to specify the file that contains the GZIP or ZLIB input stream or output stream.

-FILE [*file_path*] / *file_name*

Option

Description

*[file_path] / *file_name**

Specify the file from which the input stream is read or to which the output stream is written.

A relative file path is relative to the current path of the process.

To specify a file path that is relative to the map directory, use the GETDIRECTORY function to build an absolute path. See the *Functions and Expressions* documentation.

If the File adapter command (-FILE) is not specified for a data source, the memory buffer is the default data source.

The File adapter command is required for data targets.

For example, to specify a data source of **input.gz**, the following adapter command should be used:

-FILE *input.gz*

Stream adapter command for GZIP/ZLIB

Use the Stream adapter command (-STREAM) to specify the input and output data type. If a data type is not specified as an option to the stream command, the default data type is AUTO.

-STREAM [GZIP | ZLIB | PLAIN | AUTO]

Option

Description

GZIP

Specifies a data type of GZIP.

ZLIB Specifies a data type of ZLIB.

PLAIN Specifies the data stream is a sequence of bytes.

AUTO Specifies auto-detection of the data type. The use of this option should be limited because streams that are not GZIP or ZLIB can have a valid GZIP or ZLIB fingerprint.

For a complete list of behaviors associated with this command when used with the **-ACTION** command, see "[Command behavior](#)".

Trace adapter command for GZIP/ZLIB

Use the Trace adapter command (**-T**) to produce a diagnostics file. This file contains detailed information about adapter activity. By default, the **m4gzip.mtr** trace file is created in the directory where the map is located.

-T[+] *[full_path]*

Option

Description

+ Append trace information to the existing trace file.
full_path Creates a trace file with the specified name in the specified directory.

Syntax summary

Data sources

The following is the syntax of the GZIP/ZLIB Adapter commands used for data sources:

```
[-FILE filename]
[-STREAM [GZIP|ZLIB|PLAIN|AUTO]]
[-ACTION [DECOMPRESS|COMPRESSTOGZIP|COMPRESSTOZLIB|AUTO]]
[-T[+] [full_path]]
```

Data targets

The following is the syntax of the GZIP/ZLIB Adapter commands used for data targets:

```
-FILE filename
[-STREAM [GZIP|ZLIB|PLAIN|AUTO]]
[-ACTION [DECOMPRESS|COMPRESSTOGZIP|COMPRESSTOZLIB|AUTO]]
[-T[+] [full_path]]
```

Command behavior

The following commands behave in this manner when used in the following combinations:

-STREAM command	-ACTION command	Data Source	Data Target
GZIP	DECOMPRESS	Input stream is decompressed before it is passed to the map.	Data is decompressed using the GZIP file format and written to the file specified by -FILE command.
GZIP	COMPRESSTOGZIP	Not valid.	Data is decompressed using the GZIP file format and written to the file specified by -FILE command (-ACTION command is ignored).
GZIP	COMPRESSTOZLIB	Not valid.	Data is decompressed using the GZIP file format and written to the file specified by -FILE command (-ACTION command is ignored).
GZIP	AUTO	Input stream is decompressed before it is passed to the map.	Data is decompressed (inflated) and written to the file specified by -FILE command.
ZLIB	DECOMPRESS	Input stream is decompressed before it is passed to the map.	Data is decompressed (inflated) and written to the file specified by -FILE command.
ZLIB	COMPRESSTOGZIP	Not valid.	The -ACTION command is ignored. The data coming from the map is decompressed (inflated) and written to the file that is specified by the -FILE command.
ZLIB	COMPRESSTOZLIB	Not valid.	The -ACTION command is ignored. The data coming from the map is decompressed (inflated) and written to the file that is specified by the -FILE command.

-STREAM command	-ACTION command	Data Source	Data Target
ZLIB	AUTO	Input stream is decompressed before it is passed to the map.	-ACTION command is assumed to be decompressed and the data is decompressed (inflated) and written to the file specified by -FILE command.
PLAIN	DECOMPRESS	Not valid.	Not valid.
PLAIN	COMPRESSTOGZIP	Input stream is compressed using GZIP file format prior to being passed to the map.	Output data received from map is compressed and written to the file specified by -FILE command.
PLAIN	COMPRESSTOZLIB	Input stream is compressed using the ZLIB file format prior to being passed to the map.	Output data received from map is compressed and written to the file specified by -FILE command.
PLAIN	AUTO	The input stream is compressed using GZIP file format prior to be passed to the map.	Output data received from map is compressed and written to the file specified by -FILE command.
AUTO	DECOMPRESS	Type of the input stream is automatically detected and expanded by the adapter. It can be GZIP (COMPRESSTOGZIP), ZLIB (COMPRESSTOZLIB) or PLAIN (COMPRESSTOZLIB). The behavior of the adapter is in accordance with the previously specified cases.	Type of the input stream is automatically detected and expanded by the adapter. It can be GZIP (COMPRESSTOGZIP), ZLIB (COMPRESSTOZLIB) or PLAIN (COMPRESSTOZLIB). The behavior of the adapter is in accordance with the previously specified cases.
AUTO	COMPRESSTOGZIP	Type of the input stream is automatically detected and expanded by the adapter. It can be GZIP (COMPRESSTOGZIP), ZLIB (COMPRESSTOZLIB) or PLAIN (COMPRESSTOGZIP). The behavior of the adapter is in accordance with the previously specified cases.	Type of the input stream is automatically detected and expanded by the adapter. It can be GZIP (COMPRESSTOGZIP), ZLIB (COMPRESSTOZLIB) or PLAIN (COMPRESSTOGZIP). The behavior of the adapter is in accordance with the previously specified cases.
AUTO	COMPRESSTOZLIB	Type of the input stream is automatically detected and expanded by the adapter. It can be GZIP (COMPRESSTOZLIB), ZLIB (COMPRESSTOZLIB) or PLAIN (COMPRESSTOZLIB). The behavior of the adapter is in accordance with the previously specified cases.	Type of the input stream is automatically detected and expanded by the adapter. It can be GZIP (COMPRESSTOZLIB), ZLIB (COMPRESSTOZLIB) or PLAIN (COMPRESSTOZLIB). The behavior of the adapter is in accordance with the previously specified cases.
AUTO	AUTO	Type of the input stream is automatically detected and expanded by the adapter. It can be GZIP (COMPRESSTOZLIB), ZLIB (COMPRESSTOZLIB) or PLAIN (COMPRESSTOZLIB). The behavior of the adapter is in accordance with the previously specified cases.	Type of the input stream is automatically detected and expanded by the adapter. It can be GZIP (COMPRESSTOZLIB), ZLIB (COMPRESSTOZLIB) or PLAIN (COMPRESSTOZLIB). The behavior of the adapter is in accordance with the previously specified cases.

HL7 MLLP Adapter

The HL7 MLLP adapter uses the MLLP protocol over TCP/IP to pass data to and from IBM Transformation Extender. The adapter has a listener interface that can detect the arrival of data on a socket and trigger a map to run. With the HL7 MLLP adapter, remote applications can pass data to IBM Transformation Extender without using additional middleware software.

The HL7 MLLP adapter is implemented as a dynamic-link library (DLL) on Microsoft Windows platforms (m4mllp.dll) and as a shared object on UNIX platforms (m4mllp.sl or m4mllp.so). The type trees that are supplied with the IBM Transformation Extender Pack for HL7 define the message format.

The adapter supports:

- Conversations
- Client or server mode

The adapter provides these functions:

- Gets one or more messages from a socket for input to IBM Transformation Extender
- Puts a message to a socket as output from IBM Transformation Extender
- Listens for messages and triggers maps to run as messages arrive
- **System requirements**
Install a Command Server, Launcher, or other IBM Transformation Extender runtime component on the computer where the adapter is to be installed for runtime purposes.
- **How to use the HL7 MLLP adapter in a map**
You can use the HL7 MLLP adapter on input cards and output cards, and in **GET** and **PUT** map rules.
- **Conversation support**
- **HL7 MLLP adapter command overview**
This table summarizes the adapter commands, syntax, and whether you can use the command with data sources, data targets, or both. You can specify the commands on the adapter command line or through the adapter properties.

System requirements

Install a Command Server, Launcher, or other IBM Transformation Extender runtime component on the computer where the adapter is to be installed for runtime purposes.

See the [system requirements](#) for details about minimum system requirements for the HL7 MLLP adapter.

How to use the HL7 MLLP adapter in a map

You can use the HL7 MLLP adapter on input cards and output cards, and in **GET** and **PUT** map rules.

When used with data sources, the HL7 MLLP adapter listens on a port. When a client socket connects to the port, the server socket creates a connection for the client. When a connection is open, the adapter receives a message if the MLLP leading bytes are present. If the leading bytes are missing, the adapter does not accept the message. The MLLP delimiters indicate the end of the message.

When used with data targets, the adapter constructs a message by wrapping the data from the map with the MLLP leading bytes and trailing delimiters. The adapter delivers the message to the specified host on the specified port.

When used with a three-parameter **GET** rule, the adapter constructs a message, delivers it to the specified host on the specified port, and waits for an acknowledgement. The adapter returns the acknowledgement to the map as output from the **GET** rule.

The adapter reuses a connection when an output card or **PUT** function specifies the same host and port numbers as the input card or **GET** function in the same map. The adapter sends a response message to the client that sent the request message.

Conversation support

The HL7 MLLP adapter supports conversation scenarios in which:

- Input is an event
 1. The listener detects a message
 2. The **GET** rule retrieves the message
 3. An output card sends a response to the same socket
- Input is not an event
 1. The **GET** rule retrieves the message
 2. An output card sends a response to the same socket
- The map uses the **=GET()** mapping function
The third parameter of the function is the data that the adapter is to send to the socket. The **GET** function returns the resulting message.

By default, a single instance of a map has exclusive use of a client connection (**-CCON** command). With an exclusive client connection, you can use the number of event messages (**-NEM** command) to initiate a conversational map instance. The **-NEM** command specifies how many messages the adapter must receive to start an instance of a map. If the map runs in burst mode, each burst can process one or more additional messages from the client, and send one or more responses to the client. In this scenario, the map ends when the client disconnects from the socket.

When the client connection is shared, multiple map instances concurrently use a single client connection. You cannot use the **-NEM** command with a shared connection. Instead, use the Quantity (**-QTY**) and Listen (**-LSN**) commands to determine the number of messages that a single map instance processes. The listener receives the specified number of messages. The map instance processes the messages and sends one or more response messages to the client.

If the client continues to send messages while a map instance is running, the adapter can start another map instance. Both map instances use the same client connection. The client is responsible for associating the request message with any responses from the maps. The **-CCON shared** command does not preclude multiple clients from connecting to the socket.

From

- [Client connection \(-CCON\) command](#)

To

- [Client connection \(-CCON\) command](#)
- [Listen \(-LSN\) command](#)
- [Number of event messages \(-NEM\) command](#)
- [Quantity \(-QTY\) command](#)

HL7 MLLP adapter command overview

This table summarizes the adapter commands, syntax, and whether you can use the command with data sources, data targets, or both. You can specify the commands on the adapter command line or through the adapter properties.

The commands are not case-sensitive, but the command values are case-sensitive. Choices are separated by the OR symbol | . Optional parameters are enclosed in brackets []. The topics that follow provide the command syntax in railroad track format.

Table 1. HL7 MLLP adapter command summary

Command	Syntax	Use with Data Source	Use with Data Target
Client Connection	-CCON [excl shared]	Yes	No
Host	{-H -HOST} host_name IP_addr	Yes	Yes
Listen	-LSN seconds	Yes	No
Number of Event Messages	-NEM number	Yes	No
Port	{-P -PORT} port_number	Yes	Yes
Quantity	-QTY number	Yes	No
Service	{-S -SERVICE} service_name	Yes	Yes
Trace	{-T -TRACE} [E V] [+ file_name]	Yes	Yes

- [Command alias](#)

Use the MLLP adapter alias to use the HL7 MLLP adapter on input and output cards and in GET and PUT map rules.

- **[Client connection \(-CCON\) command](#)**
The client connection command specifies whether a client connection is shared between map instances or exclusively associated with a single map instance.
- **[Host \(-H\) command](#)**
The **-HOST** command specifies the host name or TCP/IP address of the remote host. The **-HOST** command is optional. If you omit it, the adapter uses the name of the local computer as the remote host name.
- **[Listen \(-LSN\) command](#)**
The **-LSN** command specifies the amount of time, in seconds, that the adapter waits for a message to arrive. This command is optional. If you omit it, the adapter waits for an infinite amount of time.
- **[Number of event messages \(-NEM\) command](#)**
Use this command to specify the threshold number of messages that triggers the Launcher to run a map. This command is optional. If you omit it, the threshold is the value specified on the **-QTY** command. The default value of the **-QTY** command is 1.
- **[Port \(-P\) command](#)**
The **-PORT** command specifies the TCP/IP port number of the adapter. In server mode, the port number specifies the port to which remote clients connect. In client mode, the port number specifies the port on the remote host to which the adapter connects. The **-PORT** command is mutually exclusive with the **-SERVICE** command.
- **[Quantity \(-QTY\) command](#)**
The **-QTY** command specifies the number of messages that the adapter is to retrieve. In burst mode, the total number of messages retrieved by the adapter does not exceed the number specified by the **-QTY** command, although a single burst might return fewer messages. The **-QTY** command is optional. If you omit it, the default number of messages is 1.
- **[Service \(-S\) command](#)**
The **-Service** command specifies the service name that is used to map a TCP/IP port number. The **-Service** command is mutually exclusive with the **-PORT** command.
- **[Trace \(-T\) command](#)**
The **-TRACE** command produces a log file of HL7 MLLP adapter activity.

Command alias

Use the MLLP adapter alias to use the HL7 MLLP adapter on input and output cards and in GET and PUT map rules.

Specify adapter commands by using a command string on the command line or by creating a command file that contains adapter commands. The command syntax is:

```
-IA[alias]card_num
-OA[alias]card_num
```

where:

-IA	The Input Source Override execution command
-OA	The Output Target Override execution command
<i>alias</i>	The adapter alias
<i>card_num</i>	The number of the map card

To override an input card:

```
-IAMLLPcard_num
```

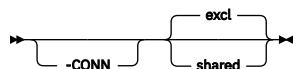
To override an output card:

```
-OAMLLLPcard_num
```

Client connection (-CCON) command

The client connection command specifies whether a client connection is shared between map instances or exclusively associated with a single map instance.

-CCON command syntax



excl

The client connection is exclusive to a particular map instance. When a client connects to an adapter that is running in server mode, the connection is associated with the map instance until the client disconnects. The map can receive multiple requests from the client and return multiple responses. Use this option in an environment where the client connects to the socket, engages in a conversation, and then disconnects.
This is the default client connection mode.

shared

The client connection can be shared by multiple map instances. You can configure the adapter to run a map after the number of messages specified by the **-QTY** command have been received, or when the listening time specified by the **-LSN** command has elapsed. Use this option in an environment where the client connects to the socket and remains connected, feeding a continuous stream of messages to the Launcher.

To

- [Conversation support](#)

- [Listen \(-LSN\) command](#)
- [Quantity \(-QTY\) command](#)

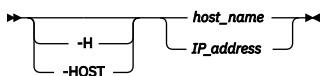
From

- [Conversation support](#)

Host (-H) command

The **-HOST** command specifies the host name or TCP/IP address of the remote host. The **-HOST** command is optional. If you omit it, the adapter uses the name of the local computer as the remote host name.

-HOST command syntax



host_name

Specifies the name of the remote host.

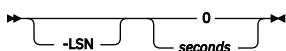
ip_address

Specifies the IP address of the remote host. You can use an Internet Protocol version 4 (IPv4) address or an Internet Protocol version 6 (IPv6) address. For example, use 192.168.1.120 as the IPv4 address or 1080:0:0:0:8:800:200C:417A as the IPv6 address.

Listen (-LSN) command

The **-LSN** command specifies the amount of time, in seconds, that the adapter waits for a message to arrive. This command is optional. If you omit it, the adapter waits for an infinite amount of time.

-LSN command syntax



0

The adapter does not wait. If no message is available, the adapter returns a warning that no messages were found.

seconds

Specifies the amount of time, in seconds, that the adapter is to wait for a message to arrive. If a message does not arrive within the specified timeframe:

- In Command Server mode, the adapter returns a warning.
- In Launcher mode, the Launcher continues to retry.

From

- [Client connection \(-CCON\) command](#)

To

- [Conversation support](#)

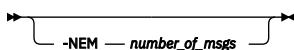
From

- [Conversation support](#)

Number of event messages (-NEM) command

Use this command to specify the threshold number of messages that triggers the Launcher to run a map. This command is optional. If you omit it, the threshold is the value specified on the -QTY command. The default value of the -QTY command is 1.

-NEM command syntax



number_of_msgs

The number of messages that triggers the Launcher to run a map.

To

- [Listen \(-LSN\) command](#)
- [Quantity \(-QTY\) command](#)
- [Conversation support](#)

From

- [Conversation support](#)

Port (-P) command

The **-PORT** command specifies the TCP/IP port number of the adapter. In server mode, the port number specifies the port to which remote clients connect. In client mode, the port number specifies the port on the remote host to which the adapter connects. The **-PORT** command is mutually exclusive with the **-SERVICE** command.

-PORT command syntax

```
► [ -P ] port_number ►  
  [ -PORT ]
```

port_number
The TCP/IP port number.

To

- [Service \(-S\) command](#)

Quantity (-QTY) command

The **-QTY** command specifies the number of messages that the adapter is to retrieve. In burst mode, the total number of messages retrieved by the adapter does not exceed the number specified by the **-QTY** command, although a single burst might return fewer messages. The **-QTY** command is optional. If you omit it, the default number of messages is 1.

-QTY command syntax

```
► [ -QTY ] [ 1 ] number_of_msgs ►  
  [ -QTY ]
```

number_of_msgs
The number of messages that the adapter is to retrieve. In burst mode, *number_of_msgs* specifies the total number of messages that the adapter is to retrieve. A single burst might contain fewer messages.

From

- [Client connection \(-CCON\) command](#)

To

- [Conversation support](#)

From

- [Conversation support](#)

Service (-S) command

The **-Service** command specifies the service name that is used to map a TCP/IP port number. The **-Service** command is mutually exclusive with the **-PORT** command.

-SERVICE command syntax

```
► [ -S ] service_name ►  
  [ -SERVICE ]
```

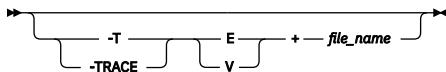
service_name
The service name that is used to map a TCP/IP port number.

From

- [Port \(-P\) command](#)

Trace (-T) command

The **-TRACE** command produces a log file of HL7 MLLP adapter activity.



E

Creates a trace file that contains only the adapter errors that occurred during map execution.

V

Specifies verbose mode. The log file records all activity that occurs while the adapter is retrieving data. The default maximum size of the data that is to be logged is 256 bytes. To trace more or fewer bytes, set the TX_SOCKET_MAX_BYTES_IN_TRACE_MESSAGE environment variable.

+

Appends the trace information to the existing log file. Omit this keyword to create a new log file.

path/file_name

The file name and, optionally, path to the log file. If the file name contains spaces or special characters, enclose the file name in quotation marks (for example, "test-file.txt"). The default log file name is m4mllp.log. By default, the log file is created in the directory where the map is located.

HTTP Adapter overview

The Hypertext Transfer Protocol (HTTP) Adapter provides support for Proxy servers and Secure Socket Layer (SSL) protocol, which is required to process HTTPS (secure) URLs.

Use the HTTP Adapter to transfer data to and from an HTTP(S) Web server. You can transfer data from a Command Server, Launcher, or Software Development Kit on one platform to an HTTP(S) Web server on another platform.

- [HTTP Adapter Introduction](#)

Use the HTTP Adapter to transfer data to and from an HTTP(S) Web server.

- [HTTP Adapter commands](#)

- [Syntax summary](#)

- [Troubleshooting](#)

- [Return codes and error messages](#)

HTTP Adapter Introduction

Use the HTTP Adapter to transfer data to and from an HTTP(S) Web server.

System requirements

The release notes detail the HTTP adapter's minimum system requirements and operating system requirements. For runtime purposes, you must have already installed a Command Server on the computer.

Additional requirements for the HTTP Adapter:

- TCP/IP must be installed on the machine where you install the HTTP Adapter.
- The machine where the HTTP Adapter is installed must be able to access the HTTP(S) server over a TCP/IP network.

Automatic HTTP event management

The HTTP Adapter allows the Launcher to directly process HTTP input events without compromising the integrity of the firewall.

Command Alias

You can specify adapter commands by using a command string on the command line or by creating a command file that contains adapter commands. The command syntax is:

```
-IA[alias]card_num
-OA[alias]card_num
```

In the command syntax, **-IA** is the Input Source Override execution command, and **-OA** is the Output Target Override execution command, *alias* is the adapter alias, and *card_num* is the number of the map card. The HTTP Adapter alias and corresponding execution commands are listed below.

Adapter	Alias	As Input	As Output
HTTP	HTTP	-IAHTTP <i>card_num</i>	-OAHHTTP <i>card_num</i>

Known limitations

The HTTP Adapter does not support:

- Document caching
- Content Codings such as x-gzip, x-compress, or deflate
- Multiparts and multipart/byte ranges
- #markers

HTTP Adapter commands

The following table lists valid commands for the HTTP Adapter, their syntax, and whether they are supported (✓) for use with data sources, targets, or both.

Name	Syntax	Source	Target
Client Certificate	-CERT {label}	✓	✓
From	-FROM address	✓	✓
Header	-HDR [+][I]	✓	✓
HTTP(S)	-URL {HTTP HTTPS}://[user[:pass]@] [host[:port]]/[dir]/[file]	✓	✓
Ignore	-IGNORE error_code[:error_code...]	✓	✓
Inline Output	-INLINE		✓
Listen	-LSN seconds	✓	
Method	-METHOD method_name	✓	✓
Modified Since	-MODIFIED http-date	✓	✓
Proxy URL	-PROXY {HTTP HTTPS}://[user[:pass]@] [host[:port]]/[dir]/[file]	✓	✓
Redirections Allowed	-REDIRECT limit	✓	✓
Session	-SESSION session_id [*]	✓	✓
Server Name Indication	-SNI	✓	✓
SOCKS URL	-SOCKS {socks4 socks5}://[user[:pass]@] [host[:port]]	✓	✓
SSL Encryption Strength	-STR {WEAK STRONG ANY}	✓	✓
SSL Protocol	-SPROTO {SSLv2 SSLv3 SSLv23 TLSv1}	✓	✓
Status Code	-STATUS_CODE {map_return_code=HTTP_status_code[,map_return_code=HTTP_status_code...]} map_return_code:map_return_code[:map_return_code...]=HTTP_status_code	✓	
Status Line	-STATUSLINE	✓	
Status Page	-STATUS_PAGE file_name[*].file_type	✓	
Status Page Default	-STATUS_PAGE_DEFAULT file_name,file_type	✓	
Timeout	-TIMEOUT time_in_seconds	✓	✓
Trace	-T[E][+][S V] [full_path]	✓	✓
Tunneling	-TUNNELING	✓	✓
Type	-TYPE content_type [charset=encoding]	✓	✓

Client Certificate

Specifies the label of the client certificate in the key store. This command is applicable only when the **SSL_CLIENT secure_mode** setting in the config.yaml file is set, and SSL/TLS encryption is in effect.

The corresponding adapter command is **-CERT {label}**.

label: The name of the entry in the certificate store, which is specified by the `key_store` field in the config.yaml file.

Note: The certificate store contains all personal and trusted certificates in a PKCS12 format file. To see the examples on how to import private keys and certificate chains into this encrypted file, refer to the `readme.txt` file located in the `<Design Studio installation folder>|examples|secureadapt|ssl` folder.
From

Specifies the email address of a contact person. The adapter sends alerts to the contact person in case of a problem using requests.

The corresponding adapter command is **-FROM**.

-FROMaddress

address: Specifies the e-mail address of a contact person

Header

This command is used to send message headers from data in a map file or to return message headers to the map file. In addition to generating new headers, you can append user-supplied headers, eliminating duplicating all adapter-generated headers to add a new header.

Note: The header and body data must be separated by `<CR><LF><CR><LF>` when sent to the HTTP Adapter. The **Header** and **Headers** groups are in the **http.mtt** example type tree provides a clear illustration of this concept. Each header is terminated by `<CR><LF>`, and the Headers group is terminated by another `<CR><LF>`.
The corresponding adapter command is **-HDR**.

For example:

-HDR[+][I]

+> Appends user-supplied headers. It eliminates the need to duplicate the existing adapter-generated headers by adding a new one.

I-> In a request-reply scenario, indicates headers are provided in the input request but should not be returned in the response.

See also: [HTTP Header Adapter Command \(-HDR\) Example](#).

[HTTP\(S\) URL](#)

This property establishes a connection between the adapter and the HTTP(S) server. Specify the URL to connect to the HTTP(S) server. You must also specify the name of the file you want to transfer.

The corresponding adapter command is **-URL**.

For example:

-URL {HTTP|HTTPS}://[user[:pass]@[host[:port]]]/[dir[/file]]

HTTP: The HTTP application protocol.

HTTPS: The HTTPS application protocol.

Note: This option is only available if the Security Option is installed.

user: Specifies the user name required to connect to the HTTP(S) server. Unless you specify a password, this must be followed by @ if you specify the host.

:pass: Specifies the password required that authenticates the user name. This parameter must be followed by @ if you also specify the host.

host: The HTTP(S) server name or address.

:port: The HTTP(S) server port name or number.

/dir: The directory path for the file.

/file: The file that the HTTP adapter retrieves.

Note: The user and pass arguments extend what the RFC supports for a URL path.

The URL does not need to be completely 'URL-encoded'. As shown in the following table, you only need to use escape characters for tokens specifically used in a URL.

Replace the tokens with escape characters respectively:

Table 1. The following tokens are replaced with escape characters, respectively.

Tokens	Escape characters
% (percent)	%25
@ (at)	%40
/ (forward slash)	%2F
:	%3A
;	%3B
& (ampersand)	Escape is not needed; use it as is.
# (pound)	Escape is not needed; use it as is.
8-bit or control characters	Escape is not needed; use it as is.

Any URL that contains spaces must be enclosed in quotes. For example:

-URL HTTP://Home Machine/HTTP Fil

Ignore

This property is used to ignore specific error codes. The corresponding adapter command is **-IGNORE**.

-IGNORE error_code[:error_code]

error_code: Specifies the HTTP error codes that are to be ignored by the adapter. A colon (:) must separate multiple error codes.

For example, the results for the following command **-I 404:410** is:

- The adapter ignores errors of type 404 (not found).
- The adapter ignores errors of type 410 (gone).

Inline Output

This property is used to append data to the end of a URL. The corresponding adapter command is **-INLINE**.

You can use this command to submit a long query string to a form that uses `method=GET`. In the example below, the `GET` function has three arguments. The third argument is optional.

GET ("HTTP", "-URL HTTP://[www.yahoo.com/cgi-bin/search?]" - INLINE", "widgets")

Result: Gets the URL `http://www.yahoo.com/cgi-bin/search?widgets`.

KEEPALIVE

This command limits the connection reuse time of an HTTP adapter to prevent the reuse of closed connections. For example, when the connection exceeds the HTTP server KeepAlive Timeout setting.

A map reuses a connection under two conditions:

- A running map reuses an *active* HTTP connection when it issues multiple HTTP requests with the same command-line connection options within the same map instance.
- A map also reuses an *idle* HTTP connection when it runs multiple times in different map instances.

The corresponding adapter command is **-KEEPALIVE**.

-KEEPALIVEtimeout

timeout: The number of seconds that can elapse between consecutive reuses of a connection. After the timeout is exceeded, the HTTP adapter does not reuse the connection. Valid values are 0 - 99999. A 0 value specifies no connection reuse.

Example

A map issues multiple HTTP requests and runs multiple times. The HTTP server has a KeepAliveTimeout of 5 seconds. The config.yaml file has an IdleHTTP=10 /runtime/Connections Manager/HTTP/idle setting.

The map processing sequence is as follows:

1. A new instance of Map 1 starts.
2. Map 1, Input Card 1, iteration 1.
3. Map 1, Output Card 1, iteration 1 runs.
 - a. The HTTP adapter establishes **Connection 1** or reuses a connection from a prior map with similar HTTP settings.
 - b. The map issues an HTTP Post operation.
4. Map 1, Input Card 1, iteration 2.
5. Map 1, Output Card 1, iteration 2 runs.
 - a. The HTTP adapter reuses **Connection 1**.
 - b. The map issues an HTTP Post operation.
6. Map 1 is completed.
7. Map 1 runs again with a new instance number and repeats step 1.

Without the **-KEEPALIVE** command, consider these potential failure scenarios:

- More than 5 seconds elapse between HTTP POST operations (steps 3b and 5b):
 - The HTTP server KeepAliveTimeout closes the connection.
 - The HTTP adapter attempts to use the active (but closed) connection in step 5b, and the map fails.
- Less than 10 seconds elapse between step 5b in the first map instance and step 3a in the new map instance:
 - The HTTP adapter reuses the idle connection.
 - Step 3b activates the reused connection.
 - If more than 5 seconds elapse between HTTP POST operations (steps 3b and 5b), the HTTP server KeepAliveTimeout closes the connection. The map fails when the HTTP adapter attempts to use the active (but closed) connection in step 5a.

The **-KEEPALIVE 5** adapter command prevents the HTTP adapter from reusing a connection if more than 5 seconds elapsed since the last time the map used the connection. Consider these successful scenarios:

- More than 5 seconds elapse between HTTP POST operations (steps 3b and 5b):
 - The HTTP server KeepAliveTimeout closes the active connection.
 - The **-KEEPALIVE 5** adapter command prevents the HTTP adapter from reusing the active (but closed) connection.
 - The HTTP adapter establishes a new connection, and the map completes successfully.
- The connection's idle time between map instances (steps 6 and 7) is between 5 - 9 seconds:
 - The idle time is less than 10 seconds, so the idle connection qualifies for reuse.
 - The time between requests exceeds the server KeepAliveTimeout value. The HTTP server closes the connection.
 - The **-KEEPALIVE 5** adapter command prevents the HTTP adapter from reusing the idle (but closed) connection in step 3a.
 - The HTTP adapter establishes a new connection, and the map completes successfully.

Listen

Specifies a time (in seconds) to wait for messages to arrive. Suppose the URL does not become available in the specified listen time. In that case, you will receive an error message, Source not available.

The corresponding adapter command is **-LSN**.

-LSNseconds

seconds: The number of seconds to wait for messages to arrive.

The command **-LSN 0** means do not wait at all. If no message is immediately available, the adapter returns a warning indicating no messages were found.

Method

Specifies the transfer method to use in an HTTP request. The HTTP request method can be any request the HTTP(S) server can process. Typical method names are PUT, GET, and POST. If the transfer is not specified, the default value for **-METHOD** is GET for input cards (to retrieve data) and POST for output cards (to store data).

The corresponding adapter command is **-M** or (**-METHOD**).

-METHODmethod_name

method_name: Transfer method used in an HTTP request.

Modified Since

This command generates the *If-Modified-Since* header information used during a GET request. The HTTP Adapter then adds the date to the request.

The corresponding adapter command is **-MOD** Or (**-MODIFIED**).

-MODIFIEDhttp-date

http-date: Dates supported by *RFC 1123* and *RFC 850*. The date can also be in the form of {*wkday date time*}, where:

- *wkday* is Mon | Tue | Wed | Thu | Fri | Sat | Sun
- *date* is month day (for example, JUN 06)
- *time* is 00:00:00 to 23:59:59

Proxy URL

This property specifies the URL to connect to a proxy server and a file name for transferring the file.

The corresponding adapter command is **-PROXY**

For example:

-PROXY {HTTP|HTTPS}://[user[:pass]@[host[:port]]]/[dir[/file]]

HTTP: The HTTP application protocol.

HTTPS: The HTTPS application protocol.

user: Specifies the user name required to connect to the proxy server. Unless you specify a password, this must be followed by @ if you specify a host.

:pass: Specifies the password required that authenticates the user name. This parameter must be followed by @ if you specify the host.

host: The proxy server name or address.

:port: The proxy server port name or number.

/dir: The directory path for the file.

/file: The file that the HTTP Adapter retrieves.

Redirections Allowed

This property specifies the maximum number of redirections or file-moved messages allowed before the transfer request fails. The request fails if the adapter cannot locate the file within the limit. Use a value of 0 to prevent the adapter from following redirections.

The corresponding adapter command is **-REDIRECT**.

-REDIRECT*limit*

Session

This property is used to coordinate the request and response of an HTTP. The corresponding adapter command is **-SESSION**.

-SESSION *session_id*[*]

session_id: Specifies the session identifier used for matching purposes. When receiving the HTTP request in a map, this contains the agent-assigned identifier. This identifier must be returned in the HTTP response.

*-> Permits all HTTP session IDs to be passed from the input card to the output card. You must enter a space between the **-SESSION** command and the asterisk (*) option for it to process correctly.

Generally, this command is written as:

-url *localhost:8080/example -session * -type text/html -tv+*

This command specifies the following:

- Specifies the entry to search for the HTTP server called *localhost* on port 8080 and request the */example* page.
- Allows all session id's to be passed from the input card to the output card
- Specifies *text/html* as the MIME Content-Type for the file used in the adapter request
- Enables a verbose trace file called **m4http.mtr** in append mode

SOCKS URL

This property is used to connect to a SOCKS proxy server. The corresponding adapter command is **-SOCKS**

Generally, the command is written as:

-SOCKS {**socks4|socks5**}://[**user[:pass]@[host[:port]]**]

socks4: The SOCKS protocol version 4.

socks5: The SOCKS protocol version 5.

user: Specifies the user name required to connect to the SOCKS server.

pass: Specifies the password required that authenticates the user name.

host: Specifies the server name or address of SOCKS.

port: Specifies the server port name or number of SOCKS.

Examples to connect to a SOCKS:

- To connect to a SOCKS server named **myhost** using SOCKS protocol version 4, use the command:
-SOCKS socks4://myhost
- To connect to a SOCKS server named **hmss** with a user name of **jbond** and password of **007** using SOCKS protocol version 5, use the command:
-SOCKS socks5://jbond:007@hmss

SSL Encryption Strength

Specifies the encryption strength for FTP connections. It can be set to either WEAK, which uses only weak algorithms, or STRONG, which uses only strong algorithms. If not specified, any available algorithm is used. If STRONG is unavailable, then WEAK is used. However, if the export version of the libraries is used, only weak algorithms are available, and this command is ignored.

This command is applicable only when the SSL **secure_mode** setting is in the config.yaml configuration file (/runtime/secure_mode) is set to 0, and RSA-based SSL encryption is in effect.

The corresponding adapter command is **-STR**.

-STR{WEAK/STRONG/ANY}

WEAK-> Use strong (non-exportable) encryption only.

STRONG-> Use exportable encryption only.

ANY-> Use any available encryption.

SSL Protocol

Specifies setting the Secure Socket Layer (SSL) protocol level. SSL enables the adapter to process FTPS URLs. The corresponding adapter command is **-SPROTO**.

-SPROTO{SSLv2/SSLv3/SSLv23/TLSv1}.

SSLv2: Specifies SSL protocol version 2.

SSLv3: Specifies SSL protocol version 3.

SSLv23: Specifies SSL protocol version 2, version 3, *TLSv1*, *TLSv1.1* or *TLSv1.2*. Whenever the protocol method of *SSLv23* is specified, the protocol of first choice is *TLSv1.2*, with a fallback to *TLSv1.1*, then to *TLSv1*, then to *SSLv3*, and finally to *SSLv2*, depending on the GSKit module that is loaded.

TLSv1: Specifies Transport Layer Security (TLS) protocol version 1.

If *-SPROTO* is not specified, the SSL Protocol defaults to *SSLv23*.

This command is applicable only when the SSL **secure_mode** setting is in the config.yaml configuration file (/runtime/secure_mode) is set to 0 and RSA-based SSL encryption is in effect. An SSL **secure_mode** setting of 1 or 2 supersedes this command.

Status Code

Returns an HTTP status code through the HTTP listener to the external HTTP client that requested a particular URL. The HTTP status code is associated with one or more map return codes. You can customize the HTTP status codes or use the default codes defined in the /runtime/M4HTTP/StatusCode path of the config.yaml configuration file. The corresponding adapter command is **-STATUS_CODE**.

Unable to insert image

- *map_return_code*: The numeric code indicating the map execution result. See the Map Designer description of map execution and warning messages for a list of execution codes.
- *HTTP_status_code*: An official HTTP status code maintained by the Internet Assigned Numbers Authority (IANA).

To specify multiple p return code/HTTP status code combinations on a single **-STATUS_CODE** keyword, separate each combination with a comma (,) character. For example:

-STATUS_CODE "30 = 400 Bad Request, 12 = 404 Not Found"

To link multiple map return codes to a single HTTP status code, delineate each map code with a colon (:) character. For example:

-STATUS_CODE "8:9:30 = 400 Bad Request"

Status Line

Allows the initial HTTP response line to be returned to the map. The response line will be pre-pended to the regular data returned by the HTTP Adapter.

If the *-STATUSLINE* command is specified on the command line, the HTTP Adapter behaves like the Header (*-HDR*) command was also specified.

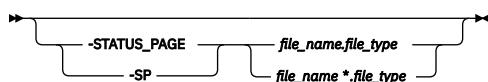
Status line is an optional command. The corresponding adapter command is **-STATUSLINE**.

For example, if you specify: **-URL http://www.google.com -STATUSLINE**

The command would return the complete HTTP response from Google, including HTTP headers and the HTTP Response status as the first line.

Status Page

Returns a text file containing the execution results of a session-specific map to an external HTTP client. The map can dynamically construct the file using predefined variables that resolve at map run time. The corresponding adapter command is **-SP** or **-STATUS_PAGE**.



file_name,file_type

file_name.file_type*

The file name of the status page file that the HTTP listener returns to the HTTP client. If the file name is not qualified, the map directory is used as the file path.

Optionally, you can generate a session-specific file name by appending an asterisk (*) wildcard character at the end of the file name. The wildcard character resolves to the Launcher session ID and timestamp.

Status page variables

With predefined status page variables, a map can dynamically create the status page the HTTP listener returns to the external HTTP client. When the map runs, the variables in the status page resolve to map execution details such as error code and map name. The variables are as follows:

- %TX_ADAPTER_STATUS_CODE%: The selected status code from the "-status_code" option.
- %TX_MAP_ERROR_CODE%: The map error code.
- %TX_MAP_ERROR_MSG%: The map error message.
- %TX_MAP_NAME%: The map name
- %TX_MAP_INSTANCE%: The map instance number
- %TX_MAP_SESSION_ID%: The unique ID assigned by the HTTP listener to this instance of the triggered input card

- %TX_TIMESTAMP%: The timestamp when the HTTP listener sends the response to the external HTTP client

Status page files for multi-threaded maps

To support multi-threaded maps, append an asterisk (*) character to the file name of the HTTP adapter status page. The asterisk is a wildcard character that generates a unique file name for the status page to ensure that other instances of the same map do not overwrite the file contents.

The wildcard character resolves to `session_ID`, `timestamp`. The session_ID is a unique alphanumeric string that the HTTP listener associates with an instance of a triggered input card.

For example, a multi-threaded map's input card can include the following command:

```
-status_page http_response_*.html
```

When the map runs, the wildcard character resolves to a unique file name, such as:

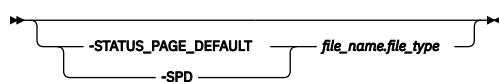
```
http_response_NEED-SESSION-ID,NEED-TIMESTAMP-FORMAT.html
```

Status Page Default

Specifies a backup status page to be returned to an external HTTP client when the page specified by the `-STATUS_PAGE` option fails. The `/runtime/M4HTTP/StatusPage` path of the config.yaml configuration file defines a single default status page.

The corresponding adapter command is `-STATUS_PAGE_DEFAULT`. This option allows you to create a custom default page for each watch.

`-STATUS_PAGE_DEFAULT` is optional. If you omit it, the HTTP listener returns the default status page defined in the `/runtime/M4HTTP/StatusPage` path of the config.yaml configuration file.



`file_name,file_type`

`file_name*.file_type`

The file name of the status page file that the HTTP listener returns to the HTTP client. If the file name is not qualified, the map directory is used as the file path.

Timeout

This property sets the socket timeouts to a specified number of seconds. The default timeout is 300 seconds.

The corresponding adapter command is `-TIMEOUT`.

-TIMEOUT time_in_seconds

`time_in_seconds`: Specifies the number of seconds for the sockets timeout to be set. The default timeout is set to 300 seconds.

Trace

Produces a diagnostics file that contains detailed information about HTTP Adapter activity. The corresponding adapter command is `-T` or `(-TRACE)`.

The default filename is `m4http.mtr`, created in the map directory unless otherwise specified.

Generally, the command is written as:

-T[E][+][S/V] [full_path]

E: Produces a trace file containing only the adapter errors during map execution.

+: Appends trace information to the existing log file.

S: Summary mode. Default value. Record only minimal information in the log file.

V: Verbose mode. Record all activity occurring while the adapter retrieves data in the log file. If not specified, summary mode is assumed.

full_path: Creates a trace file with the specified name in the directory.

Note: Because summary mode is the default, using `-TRACES` provides the same output as `-TRACE`.

Note: You can dynamically override the adapter command line trace options using the Management Console. See "Dynamic Adapter Tracing" in the *Launcher* documentation for detailed information.

Tunneling

Establishes a clean TCP connection to the proxy server and then upgrades to SSL using the `CONNECT` method. This command should only be used in cases where the specified proxy URL is non-secure (http), and the request URL is secure (https). The corresponding adapter command is `-TUNNELING`.

Generally, the command is written as:

-url <https://secure-server:8080/example> -proxy <http://user@proxy-server:2888> -tunneling

This command specifies:

- The entry to search for the HTTPS server called a `secure-server` on port `8080` and request the `/example` page.
- The proxy server called a `proxy-server` with a plain (non-secure) HTTP connection.
- The tunneling mechanism should be used to upgrade the non-secure proxy connection to a secure one for accessing the given URL.

Type

Specifies the type of content that is contained in the body of the HTTP request. The default type is `application/octet-stream`. See "Known Limitations" for additional information. The corresponding adapter command is `-TY` or `(-TYPE)`.

When you send a SOAP-based HTTP request, encoding the HTTP request's content might require you to specify the charset parameter of the **-TYPE** command. The default encoding is ISO-8859-1, as originally specified by RFC 2616. The SOAP adapter requires a UTF-8 encoded response from the HTTP server. However, the request can be in any Western or UTF-8 encoded format.

-TYPE *content_type* [**charset=encoding**]

- *content_type*: The default content type is **application/octet stream**. You might use other types. However, do not use compound types such as multipart.
- **charset**: Specifies the encoding of the content in the HTTP request's body. The charset keyword is optional. If you omit it, the default encoding is ISO-8859-1.

When you specify the charset keyword on a SOAP adapter command line, enclose both the *content_type* and the charset keyword and value pair within two sets of double quotation marks (" "**content_type**
charset=encoding" ") to avoid parsing errors.

The following command line notifies the HTTP Server that the content is UTF-8 data:

-TYPE "text/xml; charset=utf-8"

The following command line uses a SOAP adapter GET request and notifies the HTTP Server that the content is UTF-8 data. Note that both the *content_type* and the **charset** keyword and value pair are enclosed within two sets of double quotation marks:

```
=GET("SOAP", "-T -SA http://localhost/WebService/test -TRANSPORT 'HTTP(-HDR+ -T -TYPE ""text/xml; charset=utf-8"" -METHOD POST -URL http://localhost/WebService/test)' ", PACKAGE(SoapRequest))
```

Syntax summary

Data sources

The following is the command syntax of the HTTP Adapter commands used for data sources:

```
-URL {HTTP|HTTPS}://[user[:pass]@][host[:port]][/dir[/file]]  
[-CERT {label}]  
[-FROM address]  
[-HDR[+] [I]]  
[-IGNORE error_code[:error_code]]  
[-INLINE]  
[-LSN seconds]  
[-METHOD [method_name]]  
[-MODIFIED http-date]  
[-PROXY {HTTP|HTTPS}://[user[:pass]@][host[:port]][/dir[/file]]]  
[-REDIRECT limit]  
[-SESSION session_id [*]]  
[-SOCKS {socks4|socks5}://[user[:pass]@][host[:port]]]  
[-SPROTO {SSLv2|SSLv3|SSLv23|TLSv1}]  
[-STATUSLINE]  
[-STR {WEAK|STRONG|ANY}]  
[-TIMEOUT time_in_seconds]  
[-T[E] [+][S|V] [full_path]]  
[-TUNNELING]  
[-TYPE content_type [charset=encoding]]
```

Data targets

The following is the command syntax of the HTTP Adapter commands used for data targets:

```
-URL {HTTP|HTTPS}://[user[:pass]@][host[:port]][/dir[/file]]  
[-CERT {label}]  
[-FROM address]  
[-HDR[+] [I]]  
[-IGNORE error_code[:error_code]]  
[-INLINE]  
[-LSN seconds]  
[-METHOD [method_name]]  
[-MODIFIED http-date]  
[-PROXY {HTTP|HTTPS}://[user[:pass]@][host[:port]][/dir[/file]]]  
[-REDIRECT limit]  
[-SESSION session_id [*]]  
[-SOCKS {socks4|socks5}://[user[:pass]@][host[:port]]]  
[-SPROTO {SSLv2|SSLv3|SSLv23|TLSv1}]  
[-STR {WEAK|STRONG|ANY}]  
[-TIMEOUT time_in_seconds]  
[-T[E] [+][S|V] [full_path]]  
[-TUNNELING]  
[-TYPE content_type [charset=encoding]]
```

Using HTTP Adapter commands

Use the HTTP Adapter commands from the Map Designer or Integration Flow Designer as the data source of an input map card and as the data target for an output map card. You can also use the adapter as part of the M4HTTP_OPT environment variable, on the command line, or in a map rule using GET, PUT, or RUN functions.

The HTTP Adapter requires the **-URL** command in the value field of the GET>Source>Command and PUT>TARGET>Command settings. See the *Resource Adapters* documentation for general information about commands.

Example files

An example of this adapter is located at *install_dir/examples/adapters/http*.

Using the Map Designer, select HTTP from the *Source* setting in an Input Card dialog. In the GET Source>Command field, type the **-URL** command to specify the uniform resource locator (URL), specifying the HTTP server, the file to transfer, and other required adapter commands. In this input card example, the HTTP Adapter retrieves the

test.htm file from the host server **puff**:

```
-URL HTTP://user1:test@puff:8080/c:/test/tst.htm
```

Using the Integration Flow Designer to override the HTTP Adapter settings for an input card, enter the following for the Source>GET>Source>Command setting in the Launcher settings:

```
-TV -REDIR 3 -URL HTTP://user1:pass@puff:8080/c:/orb/original.htm
```

The adapter specifies the **original.htm** file to be used as the data source. The **-TV** adapter command specifies that the adapter generates a verbose trace file containing a log of all activity and events occurring while the adapter retrieves data. The **-REDIR 3** adapter command specifies that the HTTP Adapter will follow three file redirections (or "document moved" messages) before the file transfer request fails.

Using the Input Source Override - Application Execution Command (**-IA**), this command string example can be used to override the adapter commands defined in output card 1:

```
-IAHTTP1 '-URL HTTP://user1:test@puff:8080/c:/test/test.htm'
```

This example command string causes the file **test.htm** to be retrieved from the host server **puff**.

Use the **M4HTTP_OPT** environment variable to specify commands to the HTTP Adapter or a command file containing adapter commands.

Note: Commands specified in the input or output card settings override commands specified in the environment variable.

The format for specifying commands is:

M4HTTP_OPT=adaptercommands

The *adapter commands* are the HTTP Adapter commands in the list of commands.

The format for specifying a command file is:

M4HTTP_OPT=@filename.txt

Where *filename.txt* is the file containing HTTP Adapter commands, always precede the command file name with @.

HTTP header adapter command (-HDR) example

Process cookies by mapping **Set-Cookie** headers of **Cookie** headers.

This example uses the Header adapter command (-HDR), **Http.mtt** type tree, and the **HTTPGetCookie** map. It retrieves a form that sets cookies and then submits the form. The form includes user name and password fields typical of the forms found on many websites and uses cookies to maintain this login information.

In the input card, the GET > Source > Command in the Source setting uses the **-HDR** command to map the cookies in the **Set-Cookie** headers on the form that is returned.

In the output card, the PUT > Target > Command in the **Target** setting manages the cookie state information by mapping the **Set-Cookie** headers in the form of **Cookie** headers. The **-HDR** command indicates that the user is providing the headers; the target is form data submitted with:

Content-Type application/x-www-form-urlencoded

See *RFC 2109* for more information about cookies. For more information about submitting forms and the **x-www-form-urlencoded** format, see Section 17 of the HTML 4.01 specification.

Create a type tree

In this example, the group type **Message** contains the components needed for the **HTTPGetCookie** map, and the group type **Header(s)** contains the item types **HeaderName** and **HeaderText**.

Configure the Map Input and Output Cards

The **HTTPGetCookie** map input and output cards specify the **http.mtt** file as its type tree in the **TypeTree** setting. The input and output cards use the group type **Message** from that type tree in the **TypeName** setting. Both cards also use the **-HDR** command to specify that the map will operate on headers in the HTTP path in the **-URL** command.

For example, the GET > Source > Command in the **Source** setting of the input card is:

```
-HDR -URL HTTP://www.ilost.com/home/login/1,23456,,00.html -TV
```

The **PUT > Target > Command** in the **Target** setting of the output card is:

```
-HDR -URL HTTP://www.ilost.com/user_registration/manual_login/process_sign_in/1,23456,,00.html -TV
```

Enter the map rules

The **HTTPGetCookie** map output card specifies a functional map named **xCookieExtractor**, which searches for **Set-Cookie** headers.

If a **Set-Cookie** header is found, the map changes it to **Cookie** and submits cookie parameters such as a username and password.

Troubleshooting

For error codes and messages the adapters return, see [Return Codes and Error Messages](#).

Various troubleshooting tools are available if you encounter problems using the HTTP Adapter for data sources or targets for a map. If you run a map that uses the adapter and encounter problems or do not receive the expected results, use the adapter trace file (.mtr) as a troubleshooting tool.

Adapter trace file

The adapter trace file contains detailed information provided by the adapter and records its actions, such as connections established and statements executed. This trace file is generated during the adapter's execution and can be used as a debugging aid. To produce a trace file for the adapter, use the adapter command **-TRACE**. The default adapter trace file name is located in the map directory. See Trace (**-T** or **-TRACE**) for more information.

The adapter trace file contains detailed information the adapter provides, which records the actions, such as connections established and statements executed. The trace file is produced during the adapter execution and can be used as a debugging aid. To produce a trace file for the adapter, use the adapter command **-TRACE**. The default adapter trace file name is located in the map directory. See Trace (**-T** or **-TRACE**) for more information.

Sample adapter trace file

```
HTTP Adapter, Version n.n(nn)
Copyright © 2006, IBM Corporation. All rights reserved
Built for WIN32 - INTEL on Apr 9 2004 at 19:54:37.
Command Server is 'install_dir\dstxcmdsv.exe'.
Retry Count is 0. Retry Interval is 0. OnFailure is Commit.
Fetch Unit is 0. Card Mode is Integral.
Run Started at 09:33:07.685 on 04/14/04.
Options used: audit,trace,name,url
Using http: protocol.
Using HTTP/1.1.
[WSAStart: Entering]
[WSAStart: bWSStarted = 0, nWSClients = 0]
[WSAStart: Attempting WinSock initialization.]
[WSAStart: WinSock initialized successfully.]
Windows Sockets Version 1.1 will be used.
Windows Sockets Version 2.2 is supported.
Windows Socket Description is 'WinSock 2.0'.
Windows Socket System Status is 'Running'.
Windows Socket supports up to 32767 socket(s).
Windows Socket supports UDP datagram of up to 65467 bytes.
Windows Socket has no additional vendor-specific information.
[WSAStart: SSL library load succeeded.]
Sockets Client ID 1 enabled.
[WSAStart: Exiting (rc = 1)]Server host is 'kp'.
Using port '80' for the connection.
Connecting to 'kp'.
Socket Opened.
Connected.
<GET / HTTP/1.1
Host: kp
User-Agent: HTTP-Adapter/7.5(79)
Content-Type: application/octet-stream
Content-Length: 0
X-Abandon-After: 2147483647
(0 bytes sent)
>HTTP/1.1 200 OK (S)
(8 lines received)
(2062 bytes received)
Server type is 'Microsoft-IIS/6.0'.
Run Completed at 09:33:08.654 on 04/14/04. Cleanup deferred.
Socket Closed.
Connection closed.
[WSATerm: Entering]
[WSATerm: bWSStarted = 1, nWSClients = 1]
Sockets Client ID 1 disabled.
[WSATerm: nWSClients remaining = 0]
[WSATerm: Attempting WinSock shutdown.]
[WSATerm: Exiting]
Run Completed at 09:33:13.482 on 04/14/04
```

Return codes and error messages

Return codes and messages are returned when the particular activity is completed. Return codes and messages might also be recorded as specified in the audit logs, trace files, execution summary files, etc.

Messages

The following lists all the codes and messages that can be returned when using the HTTP Adapter for sources or targets.

Note: Adapter return codes with positive numbers are warning codes that indicate a successful operation, while those with negative numbers are error codes that indicate a failed operation.

-??-> Sever specific error message.

-1-> doHTTP failed.

IBM MQ adapter overview

The IBM® MQ adapter works with IBM MQ to accomplish the task of maintaining data compatibility, handling all kinds of data transformations from the simplest to the most complex. The IBM MQ adapter can be used at either the source or target application, or at both. There is an adapter for IBM MQ client software and an adapter for IBM MQ server software.

IBM MQ software transports data between applications. For the receiving application to be able to successfully use data, the data must be transformed. IBM MQ can successfully move data in heterogeneous computer networks, allowing systems of dissimilar computers to move data to and from each other. Although data successfully arrives at a target, it often cannot be used in its native form and must be transformed into a format acceptable for the receiving application's use.

- [IBM MQ message content](#)
 - [Transmission queues](#)
 - [IBM MQ client or server adapter](#)
 - [System requirements](#)
 - [Command aliases](#)
 - [IBM MQ adapter commands](#)
 - [Syntax summary](#)
 - [Using the messaging adapter](#)
 - [Group message support](#)
 - [Example files](#)
 - [Troubleshooting](#)
 - [Return codes and error messages](#)
-

IBM MQ message content

The IBM® MQ message data content can be any content type. Under some conditions, message data contains a message header known as an IBM MQ Message Descriptor structure. For example, if you use the IBM MQ Link for R/3, the input message data always contains a header and a header must be created for an output message.

Use the IBM MQ adapter to include the message header in the data using the optional Header adapter command (-HDR). When the command is specified for an input message, the message header appears before the message content. If the command is specified for an output message, the adapter assumes that the first block of data supplied for the target consists of the appropriate size and values of a valid IBM MQ Message Descriptor structure. When the Header adapter command (-HDR) is specified, any of the message header components can be included in mapping rules.

For mapping purposes, definitions for the message descriptors and the IBM MQ Link for the R/3 header are provided in the **mq.mtt** type tree, located in the **examples\adapters\mq** folder.

Transmission queues

An IBM® MQ transmission queue is a local queue that stores messages destined for a remote queue. The messages are forwarded to their destination queue through a message channel when a communication program and link are available. The channel provides a one-way link to the remote queue manager. Messages are queued at the transmission queue until the channel can accept them. A message channel agent (channel program) that actually transmits the message must be associated with the transmission queue and the remote queue manager. When the message has been transmitted, it is deleted from the transmission queue. The channel definition must specify a transmission queue name at the transmitting end of the message channel.

The IBM MQ adapter supports the definition of transmission queues for data targets with the Transmission Queue Name adapter command (-XQN) and the Transmission Error Queue Name adapter command (-XEQN).

IBM MQ client or server adapter

IBM® MQ supports both client and server modes either from a single platform or distributed across multiple platforms. To select the correct client or server adapter, you must specify the adapter to use for a source or target in map cards, execution settings, or on the command line.

If you do not know the version of IBM MQ software you are running, contact your system administrator.

System requirements

The adapters for the IBM® MQ messaging system can be installed and operated in a Windows environment. The minimum system requirements and operating system requirements for this adapter are detailed in the release notes. The adapters must be installed on a machine running the IBM MQ client or server software.

This section identifies the requirements to install and run the IBM MQ adapter in an environment running IBM MQ client software or IBM MQ server software.

- [Client system](#)
 - [Server system](#)
-

Client system

IBM® MQ client system and configuration requirements include:

- IBM MQ client software is installed and configured. One or more queue managers and queues must have been defined on the system. For information about how to configure IBM MQ, see the IBM MQ documentation.
- The access methods for the IBM MQ adapter follow standard IBM MQ authentication guidelines and do not require any special permissions to work with IBM MQ. The adapter requires permission to those IBM MQ functions used in the IBM MQ adapter. Although registering the user account as a member of the IBM MQ Group (mqm) would simplify the configuration (because of not having to set individual user or group permissions), it is not a requirement. Permissions can be set at the user account level to connect to, get messages from, and put messages on queues. These minimal permissions are: Connect, Put, Get, Browse, and Inq. The Set permission is not required because the maps do not alter queue managers or queues.

Server system

IBM® MQ server system and configuration requirements include:

- IBM MQ server software is installed and configured. One or more queue managers and queues must have been defined on the system. For information about how to configure IBM MQ, see the IBM MQ documentation.
- The access methods for the IBM MQ adapter follow standard IBM MQ authentication guidelines and do not require any special permissions to work with IBM MQ. The adapter requires permission to those IBM MQ functions used. Although registering the user account as a member of the IBM MQ Group (mqm) would simplify the configuration (because of not having to set individual user or group permissions), it is not a requirement. Permissions can be set at the user account level to connect to, get messages from, and put messages on queues. These minimal permissions are: Connect, Put, Get, Browse, and Inq. The Set permission is not required because the maps do not alter queue managers or queues.

Command aliases

Adapter commands can be specified by using a command string on the command line or by creating a command file that contains adapter commands. The execution command syntax is:

```
-IM[ alias ] card_num  
-OM[ alias ] card_num
```

where **-IM** is the Input Source - Override execution command and **-OM** is the Output Target - Override execution command, **alias** is the adapter alias, and **card_num** is the number of the input or output card, respectively. The following table shows the adapter aliases and execution commands.

Adapter	Alias	As Input	As Output
IBM® MQ (client)	MQSC	-IMMQSCcard_num	-OMMQSCcard_num
IBM MQ (server)	MQS	-IMMQScard_num	-OMMQScard_num

When using an adapter alias with the execution command, the IBM MQ adapter commands can be issued on the command line or in a command file. Use the adapter commands to specify adapter functions such as:

- specifying a particular message identifier
- allowing output data to be broken up into multiple messages
- retrieving a logical message from a source queue with a correlation identifier

For example, to override the adapter commands defined in output card 1, the command string for the adapter in an IBM MQ server environment might be:

```
-IMMQS1 '-QM1 queuemgr -QN topqueue -T'
```

This sample execution command string specifies the queue manager named **queuemgr** and the queue named **topqueue** as the source for this card's data. The **-T** command dictates that a trace file be created to report adapter activity information, recording the events that occur while the adapter is retrieving this data.

- [GET > Source settings](#)
- [PUT > Target settings](#)

GET > Source settings

Use the Map Designer **GET** settings to configure options for input map cards that use the IBM® MQ (client) or IBM MQ (server) adapters. For the **Source** setting, select the data source as either **MQSeries® (client)** or **MQSeries (server)** to use the IBM MQ adapter.

- [GET > Source > Command setting](#)
- [Source > Transaction > OnSuccess setting](#)
- [Source > Transaction > OnFailure setting](#)
- [Source > Transaction > Scope setting](#)
- [Source > Transaction > Warnings setting](#)

GET > Source > Command setting

Use the **Command** setting to enter adapter commands.

Source > Transaction > OnSuccess setting

Use the **OnSuccess** setting with adapter sources to prevent deletion of the source message(s) from the server from which the messages originate. The default value is **Delete**.

Value

Description

Keep

Keep messages on the source queue.

Delete

Delete messages on the source queue depending upon the map completion status and the value of **OnFailure**.

For example, if the Source->Transaction->OnSuccess setting is **Delete** and the Source->Transaction->OnFailure setting is **Rollback** and the map fails, the message is rolled back and is left on the input queue. If the setting is **Rollback** and the map succeeds, the transaction is committed.

If **Rollback** is not set, there is no IBM® MQ transaction syncpoint control activation. The message remains deleted, regardless of the completion status of the map.

Source > Transaction > OnFailure setting

Use the **OnFailure** setting with adapter sources to select the rollback or commit behavior if the map does not successfully complete. The default value is Rollback.

Value

Description

Rollback

If the map does not complete successfully, sources retrieved through the messaging adapter are not deleted.

Commit

The adapter behaves as if the map was successful and the action specified by the **OnSuccess** setting is taken.

For example, if the Source->Transaction->OnSuccess setting is **Delete** and the Source->Transaction->OnFailure setting is **Rollback** and the map succeeds, the messages processed on the input queue are deleted. If the map fails, the messages are rolled back and left on the input queue.

However, if the Source->Transaction->OnSuccess setting is **Delete** and Source->Transaction->OnFailure setting is **Commit**, there is no IBM® MQ transaction syncpoint control activation. The message remains deleted, regardless of whether the map is completed. The point is that if the Source->Transaction->OnFailure setting is not **Rollback**, a message can be deleted even if a map fails.

Source> Transaction> Scope setting

Use the **Scope** setting to specify when to check for success (**OnSuccess**) or failure (**OnFailure**) at the completion of the processing of either a map, burst, or card. This is so that the rollback and retry actions can be performed as specified. The default value is Map.

Value

Description

Map

Check for success or failure at the completion of each map. If the map successfully completes, use the **OnSuccess** setting. If the map fails, use the **OnFailure** setting.

Burst

Check for success or failure at the completion of each burst. If the burst is successful, use the **OnSuccess** setting. If the burst fails, use the **OnFailure** setting.

Card

Check for success or failure at the completion of each card. If the card is successfully processed, use the **OnSuccess** setting. If the card fails, use the **OnFailure** setting.

The Burst and Card transaction scope settings cause subsequent MQ read-only GET operations within the same map to start from the beginning of the queue. This can be useful when you perform multiple consecutive ad-hoc searches (by using a RUN map or GET calls) for messages that are not in any specific order on the queue.

Source> Transaction> Warnings setting

Use the **Warnings** setting to determine whether to fail the map or ignore warning conditions for the specific input map card. The warning messages are written to the execution audit log. The default value is Ignore.

Value

Description

Ignore

Ignore warnings for this map card.

Fail

Fail the map upon receiving a warning for this map card.

Warning messages have a positive return code.

PUT > Target settings

Use the Map Designer **PUT** settings to configure options for output map cards that use the IBM® MQ (client) or IBM MQ (server) adapters. Use the **PUT->Target** setting to select the data target. Select **MQSeries** (client) or **MQSeries® (server)** to use the IBM MQ adapter.

- [PUT > Target > Command setting](#)
- [Target > Transaction > OnSuccess setting](#)
- [Target > Transaction > OnFailure setting](#)
- [Target > Transaction > Scope setting](#)
- [Target > Transaction > Warnings setting](#)
- [PUT > Target > Retry settings](#)

PUT > Target > Command setting

Use the `PUT > Target > Command` setting to enter adapter commands.

Target > Transaction > OnSuccess setting

Use the **OnSuccess** setting with adapter targets to prevent a message from being unnecessarily created during any processes. The default value is Create.

Value

Description

Create

Upon successful completion of the burst, map, card, or rule (determined by **Scope** settings), send the message.

CreateOnContent

Upon successful completion of the burst, map, card, or rule (determined by **Scope** settings), send the message only if it has content.

Target > Transaction > OnFailure setting

Use the **OnFailure** setting with adapter targets to select the rollback or commit behavior if the map does not complete successfully. The default value is Rollback.

Value

Description

Rollback

If the map does not complete successfully, the message is not sent.

Commit

The adapter behaves as if the map was successful and the action (**Create** or **CreateOnContent**) specified by the **OnSuccess** setting is taken.

Target > Transaction > Scope setting

Use the **Scope** setting to specify when to check for success (**OnSuccess**) or failure (**OnFailure**) of a card so that the actions specified with the **OnSuccess/OnFailure** settings can be performed. The default value is Map.

Value

Description

Map

Check for success or failure at the completion of each map. If the map completes successfully, use the **OnSuccess** setting. If the map fails, use the **OnFailure** setting.

Burst

Check for success or failure at the completion of each burst. If the burst is successful, use the **OnSuccess** setting. If the burst fails, use the **OnFailure** setting.

Card

Check for success or failure at the completion of each card. If the card is processed successfully, use the **OnSuccess** setting. If the card fails, use the **OnFailure** setting.

Target > Transaction > Warnings setting

Use the **Warnings** setting to determine whether to fail the map or ignore warning conditions for the specific output map card. The warning messages are written to the execution audit log. The default value is Ignore.

Value

Description

Ignore

Ignore warnings for this map card.

Fail

Fail the map upon receiving a warning for this map card.

Warnings have a positive return code.

PUT > Target > Retry settings

Use the **Retry** settings to specify the adapter response to a failure to establish a connection to a queue manager or queue. **Retry** settings include **Switch**, **MaxAttempts**, and **Interval**.

If adapter errors occur, such as failure to attach to a queue and retry is specified (**Switch** is set to ON), the adapter process is retried at the interval specified in the **Interval** setting, up to as many times as requested in the **MaxAttempts** setting.

- [Target > Retry > Switch setting](#)
- [Target > Retry > MaxAttempts setting](#)
- [Target > Retry > Interval setting](#)

Target > Retry > Switch setting

Use the **Switch** setting to determine whether to activate **Retry** for the specific adapter (ON) or not (OFF). The default value is OFF.

Value

Description

ON

This activates **Retry** for the specific adapter.

OFF

Retry is inactive.

Target > Retry > MaxAttempts setting

Use the **MaxAttempts** setting to specify the maximum number of times that **Retry** is attempted. The default value is 0.

Value

Description

0 - value

This is an integer representing the maximum number of times that **Retry** is attempted.

Target > Retry > Interval setting

Use the **Interval** setting to specify the number of seconds to wait before attempting **Retry** again. The default value is 0.

Value

Description

0 - value

This is an integer representing the number of seconds to wait before attempting **Retry** again.

IBM MQ adapter commands

The following table lists valid commands for the IBM® MQ adapter, the command syntax, and whether the command is supported (✓) for use with sources, targets, or both:

Name	Syntax	Source	Target
Channel Definition (-CD) (use with Client side only)	-CD <i>channel_name/transport_type/connection_name</i>	✓	✓
Cluster Queue Manager Name (-CQMN)	-CQMN [<i>cluster_manager_name</i>]		✓
Coded Character Set ID (-CCSID)	-CCSID <i>numeric_identifier</i>	✓	✓
Complete Message (-CMMSG)	-CMMSG	✓	
Convert (-CVT)	-CVT	✓	
Correlation ID (-CID)	-CID [<i>correlation_ID</i>]	✓	✓
Default Header (-DH)	-DH		✓
Error Queue Manager Name (-EQMN)	-EQMN <i>error_queue_mgr</i>	✓	✓
Error Queue Name (-EQN)	-EQN <i>error_queue_name</i>	✓	✓
Message expiration	-EXPIRY <i>time</i>		✓
Global Transaction Management (-GTX)	-GTX	✓	✓
Group Message (-GRP)	-GRP [<i>group_ID</i>]	✓	
Group Message 2 (-GRP2)	-GRP2	✓	
Header (-HDR)	-HDR	✓	✓
Hex Correlation ID (-HCID)	-HCID <i>hex_correlation_ID</i>	✓	✓
Hex Group Message ID (-HGRP)	-HGRP <i>hex_group_ID</i>	✓	
Hex Message ID (-HMID)	-HMID <i>hex_message_ID</i>	✓	✓
Ignore Group Message (-XGRP)	-XGRP	✓	
Listen (-LSN)	-LSN {0 S <i>wait_time_in_sec</i> }	✓	
Message Buffer Size (-MBS)	-MBS <i>bytes</i>	✓	
Message Format (-FORMAT)	-FORMAT <i>message_format</i>		✓
Message ID (-MID)	-MID [<i>message_ID</i>]	✓	✓
MQOPEN Options (-MQOO)	-MQOO <i>option_number</i>	✓	✓
Packet (-PKT)	-PKT <i>packet_size</i>		✓
Password (-P)	{-PASSWORD -P} <i>password</i>	✓	✓
Quantity (-QTY)	-QTY { <i>value</i> S}	✓	
Queue Manager Name (-QMN)	-QMN <i>queue_manager</i>	✓	✓
Queue Name (-QN)	-QN <i>queue_name</i>	✓	✓
Refresh Message Cursor (-REFRESH)	-REFRESH [<i>reset_time</i>]	✓	

Name	Syntax	Source	Target
Require All Messages (-ALLMSG)	-ALLMSG	✓	
Require All Segments (-ALLSEG)	-ALLSEG	✓	
Trace (-T)	-T[E S] [full_path]	✓	✓
Transmission Error Queue Name (-XEQN)	-XEQN x_err_queue_name	✓	✓
Transmission Queue Name (-XQN)	-XQN x_queue_name		✓
User (-U)	{-USERID -U} user_ID	✓	✓
Version 2 (-V2)	-V2	✓	✓
Wait Interval (-WI)	-WI {0 S wait_interval_in_milliseconds}	✓	

- [Channel definition \(-CD\)](#)

- [Cluster queue manager name \(-CQMN\)](#)

- [Coded character set identifier \(-CCSID\)](#)

Use the Coded Character Set ID (-CCSID) adapter command to identify or transform the character set encoding of message data.

- [Complete message \(-CMMSG\)](#)

- [Convert \(-CVT\)](#)

- [Correlation ID \(-CID\)](#)

- [Default header \(-DH\)](#)

- [Error queue manager name \(-EQMN\)](#)

- [Error queue name \(-EQN\)](#)

- [Message expiration \(-EXPIRY\)](#)

The **-EXPIRY** command specifies the amount of time that a message can spend on a queue before it becomes eligible for the queue manager to discard it.

- [Global transaction management \(-GTX\)](#)

- [Group message \(-GRP\)](#)

- [Group message 2 \(-GRP2\)](#)

- [Header \(-HDR\)](#)

- [Hex correlation ID \(-HCID\)](#)

- [Hex group message ID \(-HGRP\)](#)

- [Hex message ID \(-HMID\)](#)

- [Ignore group message \(-XGRP\)](#)

- [Listen \(-LSN\)](#)

- [Message buffer size \(-MBS\)](#)

- [Message format \(-FORMAT\)](#)

- [Message ID \(-MID\)](#)

- [MQOPEN options \(-MQOO\)](#)

- [Packet \(-PKT\)](#)

- [Password \(-P\)](#)

The **-PASSWORD** command specifies the password that the IBM MQ adapter uses to authenticate its connection to the IBM MQ queue manager.

- [Quantity \(-QTY\)](#)

- [Queue manager name \(-QMN\)](#)

- [Queue name \(-QN\)](#)

- [Refresh message cursor \(-REFRESH\)](#)

- [Require all messages \(-ALLMSG\)](#)

- [Require all segments \(-ALLSEG\)](#)

- [Trace \(-T\)](#)

- [Transmission Error queue name \(-XEQN\)](#)

- [Transmission queue name \(-XQN\)](#)

- [User \(-U\)](#)

The **-USERID** command specifies the user ID that the IBM MQ adapter uses to authenticate its connection to the IBM MQ queue manager.

- [Version 2 \(-V2\)](#)

- [Wait Interval \(-WI\)](#)

Related reference

- [Wildcard support](#)
-

Channel definition (-CD)

Use the optional Channel Definition adapter command (-CD) to specify the client-connection channel definition, which is used by the IBM® MQ Client adapter at runtime to establish the connection to the queue manager.

Note: The Channel Definition adapter command (-CD) can be used only with the IBM MQ Client adapter. (The connection will fail if the IBM MQ Server adapter is used.) If this command is used, the MQSERVER, MQCHLLIB, and MQCHLTAB environment variables are ignored by the adapter for connecting to the IBM MQ Queue Manager.

By defining this adapter command, it is possible to connect to multiple queue managers from the same map using the MQ Client adapter, because each connection will be specified using a different -CD adapter command.

-CD channel_name/transport_type/connection_name

Option

Description

channel_name

Specify the name of the server-connection channel that must be set up on the queue manager to which the IBM MQ Client adapter is connecting.

transport_type

Specify the name of the communication protocol used to establish the connection with the queue manager on the IBM MQ server. The following values are valid:

- LU62
- TCP
- NETBIOS
- SPX
- DECNET

It is the user's responsibility to correctly specify this value. The chosen transport protocol must be supported in the underlying IBM MQ environment.

connection_name

Specify the name of the connection. The format of this value depends on the communication protocol that was selected with *transport_type*. For example, for the TCP communication protocol, the connection name consists of the IP address and the port number on which the MQ listener on MQ server host is waiting for the client requests.

For example:

```
-CD CHANNEL1/TCP/127.0.0.1(1414)
```

This command specifies the following:

- **CHANNEL1** is the name of the server-connection channel.
- **TCP** is the name of the communication protocol used to establish the connection.
- **127.0.0.1(1414)** is the name of the connection. The IP address can be in IPv4 (for example, 1.2.3.4) or IPv6 (for example, a:b:c:d:0:1:2:3) format.

Cluster queue manager name (-CQMN)

Use the Cluster Queue Manager Name adapter command (-CQMN) to put messages to a queue accessible from a cluster queue manager.

```
-CQMN [cluster_manager_name]
```

where *cluster_manager_name* is the optional name of the cluster queue manager upon which the target queue exists. If *cluster_manager_name* is not specified, IBM® MQ selects the queue manager from the cluster.

To put a message to a specific instance of a cluster queue (that is, a queue instance that resides on a particular queue manager), specify the name of that queue manager as the argument for the Cluster Queue Manager Name adapter command (-CQMN). This forces the local queue manager to send the message to the specified destination queue manager.

The Cluster Queue Manager Name adapter command (-CQMN) must be accompanied by the Queue Name adapter command (-QN) and the local Queue Manager Name adapter command (-QMN).

If the argument to -QN is the name of a cluster queue and the -CQMN argument is not supplied, the actual destination of messages sent using the queue handle returned by the MQOPEN call is chosen by the queue manager (or by a cluster workload exit if one exists) according to the value of the DefBind queue attribute. This attribute may have one of two values:

- If MQOO_BIND_ON_OPEN is specified, the queue manager selects a particular instance of the cluster queue during the processing of the MQOPEN call, and all messages put using this queue handle are sent to that instance.
- If MQOO_BIND_NOT_FIXED is specified, the queue manager may choose a different instance of the destination queue (residing on a different queue manager in the cluster) on each successive MPUT call that uses this queue handle.

For more information about setting these options, see "[MQOPENoptions \(-MQOO\)](#)".

- [Examples](#)

Examples

To put a message to a queue named **q1** (which exists on the cluster queue manager **cqm1**) where the adapter is connecting to the cluster queue manager **cqm_default**, enter:

```
-QMN cqm_default -QN q1 -CQMN cqm1
```

To put a message to a queue named **q2** residing on a cluster queue manager, for which the local queue manager of the cluster is **cqm_default**, enter:

```
-QMN cqm_default -QN q2 -CQMN
```

The Cluster Queue Manager Name adapter command (-CQMN) applies only to data targets. IBM® MQ does not support gets (data sources) from remote queues not residing on the local queue manager, which include cluster queues.

Coded character set identifier (-CCSID)

Use the Coded Character Set ID (-CCSID) adapter command to identify or transform the character set encoding of message data.

For data targets, the *numeric_identifier* parameter identifies the character set encoding of the message data that is being put on the queue.

For data sources, the *numeric_identifier* parameter identifies the character set encoding to which the message data is to be transformed. To use this function, you must also specify the -CVT adapter command.

```
-CCSID numeric_identifier
```

Option

Description
<i>numeric_identifier</i> The decimal coded-character-set identifier (CCSID).

Complete message (-CMMSG)

Use the Complete Message adapter command (-CMMSG) for data sources to specify retrieval of complete logical messages within group messages.

-CMMSG

The Complete Message adapter command (-CMMSG) specifies that only a complete logical message can be returned by the MQGET call. If the logical message is segmented, the queue manager reassembles the segments and returns the complete logical message.

The Complete Message adapter command (-CMMSG) is the only adapter command that causes the queue manager to reassemble message segments. Using the Complete Message adapter command (-CMMSG) directs the messaging adapter to set the **MQGMO_COMPLETE_MSG** flag in the **MQGMO.Options** field.

If the queue contains segmented messages with some of the segments missing (perhaps the messages were delayed in the network and have not yet arrived), use the Complete Message adapter command (-CMMSG) to prevent retrieval of segments belonging to incomplete logical messages.

For information about incomplete message groups contributing to the size of **CurrentQDepth**, see "[Incomplete message segments](#)".

The Complete Message adapter command (-CMMSG) implies that all segments in a logical message must be available for retrieval. Therefore, it is not necessary to specify the Require All Segments adapter command (-ALLSEG).

Convert (-CVT)

Use the Convert adapter command (-CVT) for data sources to enable the IBM® MQ server queue manager to perform a data conversion based upon the client-server coded character set identifier (CCSID) settings, or to execute data conversion user exits. For example, you can use this command to convert from the EBCDIC character set to the ASCII character set to implement distributed messaging using IBM MQ.

-CVT

To satisfy all the requirements for IBM MQ when using this command, see your IBM MQ documentation.

Correlation ID (-CID)

Use the Correlation ID adapter command (-CID) to specify a particular correlation identifier (CID) for a source or target. Use this adapter command for a data source to retrieve messages from a queue with a specific correlation ID. You can also use it for a data target to assign a correlation ID to a message when placing it on a queue. A correlation identifier can be up to 24 bytes long and is expected to contain printable characters.

-CID [*correlation_ID*]

Spaces can also be specified in values. However, if the value contains a space, it must be enclosed with double quotation marks as shown in the example below:

-CID "my ID"

The double quotation mark character can also be used as part of a value. To do this, the double quotation mark value must be represented by a pair of double quotation marks. For example, if you wanted to specify a correlation ID of **My "New" ID**, you would need to enter the following:

-CID "My ""New"" ID"

When a correlation identifier is specified for a source, the adapter retrieves the first message on the queue with that correlation ID. If no argument is supplied, the correlation ID of the first message retrieved is used.

If the Correlation ID adapter command (-CID) is specified and is used with the Message ID adapter command (-MID), the retrieved messages begin with the first message on the queue that has the specified message ID and the correlation ID. Additional messages are retrieved that have the same message ID and correlation ID. If no correlation ID is specified, the message is identified only by the message ID.

When a correlation identifier is specified for a data target, the adapter places the message on the queue with the assigned correlation identifier.

Related reference

- [Wildcard support](#)

Default header (-DH)

The Default Header adapter command (-DH) specifies that the default IBM® MQ Message Descriptor (MQMD) is used as the message descriptor for data targets.

-DH

Any null rules (=NONE) in the mapped header field are ignored and the default MQMD values generated by the queue manager are used. Any non-null header fields are set in the MQMD to replace the default MQMD values. The only exception to this is fields that describe the origin and identity context of the message.

Fields describing the origin context of the message are the following (listed in the order in which they appear in the example type tree, **mq.mtt**):

- PutApplType
- PutApplName
- PutDate
- PutTime
- ApplOriginData

Fields describing the identity context of the message are the following (listed in the order in which they appear in the example type tree, **mq.mtt**):

- UserIdentifier
- AccountingToken
- ApplIdentityData

Do not try to specify values for these fields if you use the **-DH** option in the adapter command. Using that option causes values you provided in these fields to be ignored by the adapter and replaced with the default IBM MQ values. If you need to explicitly specify values for these fields, use the **-HDR** option in the adapter command (without the **-DH** option).

The Default Header adapter command (**-DH**) overrides the Header adapter command (**-HDR**).

The appropriate header information must be mapped if the Default Header adapter command (**-DH**) is used. If the header information is not mapped and the Default Header adapter command (**-DH**) is used, the mqseries trace shows that the map fails with the following error:

```
<1476-2248>: [m4mqspPut]
<1476-2248>: | [intm4mqsobtainQueueNode]
<1476-2248>: | | Queue node found: QNAME
<1476-2248>: | [intm4mqsobtainQueueNode] (rc = 0) OK
<1476-2248>: | Adapter error rc = -1011 [Invalid message format]
<1476-2248>: [m4mqspPut] (rc = -1011) *** ERROR ***
<1476-2248>: [m4mqspEndTransaction]
<1476-2248>: | Rolling back unit of work
<1476-2248>: [m4mqspEndTransaction] (rc = 0) OK
```

Error queue manager name (-EQMN)

Use the Error Queue Manager Name adapter command (**-EQMN**) for data sources and data targets to specify the name of the queue manager for the error queue.

-EQMN error_queue_manager_name

Only use the Error Queue Manager Name adapter command (**-EQMN**) when the Error Queue Name adapter command (**-EQN**) is specified and you want to designate a different queue manager for the error queue than the default. The default queue manager is specified with the Queue Manager Name adapter command (**-QMN**).

Only use this adapter command when using the IBM® MQ Client adapter; using it with the IBM MQ Server adapter causes an error. This is because only the IBM MQ Client application can be connected to more than one queue manager at the same time.

Error queue name (-EQN)

Use the Error Queue Name adapter command (**-EQN**) for data sources and data targets to specify the error queue to which messages are copied when errors occur. If a message has the rollback option set and a map does not successfully complete, the message is copied to the designated error queue. The original message is removed from the source queue.

-EQN error_queue_name

The name of the error queue must be specified. By default, the queue manager used is the one specified with the Queue Manager Name adapter command (**-QMN**).

Message expiration (-EXPIRY)

The **-EXPIRY** command specifies the amount of time that a message can spend on a queue before it becomes eligible for the queue manager to discard it.

-EXPIRY time

Option

Description

time

Expiry time in tenths (0.1) of a second. The default value (-1) specifies no message expiration.

Global transaction management (-GTX)

Use the Global Transaction Management adapter command (**-GTX**) to indicate that the transactions for this card (whether input or output) should be processed as global transactions.

-GTX

For more information about global transaction management, see the Global Transaction Management documentation.

Group message (-GRP)

Use the Group Message adapter command (-GRP) for data sources to specify retrieval of messages from a single group from the source.

-GRP [group_ID]

Option	Description
group_ID	This is the GroupId field of the MQ Message Descriptor (MQMD)

For example, to specify a message group with the GroupId of **GroupA**, the syntax would be:

-GRP GroupA

If the Quantity adapter command (-QTY) is not specified, only one message belonging to the group will be retrieved. If *group_ID* is not specified, the first message group encountered is considered the active message group and is retrieved. You can explicitly specify the number of messages to retrieve using the Quantity adapter command (-QTY). The group name has a maximum size of 24 bytes and may contain spaces. If the value contains a space, it must be enclosed with double quotation marks as shown in the example below:

-GRP "Group A"

The double quotation mark character can also be used as part of a value. To do this, the double quotation mark value must be represented by a pair of double quotation marks. For example, if you wanted to specify a message group with the GroupId of **Group "A"**, you would need to enter the following:

-GRP "Group ""A"""

If the Group Message adapter command (-GRP) is used with an event source queue, the adapter will initially browse the queue and look for a message that belongs to the group whose *group_ID* was provided. If *group_ID* was not provided, the first message that belongs to any group will be retrieved and the adapter will keep note of the ID of that group. From that point on, only messages that belong to the same group to which the first message belonged will be retrieved.

Related reference

- [Wildcard support](#)

Group message 2 (-GRP2)

Use the Group Message 2 adapter command (-GRP2) for data sources to specify retrieval of message groups and non-group messages from the source queue.

-GRP2

The -GRP2 adapter command enables better control over group messages than the -GRP adapter command. The biggest difference between the -GRP2 and -GRP adapter commands is that -GRP2 can be used to process multiple message groups on the source queue. In addition to this, message groups are processed as a unit with -GRP2 and not as individual messages.

While the -GRP adapter command detects a group on the queue and then starts retrieving messages that belong only to that group, the -GRP2 adapter command detects a group on the queue, retrieves all the messages from that group in a single block of data, and keeps looking for other message groups on the queue and other individual non-group messages.

The following restrictions apply to the -GRP2 adapter command:

- It has no arguments.
- It can only be used on input (input cards and GET map function).
- It cannot be combined with the -GRP adapter command.
- It assumes usage of the -XGRP, -ALLSEG, and -ALLMSG adapter commands. Whether explicitly specified or not, these three adapter commands will be automatically included by the -GRP2 adapter command.
- It can be used with the -EQMN and -EQN adapter commands. In case of an error, all messages from the retrieved groups as well as all retrieved non-group messages are moved to the specified error queue.
- in the Launcher scenario, it can be combined with the -REFRESH adapter command. This is particularly useful when there is a possibility that an incomplete message group gets skipped by the queue cursor in order to obtain another complete message group (or non-group message), and the skipped group later becomes complete.

The following table lists some examples of the -GRP2 adapter command usage.

Presume that the following content of queue **QUEUE1** is defined under queue manager **QMNAME**:

Message ID	Message in a group?	Group ID	Logical Sequence Number	Last message in the group?	Segmented message?	Offset	Last Segment?	Data
1	NO							PhysMess01
2	YES	1	1	NO	NO			PhysMess02
3	YES	1	2	YES	NO			PhysMess03
4	NO							PhysMess04
5	YES	1	3	NO	NO			PhysMess05
6	YES	2	1	NO	NO			PhysMess06
7	NO							PhysMess07
8	NO							PhysMess08
9	YES	2	2	NO	YES	0	NO	PhysMess09

Message ID	Message in a group?	Group ID	Logical Sequence Number	Last message in the group?	Segmented message?	Offset	Last Segment?	Data
10	YES	2	2	NO	YES	10	NO	PhysMess10
11	YES	2	2	NO	YES	20	YES	PhysMess11
12	YES	2	3	YES	NO			PhysMess12
13	YES	3	1	YES	NO			PhysMess13
14	NO							PhysMess14

There are few important things to note in this table:

- Physical messages on positions 1, 4, 7, 8 and 14 are logical messages that do not belong to any group.
- Physical messages on positions 2, 3 and 5 are logical messages that belong to a group with the group ID equal to "1".
- Physical messages on positions 6, 9, 10, 11 and 12 belong to a group with the group ID equal to "2". Physical messages on positions 6 and 12 are logical messages, and physical messages on positions 9, 10 and 11 are message segments that together form a single logical message. Physical message on position 11 is the last segment of the logical message. Physical message on position 12 is the last logical message of the group.
- Physical message on position 13 is a logical message and it belongs to a group with the group ID equal to "3". This message is the only logical message in the group.

Presume that a map has one input card and one output card. The input card is of IBM® MQ type, and the output card is of "File" type. The data content obtained in the input card is stored to a file specified in the output card.

- [Example 1](#)
- [Example 2](#)

Example 1

For example, enter the following command:

```
-QMN QMNAME1 -QN QUEUE1 -GRP2 -QTY 2
```

The data returned will be:

```
PhysMess01PhysMess02PhysMess03PhysMess05
```

Explanation:

-QTY 2 required two message units to be retrieved from the queue. The first message on the queue is PhysMess01. This message does not belong to a group and is returned as a single message. It contributes to one unit in the -QTY 2 value.

The next message on the queue is PhysMess02. This message belongs to a complete group with the group id "1". The adapter retrieves the whole group (messages PhysMess02, PhysMess03 and PhysMess05) and returns them as a chunk of data that contributes to the second unit in the -QTY 2 adapter command.

If the -GRP2 adapter command was omitted, the returned data would be:

```
PhysMess01PhysMess02
```

Example 2

For example, enter the following command:

```
-QMN QMNAME1 -QN QUEUE1 -GRP2 -QTY 4
```

The data returned will be:

```
PhysMess01PhysMess02PhysMess03PhysMess05PhysMess04  
PhysMess06PhysMess09PhysMess10PhysMess11PhysMess12
```

Explanation:

-QTY 4 required four message units to be retrieved from the queue. The adapter first retrieved PhysMess01, and then PhysMess02, PhysMess03 and PhysMess05, as in the previous example. After that, there are two more message units that need to be retrieved.

The third message unit is message PhysMess04. It is a non-group message, and the adapter retrieves it after retrieving the message group "1". Finally, the forth unit is message group "2" that consists of the following physical messages:

```
PhysMess06, PhysMess09, PhysMess10, PhysMess11 and PhysMess12
```

If -GRP2 adapter command was omitted, the returned data would be:

```
PhysMess01PhysMess02PhysMess03PhysMess04
```

Header (-HDR)

Use the Header adapter command (-HDR) to specify the inclusion of the IBM® MQ message descriptor in the source and target data.

-HDR

When Header adapter command (-HDR) is used for a data source, the message descriptor appears first, before the message content. Use the Header adapter command (-HDR) to include any of the message descriptor components in mapping rules.

When the Header adapter command (-HDR) is used for a data target, the message descriptor must be the first component of the output type. Mapping rules for the message descriptor components must be provided.

Included in the **examples** folder is a type tree that can be used when specifying this command. For more information, see ["Example files"](#).

For data targets only, if the Header adapter command (-HDR) is used in conjunction with the Message ID adapter command (-MID), the Header command (-HDR) takes precedence over the Message ID command, which is ignored. If the Header adapter command (-HDR) is used in conjunction with the Version 2 adapter command (-V2), then the version information mapped in the header is ignored.

Hex correlation ID (-HCID)

Use the Hex Correlation ID adapter command (-HCID) to specify correlation identifiers using hexadecimal pair notation. This passes to the adapter the exact correlation ID to locate, even if that correlation ID contains non-printable values, spaces, or even null characters.

-HCID hex_correlation_ID

Option

Description
<i>hex_correlation_ID</i>
This is the hexadecimal correlation identifier. For example, the ASCII correlation ID My Msg is represented as 4D79204D7357 when specified with the Hex Correlation ID adapter command (-HCID). The space character must be encoded in the hex pair notation.

The hexadecimal correlation ID wildcard notation is not encoded in the hex pair notation. The wildcard character * is entered literally. For example, the hexadecimal pair notation for x*z is 78*7A, not 782A7A. The Hex Correlation ID (-HCID) cannot be used with the Correlation ID adapter command (-CID).

Related reference

- [Wildcard support](#)

Hex group message ID (-HGRP)

Use the Hex Group Message ID adapter command (-HGRP) for data sources to specify group message identifiers using hexadecimal pair notation. This passes to the adapter the exact group message ID to locate, even if that group message ID contains non-printable values, spaces, or even null characters.

-HGRP hex_group_ID

Option

Description
<i>hex_group_ID</i>
This is the hexadecimal group message identifier. For example, the ASCII group message ID My Msg is represented as 4D79204D7357 when specified with the Hex Group Message ID adapter command (-HGRP). The space character must be encoded in the hex pair notation.

The hexadecimal group message ID wildcard notation is not encoded in the hex pair notation. The wildcard character * is entered literally. For example, the hexadecimal pair notation for x*z is 78*7A, not 782A7A. The Hex Group Message ID (-HGRP) cannot be used with the Group Message adapter command (-GRP).

Related reference

- [Wildcard support](#)

Hex message ID (-HMID)

Use the Hex Message ID adapter command (-HMID) to specify message identifiers using hexadecimal pair notation. This passes to the adapter the exact message ID to locate, even if that message ID contains non-printable values, spaces, or even null characters.

-HMID hex_message_ID

Option

Description
<i>hex_message_ID</i>
This is the hexadecimal message identifier. For example, the ASCII message ID My Msg is represented as 4D79204D7357 when specified with the Hex Message ID adapter command (-HMID). The space character must be encoded in the hex pair notation.

The hexadecimal message ID wildcard notation is not encoded in the hex pair notation. The wildcard character * is entered literally. The hexadecimal pair notation for x*z is 78*7A, not 782A7A.

The Hex Message ID (-HMID) cannot be used with the Message ID adapter command (-MID).

Related reference

- [Wildcard support](#)

Ignore group message (-XGRP)

Use the Ignore Group Message adapter command (-XGRP) for data sources to ignore the group messages on the queue and not to treat them as individual, physical messages. The default behavior of the IBM® MQ adapter is to process all physical messages, including group messages, as individual events.

-XGRP

The Ignore Group Message adapter command (-XGRP) causes the adapter to ignore any group messages on that queue for the source event card. If both groups and non-group messages are to be processed as events from the same source queue, the Group Message 2 (-GRP2) command can be used as described on page [Group message 2 \(-GRP2\)](#).

If the Group Message adapter command (-GRP) is used with an event source queue, the adapter ignores non-group messages on that queue. It only processes the messages from the group specified with the -GRP adapter command, or, if no group was specified, the first group that it locates on the queue.

If neither -XGRP nor -GRP is used, all messages are treated as non-group messages and processed individually.

Listen (-LSN)

Use the Listen adapter command (-LSN) for data sources to dictate the length of time (in seconds) that the IBM® MQ queue manager allows the adapter to wait for a message to be received. Map execution is suspended until the message is received or until the specified time lapses.

-LSN {0|S|wait_time_in_sec}

Option

Description

0

Do not wait at all. If there is no input available, the adapter does not wait for more messages.

S

This is an infinite wait time.

wait_time_in_sec

This is the number of seconds that the adapter waits (in seconds) for a message.

If the Listen adapter command (-LSN) is not issued and the message is not available for the adapter to retrieve immediately, the map completes, but returns the following warning in the audit log:

(2033) MQSeries Warning: No message available.

Message buffer size (-MBS)

Use the Message Buffer Size adapter command (-MBS) to specify the size of the message buffer. The adapter allocates this message buffer size for retrieving messages from a source queue. If the buffer size is insufficient, the buffer is automatically reallocated to accommodate the messages.

-MBS bytes

Option

Description

bytes

This is the size of the message buffer in bytes. The default buffer size is 4096 bytes.

Message format (-FORMAT)

Use the Message Format adapter command (-FORMAT) for data targets to specify the format of the application data in the message. Either pre-existing (defined by IBM® MQ) or custom (user-defined) message formats can be used.

-FORMAT message_format

Option

Description

message_format

A message format, either pre-existing (predefined by IBM MQ) or custom (user-defined). The format can contain up to eight (8) characters. If less than eight, it will be padded with spaces.

For example, to specify MQFMT_STRING as the format of the message, the following adapter command should be used:

-FORMAT MQSTR

where MQSTR is the value of the MQFMT_STRING message format. For information about other built-in message formats, see your IBM MQ documentation.

You must always specify a value when using the -FORMAT adapter command. Failure to do so will result in map failure and a corresponding error message is generated.

If you use a user-defined format, you must also provide the corresponding data-conversion exit so that receiving applications can read the message using the MQGMO_CONVERT option. For IBM Transformation Extender applications, this means using the -CVT adapter command for the IBM MQ inputs.

If the message format is not provided (either with the usage of the -FORMAT adapter command or through the message descriptor type tree with the -HDR/-DH adapter command), the MQFMT_STRING message format is used as a default value.

To specify a format value that has spaces interleaved with text (or all spaces), use double quotes.

For example:

```
-FORMAT "ab cd"
```

or

to specify a blank format (all spaces), you can insert eight (8) spaces in double quotes:

```
-FORMAT "     "
```

Note that the value of the -FORMAT adapter command is the value of the predefined format constant, not the name of the constant.

For example, use:

```
-FORMAT MQSTR
```

or:

```
-FORMAT "MQSTR "
```

and not:

```
-FORMAT MQFMT_STRING
```

MQFMT_STRING is the name of the predefined IBM MQ message format constant, and its value is "MQSTR "

Message ID (-MID)

Use the Message ID adapter command (-MID) to specify a particular message ID for a data source or target.

```
-MID [message_ID]
```

Option

Option	Description
message_ID	A message ID can be up to 24 bytes long and may contain spaces.

If the value contains a space, it must be enclosed with double quotation marks as shown in the example below:

```
-MID "my ID"
```

The double quotation mark character can also be used as part of a value. To do this, the double quotation mark value must be represented by a pair of double quotation marks. For example, if you wanted to specify a message ID of My "New" ID, you would need to enter the following:

```
-MID "My ""New"" ID"
```

When using the Launcher, the message ID can be specified with wildcard notation. For more information about wildcard notation, see the Launcher documentation.

When a message ID is specified for a data source, the adapter retrieves the first message with the designated message ID. If no argument is supplied, the message ID of the first message retrieved will be used.

When a message ID is specified for a target, the adapter places the message on the queue with that message ID assigned. If this command is not specified for a target, the message is put on the queue using a unique message identifier generated by the queue manager.

For targets only, if the Header adapter command (-HDR) is used in conjunction with the Message ID adapter command (-MID), the Header adapter command (-HDR) takes precedence over the Message ID adapter command (-MID), which is ignored.

Related reference

- [Wildcard support](#)

MQOPEN options (-MQOO)

Use the MQOPEN Options adapter command (-MQOO) to explicitly set the numeric value of the Options parameter in the MQOPEN API call made by the adapter to the MQSeries® library when opening a queue.

```
-MQOO option_number
```

Option

Option	Description
option_number	Valid values are any positive integer that represents the value of the option parameter in the MQOPEN API call made when opening a queue.

As an example of the required option usage, if you want to specify an open option for the queue to be used on the output side with an MQOO_FAIL_IF_QUIESCING property, you must specify more than the MQOO_FAIL_IF_QUIESCING value (8192). You must also add the MQOO_OUTPUT option value (16), which indicates that the queue will be opened for output (storing messages). The correct syntax for this example would be:

```
-MQOO 8208
```

because MQOO_OUTPUT + MQOO_FAIL_IF_QUIESCING = 16+8192 = 8208.

The valid value is important; otherwise, the MQOPEN, MQGET, or MQPUT calls made by the adapter may fail. For the options and their corresponding values that can be used with the -MQOO adapter command, see the MQ Open Options section in the *IBM® MQ Application Programming Reference*.

The MQOPEN Options adapter command (-MQOO) must be accompanied by the Queue Name adapter command (-QN) and the Queue Manager Name adapter command (-QMN).

For example, to specify the target queue named **target_q** under the queue manager **qmgr** for message outputs (where 16 is the numeric value of the MQOO_OUTPUT option), the syntax would be:

```
-QMN qmgr -QN target_q -MQOO 16
```

To specify the source queue named **source_q** under the queue manager **qmgr**, for message input in browse mode with shared access (where 10 is the numeric superposition of the MQOO_INPUT_SHARED (2) and MQOO_BROWSE (8) options), the syntax would be:

```
-QMN qmgr -QN source_q -MQOO 10
```

Packet (-PKT)

Use the Packet adapter command (-PKT) for data targets to break output data into multiple messages and to put these messages on the target queue with the same message identifier.

```
-PKT packet_size
```

Option

Description

packet_size

The packet size must be specified in units of 1000 bytes with a minimum of one unit.

For example, to specify a packet size of 3000 bytes, the syntax would be:

```
-PKT 3
```

If one of the following conditions is met:

- If the -HDR or -DH command is used on output, and the message header has the **Version** field set to a value of one (1).
- If the Version 2 (-v2) command is used (which enforces the use of the Version 1 header from MQSeries® 2.x). For more information about this command, see ["Version 2 \(-V2\)"](#).

The created physical messages will be independent logical messages that will have the same message id. These messages cannot be automatically reassembled on input, however, this can be achieved through the proper map design.

On the other hand, if BOTH of the following conditions are met:

- If -HDR or -DH is used on output and if the message header has **Version** field set to a value of two (2).
- If the Version 2 (-v2) command is omitted (which would enforce the use of Version 1 header from MQSeries 2.x). For more information about this command, see ["Version 2 \(-V2\)"](#).

The created physical messages will be message segments of the original message. The Offset property of the message will provide information about the relative position of each of the segments in the original message. In this case, -CMMSG adapter command can be used on input to automatically reassemble the original message.

The Packet adapter command (-PKT) is ignored for a message source.

Password (-P)

The **-PASSWORD** command specifies the password that the IBM® MQ adapter uses to authenticate its connection to the IBM MQ queue manager.

```
{-PASSWORD | -P} password
```

Option

Description

password

The password to authenticate.

You must issue both the **-USER** command and the **-PASSWORD** command to initiate authentication and to store the user ID and password information in the IBM MQ MQCSP structure.

Related information

- [User \(-U\)](#)

Quantity (-QTY)

Use the Quantity adapter command (-QTY) to specify the number of messages to retrieve from the source queue, without regard to message IDs. If the Quantity adapter command (-QTY) is not specified, the default value is 1.

```
-QTY {value|S}
```

Option	Description
S	This returns all messages on the queue.
value	This is a positive integer representing the number of concurrent multiple messages to be retrieved.
For example, to specify a quantity of ten messages, the syntax would be:	
-QTY 10	
The Quantity adapter command (-QTY) defines the maximum number of messages across all bursts for a map.	
The number of bursts in a single map execution can be controlled using the combination of values set in the Map Designer Source > FetchAs > FetchUnit setting and the Quantity adapter command (-QTY).	
When using the Quantity adapter command (-QTY) with the S option through either the Launcher or the Command Server, the Listen adapter command (-LSN) is also required with a value specified other than S.	

Queue manager name (-QMN)

Use the Queue Manager Name adapter command (-QMN) to specify the name of the queue manager on which the queue (specified by the -QN command) exists.

-QMN queue_manager_name

Option	Description
queue_manager_name	This is the name of the queue manager. This name is case-sensitive.
When the Queue Manager Name adapter command (-QMN) is not specified, the default queue manager is used.	

Queue name (-QN)

Use the Queue Name adapter command (-QN) to specify the name of the queue from which the message(s) are to be retrieved or to which the message(s) are to be sent.

-QN queue_name

Option	Description
queue_name	This is the name of the queue. This name is case sensitive.
The Queue Name adapter command (-QN) is a required adapter command.	

Refresh message cursor (-REFRESH)

Use the Refresh Message Cursor adapter command (-REFRESH) to ensure that higher-priority messages are enqueued in front of any existing messages with a lower priority so that the higher-priority messages can trigger the associated maps to run. This command is only used when using an IBM® MQ event input with the Launcher. If messages of a higher priority are enqueued after messages of a lower priority, the default behavior is that the higher priority messages would not trigger the maps.

-REFRESH [reset_time]

Option	Description
reset_time	This is a non-negative integer defining the number of seconds that elapse before the message cursor is reset to the head of the queue.
The Refresh Message Cursor adapter command (-REFRESH) resets the message cursor of the source queue message event listener to the head of the queue after every listener cycle (detection of all messages currently on the queue) is completed. You can optionally specify a non-negative integer argument (reset_time) that defines the number of seconds that elapse before the messages cursor is reset to the head of the queue. If reset_time is set to zero, the message cursor will be constantly reset to the beginning of the queue after each event. Thus, the higher priority messages are processed as soon as possible. However, frequent repositioning of the cursor will degrade overall performance.	

The Refresh Message Cursor adapter command (-REFRESH) is only needed when using an IBM MQ event input with the Launcher, and only in two specific situations:

- When messages of higher priority are placed at the beginning of the queue that are not recognized because the message cursor has already begun moving towards the bottom of the queue.
- When there is a possibility of the message cursor skipping the uncommitted messages on the queue. As an example, suppose that two applications, APP1 and APP2, simultaneously place messages on an empty queue and the following sequence of operations occurs (a possibility because APP1 and APP2 are independent and are executing concurrently):
 - TO: APP1 PUT(M1)
 - T1: APP2 PUT(M2)
 - T2: APP2 COMMIT(M2)
 - T3: APP1 COMMIT(M1)

After T1 occurs, both messages have already been placed on the queue (with M1 in front of M2). However, they are not being recognized by the message cursor.

After T2 occurs, the message cursor recognizes message M2 and it positions itself on that message. After T3 occurs, message M1 is also committed, but it is now beyond the message cursor. The message cursor will never be able to recognize that message unless it is repositioned to the beginning of the queue. Use the Refresh Message Cursor adapter command (-REFRESH) to accomplish this repositioning.

Using the Refresh Message Cursor adapter command (-REFRESH) may degrade event notification and mapping performance and should only be used when necessary. The adapter must be careful not to trigger the map more than once for the same input message. To ensure this, the adapter maintains a list of messages that have already been seen by the cursor. Each time the cursor is repositioned to the beginning of the queue, each message that the cursor finds on the queue is compared with all of the messages in this list. Only in situations in which this message is not in the adapter list is that message considered to be a new message to be used for the event notification. As part of this process, this new message will be added to the adapter list. To ensure that this list does not grow infinitely, there are two important limitations for the input cards that use the -REFRESH adapter command:

- The **Rollback** option for the **OnFailure** setting should not be used.
- The **Keep** option for the **OnSuccess** setting should also not be used.

If either of these two options is specified, running the Launcher for a long period of time with many messages being processed may cause excessive memory usage and may slow down processing time significantly, and eventually may cause the system to stop.

The reason for this is that messages once seen by the cursor are added to the internal list, and as long as they are not removed from the source queue, they are also not removed from the internal adapter's list so that they are not processed more than once. Messages are freed from the internal list only after they have been removed from the source queue.

In situations in which it is necessary to use the Refresh Message Cursor adapter command (-REFRESH) and to preserve the messages for the failed map, it is recommended that you use Error Queue Name adapter command (-EQN) in combination with -REFRESH adapter command for the input card. In this case, the Source->Transaction->OnSuccess setting should be **Delete** and the Source->Transaction->OnFailure setting should be **Rollback**. The messages for the map failure will be placed on the error queue and removed from the source queue. Thus, messages will be preserved (on the error queue), and no memory leak will occur because it will be possible to remove messages from the internal adapter's list upon moving them to the error queue. Note, however, that if, for some reason, messages cannot be stored on the error queue (for example, PUT is disabled for the error queue), the messages will remain on the input queue and in the internal adapter's list which again provides the possibility of a memory leak.

Require all messages (-ALLMSG)

Use the Require All Messages adapter command (-ALLMSG) for data sources to specify retrieval of messages in a group only when all messages in that group are available.

The use of the -ALLMSG adapter command automatically sets the requirement that group messages are returned in their logical order and that segments of logical messages are returned in the order of their offsets.

This must be considered when specifying additional selection criteria (such as the -MID or -CID adapter commands). It may be possible that the message which satisfies this additional criteria is not the message with the smallest sequence number and/or the smallest offset in its group, which will prevent it from being retrieved from the queue.

-ALLMSG

If the source queue contains an incomplete message group, the Require All Messages adapter command (-ALLMSG) prevents retrieval of any messages belonging to the incomplete groups.

For information about incomplete message groups contributing to the size of **CurrentQDepth**, see "[Incomplete message segments](#)".

The Require All Messages adapter command (-ALLMSG) implies that all segments in a logical message must be available for retrieval. Therefore, it is not necessary to specify the Require All Segments adapter command (-ALLSEG) in conjunction with -ALLMSG.

Require all segments (-ALLSEG)

Use the Require All Segments adapter command (-ALLSEG) for data sources to specify retrieval of segments in a logical message group only when all segments in the logical message are available.

The use of the -ALLSEG adapter command automatically sets the requirement that group messages are returned in their logical order and that segments of logical messages are returned in the order of their offsets.

This must be considered when specifying additional selection criteria (such as the -MID or -CID adapter commands). It may be possible that the message which satisfies this additional criteria is not the message with the smallest sequence number and/or the smallest offset in its group, which will prevent it from being retrieved from the queue.

-ALLSEG

If the source queue contains segmented messages with some missing segments, the Require All Segments adapter command (-ALLSEG) prevents retrieval of the message segments belonging to incomplete logical messages.

For information about incomplete message groups contributing to the size of **CurrentQDepth**, see "[Incomplete message segments](#)".

Trace (-T)

Use the Trace adapter command (-T) to produce a trace file that contains information about connections to IBM® MQ, message sizes, IBM MQ error messages, and adapter error messages.

The default filename is **mqstrace.log** and it is created in the map directory unless otherwise specified.

For z/OS® Batch and CICS® environments, **mqstrace.log** does not conform to the naming convention. For z/OS Batch and CICS environments, the trace output is written to the dataset indicated by the ddname **MQTRACE** that is (in most cases) defined in the JCL as : //MQTRACE DD SYSOUT=*

If the trace file does not already exist, it is created. If it already exists it is automatically appended.

-T[E|S] [full_path]

Option

Description
E
Error mode. Produces a trace file containing only the adapter errors that occurred during map execution.
S
Summary mode. Use this option to reduce the size of the trace file. The summary trace file excludes polling information and function entries and exits-reporting only minimal information, such as the activity that occurs within a function (when applicable).
<i>full_path</i>
Create a trace file with the specified name in the specified directory. By default, the directory is where the map is located and the file name is mqstrace.log .

You can override the adapter command line trace options dynamically using the Management Console. For detailed information, see "Dynamic Adapter Tracing" in the *Launcher* documentation .

Transmission Error queue name (-XEQN)

Use the Transmission Error Queue Name adapter command (-XEQN) for data sources and targets to specify the name of the transmission queue for the remote error queue to which messages are copied when errors occur. For more information about using the IBM® MQ adapter with IBM MQ transmission queues, see "[Transmission Queues](#)".

-XEQN x_err_queue_name

Option

Description
<i>x_err_queue_name</i>

This is the name of the transmission queue. The transmission queue name is case-sensitive.

For example, to specify that the transmission queue xmit.q should be used as the transmission queue for a remote error queue whose remote queue definition is named error.q, the syntax would be:

-QMN** qmlocal -QN qlocal -EQN error.q -XEQN xmit.q**

In this syntax, **qmlocal** is the name of the queue manager and **qlocal** is the name of the queue that is used as data source or data target in the map.

If a local transmission queue specified with the Transmission Error Queue Name adapter command (-XEQN) exists, it is used. If it does not exist, the queue specified by the IBM MQ DefaultXmitQ attribute (DEFXMITQ in MQSC) on the local queue manager is used.

Transmission queue name (-XQN)

Use the Transmission Queue Name adapter command (-XQN) for data targets to specify the name of the transmission queue that holds messages destined for a remote queue. For more information about using the IBM® MQ adapter with IBM MQ transmission queues, see "[Transmission queues](#)".

-XQN x_queue_name

Option

Description
<i>x_queue_name</i>

This is the name of the transmission queue. The transmission queue name is case sensitive.

For example, to specify that the local transmission queue **xmit.q** be used as the data target for a remote queue named **dest.q** and the local queue manager of **qmlocal**, the syntax would be:

-QMN** qmlocal -QN dest.q -XQN xmit.q**

For this command to work as intended, the local transmission queue must have the same name as the remote queue manager. If the local object **xmit.q** is not defined, the queue manager default transmission queue will be used.

If a local transmission queue exists with the same name as the transmission queue specified with the Transmission Queue Name adapter command (-XQN), the local transmission queue is used. If the transmission queue specified with the Transmission Queue Name adapter command (-XQN) does not exist, the queue specified by the MQSeries® DefaultXmitQ attribute (DEFXMITQ in MQSC) on the local queue manager is used.

User (-U)

The **-USERID** command specifies the user ID that the IBM® MQ adapter uses to authenticate its connection to the IBM MQ queue manager.

{-USERID | -U} user_ID

Option

Description
<i>user_ID</i>

The user ID to authenticate.

You must issue both the **-USERID** command and the **-PASSWORD** command to initiate authentication and to store the user ID and password information in the IBM MQ MQCSP structure.

Related information

- [Password \(-P\)](#)

Version 2 (-V2)

Use the Version 2 adapter command (-V2) to set all message descriptor and option structure version fields to _VERSION_1, which corresponds to IBM® MQ version 2.x.

-V2

If the Version 2 adapter command (-V2) is not specified, the default is _VERSION_2, which corresponds to IBM MQ version 5.x.

The Version 2 adapter command (-V2) is used with sources and targets.

If the Header adapter command (-HDR) is used in conjunction with the Version 2 adapter command (-V2), then the version information mapped in the header is ignored.

Wait Interval (-WI)

Use the Wait Interval adapter command (-WI) to set the length of time (in milliseconds) that the IBM® MQ queue manager allows the adapter to wait for a message to be received. Map execution is suspended until the message is received or until the specified time lapses.

-WI {0|S|wait_interval_in_milliseconds}

Option

Description

0

Do not wait at all. If there is no input available, the adapter does not wait for more messages.

S

This is an infinite wait time.

wait_interval_in_milliseconds

This is the number of milliseconds that the adapter waits for a message.

If the Wait Interval adapter command (-WI) is not issued, the default time that the adapter waits for a message is once for each second.

If the Wait Interval adapter command (-WI) is issued, the Launcher cannot be paused or shut down for the length of time you specified in the *wait_interval_in_milliseconds* command option. Do not specify the S command option and attempt to pause or shut down the Launcher. This will cause the Launcher to stop responding.

Syntax summary

- [Data sources](#)
- [Data targets](#)
- [Wildcard support](#)

Data sources

Syntax of the IBM® MQ adapter commands used for data sources:

```
-QN queue_name
[-ALLMSG]
[-ALLSEG]
[-CD channel_name/transport_type/connection_name]
[-CCSID numeric_identifier]
[-CID [correlation_ID] | -HCID hex_correlation_ID]
[-CMMSG]
[-CVT]
[[-EQMN error_queue_manager_name] [-EQN error_queue_name]]
[[-GRP [group_ID]] [-HGRP hex_group_ID] [-XGRP]]
[-GTX]
[-HDR]
[-LSN {0|S|wait_time_in_sec}]
[-MBS]
[-MID [message_ID] | -HMID hex_message_ID]
[-MQOO option_number]
[(-PASSWORD | -P) password]
[-QMN queue_manager_name]
[-QTY value]
[-REFRESH [reset_time]] [-T[E|S] [full_path]]
[(-USER | -U) user_ID]
[-V2]
[-WI {0|S|wait_interval_in_milliseconds}]
[-XEQN x_err_queue_name]
```

Data targets

Syntax of the IBM® MQ adapter commands used for data targets:

```
-QN queue_name [-CD channel_name/transport_type/connection_name]
[-CQMN [cluster_manager_name]]
[-CCSID numeric_identifier]
[-EXPIRY time]
[-GTX]
[-HDR |-DH
    | [-MID [message_ID] |-HMID hex_message_ID]
    | [-CID [correlation_ID] |-HCID hex_correlation_ID]
    | [-FORMAT message_format]
]
[-MQOO option_number]
[(-PASSWORD | -P) password]
[-PKT packet_size]
[-QMN queue_manager_name]
[-TE(S) [full_path]]
[(-USER | -U) user_ID]
[-V2]
[-XEQN x_err_queue_name]
[-XQN x_queue_name]
```

Wildcard support

Wildcards are supported for the following adapter commands:

- Message ID (-MID)
- Hex Message ID (-HMID)
- Correlation ID (-CID)
- Hex Correlation ID (-HCID)
- Group Message (-GRP)
- Hex Group Message ID (-HGRP)

For more information about using wildcards, see the Launcher documentation.

Related reference

- [Message ID \(-MID\)](#)
- [Hex message ID \(-HMID\)](#)
- [Correlation ID \(-CID\)](#)
- [Hex correlation ID \(-HCID\)](#)
- [Group message \(-GRP\)](#)
- [Hex group message ID \(-HGRP\)](#)

Using the messaging adapter

When you use the IBM® MQ adapter as a source, by default messages are served from the queue in a first-in-first-out (FIFO) manner unless you specify a particular message by using a message identifier, correlation identifier, or group identifier. When you use the adapter as a target, messages are placed on the queue that you specify.

When you use the IBM MQ adapter in a Launcher environment, the user account that starts the Launcher service must be registered in the MQM group.

If more than one thread or process updates the watched queue with new messages, specify the **-REFRESH** and **-EQN** adapter commands on the command line of an input card. The **-REFRESH** command enables the IBM MQ adapter listener to process all messages on the queue, particularly when the messages are inserted and committed at different time intervals by multiple threads. The **-EQN** adapter command identifies a queue to which messages are copied when an error occurs.

By default, the IBM MQ adapter listener enables cooperative browsing. Cooperative browsing enables multiple IBM MQ adapter watches to monitor the same queue. When you deploy additional IBM MQ adapter listeners to watch the same queue from different Launchers, message handling is automatically coordinated and load-balanced with fail-over support.

Cooperative browsing requires no additional command line settings. To disable cooperative browsing, set the WTX_ADAPTER_MQS_DISABLE_COOP environment variable to any value before you start the Launcher.

Cooperative browsing is designed to process physical messages. The IBM MQ adapter listener automatically disables cooperative browsing when the command line of the watched input card specifies any of these commands:

- **-ALLMSG**
- **-ALLSEG**
- **-CMMSG**
- **-GRP2**
- [Threading for IBM MQ](#)
- [Synchronizing multiple watches on the same queue](#)

You can synchronize multiple watches on the same queue to help ensure that only one MQ Adapter attempts to process a message. If multiple adapters do attempt to process the same message, only one adapter processes the message. The other adapters cannot access the message and report a failure. Synchronize the time frame for processing pending maps with the mark interval of the queue to prevent multiple adapters from attempting to process the same message.

Threading for IBM MQ

The IBM® MQ adapter is thread-safe on the Windows, AIX®, Sun, and HP platforms. However, due to the IBM MQ default system configuration parameters for queues and queue managers, the default IBM MQ resources might be insufficient to support multithreading with the IBM MQ adapter.

When operating in the multithreaded mode, the Launcher executes one thread for each map being concurrently executed. In addition to the map thread, the IBM MQ queue manager connection for each distinct queue manager on a source or target card requires a separate connection thread. Because IBM MQ does not allow connection sharing between threads, each map thread maintains its own set of connection threads.

In addition to the map and connection threads, the IBM MQ system spawns threads to service internal requests and processes. The result is that, as the number of concurrent maps increases, the use of IBM MQ and system resources multiplies. When the system resources exceed a threshold based upon the IBM MQ and system configuration settings, the excessive thread context-switching and system memory resource allocation can significantly reduce the efficiency of the overall IBM MQ message throughput.

In addition to the reduction in efficiency, it is possible that when the IBM MQ systems resources are exceeded, the IBM MQ behavior can become unpredictable and can result in messages being ignored on event queues.

To address these limitations, the following recommendations should be implemented:

- Limit the number of concurrent messaging adapter threads to 20. This is a configuration parameter (/runtime/launcher/MaxThreads) set in the **config.yaml** file.
- Ensure that the maximum number of handles (queue and queue manager connections) supported by the queue manager exceeds the total number of messaging adapter and IBM MQ threads. This value can be changed by using an MQSC command at the **runmqsc** prompt as:

```
alter qmgr MAXHANDS(2048)
```

- Set the maximum number of uncommitted messages allowed by the queue manager to an acceptably large value:

```
alter qmgr MAXUMSGS(100000)
```

- Set the maximum number of messages allowed on a queue to an acceptable value:

```
alter ql(my.queue.name) MAXDEPTH(200000)
```

The limit on non-z/OS® Batch and CICS® systems is 640K. The queue manager can enforce smaller actual limits based upon DASD resource limitations.

Synchronizing multiple watches on the same queue

You can synchronize multiple watches on the same queue to help ensure that only one MQ Adapter attempts to process a message. If multiple adapters do attempt to process the same message, only one adapter processes the message. The other adapters cannot access the message and report a failure. Synchronize the time frame for processing pending maps with the mark interval of the queue to prevent multiple adapters from attempting to process the same message.

1. On the input card of the watched queue, set the Pending Instance Thresholds - High Launcher setting to a value that limits the number of messages that can be processed within a particular time frame.
2. Set the mark interval of the watched queue to a time (in milliseconds) that is, at a minimum, long enough for the Launcher to process the maximum number of pending messages that you set in step 1. Run the following command within the IBM® MQ **runmqsc** application to set the mark interval of the watched queue:
ALTER QMGR MARKINT [time in milliseconds | NOLIMIT]

Group message support

The adapter for IBM® MQ supports retrieval of IBM MQ group messages as data source objects. The group message feature allows multiple messages to be placed on a message queue while retaining an organizational connection through the **GroupId** field of the MQ Message Descriptor (MQMD).

- A message group consists multiple logical messages.
- A logical message consists of one or more physical messages.
- When a logical message contains more than one physical message, the logical message is segmented. Each physical message is a segment.
- Each logical message has the same **GroupId** and a distinct **MsgSeqNumber**.
- Each physical message has distinct MQ Message Descriptor (MQMD).
- Each segment has the same **GroupId** **MsgSeqNumber** and a distinct **Offset**.
- [Physical message](#)
- [Logical message](#)
- [Incomplete message segments](#)

Physical message

A physical message is the smallest unit of information that can be placed on or removed from a queue. A physical message often corresponds to the information specified in or retrieved from a single MQPUT or MQGET call. Each physical message has a unique message descriptor (MQMD).

Generally, physical messages are distinguished by differing values for the message identifier (**MsgId** field in the MQMD) although this is not enforced by the queue manager.

Logical message

A logical message is a single unit of application information. In the absence of system constraints, a logical message is the same as a physical message. Logical messages can be extremely large; system constraints might make it advisable or necessary to split a logical message into two or more physical messages known as segments.

A logical message that has been segmented consists of two or more physical messages that have the same non-null group identifier (**GroupId** field in MQMD) and the same message sequence number (**MsgSeqNumber** field in MQMD). The segments are distinguished by differing values for the segment offset (**Offset** field in MQMD), which gives the offset of the data in the physical message from the start of the data in the logical message.

- [Message group](#)

Message group

A message group is a set of one or more logical messages that have the same non-null group identifier. The logical messages in the message group are distinguishable by differing values for the message sequence number. The message sequence number is an integer in the range 1 to n , where n is the number of logical messages in the group. If one or more of the logical messages is segmented, the group will contain more than n physical messages in the group.

Incomplete message segments

If the queue contains segmented messages with some of the segments missing (perhaps the messages were delayed in the network and have not yet arrived), specifying the Complete Message adapter command (-CMMSG) prevents the retrieval of segments belonging to incomplete logical messages.

However, those message segments belonging to incomplete logical messages still contribute to the value of the **CurrentQDepth** queue attribute. This segmentation could result in no retrievable logical messages, even when **CurrentQDepth** is greater than zero.

- [Reassembly of segments](#)
- [Retrieval of group messages](#)

Reassembly of segments

Each physical message that is a segment has a unique message descriptor. For segments that comprise a single logical message, most of the fields in the message descriptor are the same for all segments in the logical message. Usually, it is only the **MsgId**, **Offset**, and **MsgFlags** fields that differ between segments in the logical message. However, when segments take different paths through the network, and some of those paths have MCA sender conversion enabled, it is possible for the **CodedCharSetId** and **Encoding** fields to differ between segments when the segments eventually arrive at the target queue.

A logical message consisting of segments in which the **CodedCharSetId** and/or **Encoding** fields differ cannot be reassembled by the queue manager into a single logical message. Instead, the queue manager reassembles and returns the first few consecutive segments at the start of the logical message having the same character-set identifiers and encodings, and the MQGET call completes with completion code **MQCC_WARNING** and reason code **MQRC_INCONSISTENT_CCSIDS** or **MQRC_INCONSISTENT_ENCODINGS**, as appropriate.

When the queue manager reassembles a logical message, the message descriptor is returned with the values from the message descriptor for the first segment; the only exception is the **MsgFlags** field, which the queue manager sets to indicate that the reassembled message is the only segment.

Retrieval of group messages

The IBM® MQ adapter selects group messages for retrieval based upon three possible selection criteria:

- group ID (GroupID)
- message ID (MsgId)
- correlation ID (CorrelId)

When the Group Message adapter command (-GRP) is specified, the GroupId is used as the sole selection criteria for group message retrieval. If the Message ID adapter command (-MID) or the Correlation ID adapter command (-CID) is specified, the respective selection criteria are also used.

Be sure to organize message segments and groups on output when you specify the **GroupId**, **MsgSequenceNumber**, **Offset**, and **MsgFlags** fields in the **MessageDescriptor** type. Improper use of these fields could result in incomplete message groups and illogical messages on the output queue.

Example files

The IBM® MQ type tree (named **mq.mtt**) is provided in the **examples\adapters\mq** folder. Types describing different IBM MQ message formats are provided so that you can easily select the message types needed for input or output or the descriptor that you need to include with any message data you want.

When mapping information to or from the message descriptor, you must include the message descriptor in your data by using the Header adapter command (-HDR).

- [Using the sample type tree](#)

Using the sample type tree

Depending on how you want to map, select one of the **Message** types as input or output or add a message content type as a **MessageData** component.

The **mq.mtt** type tree is arranged so that you can easily integrate the provided definitions with any message data that you want to define.

The **mq.mtt** type tree has definitions for:

- **MQSeries® Header MessageDescriptor**

The MessageDescriptor Header type defines the components and format of the message descriptor.

- **MQSeries Header MessageDescriptor2**

The MessageDescriptor2 Header type defines the components and format of the message descriptor for IBM® MQ 5.x.

- **MQSeries Header MQSLink4R3**

The MQSLink4R3 Header type defines the components and format of the IBM MQ Link for R/3 Header. When the IBM MQ Link for R/3 is used for source data, the input message data always contains a header; however, you must create the header for an output message. The IBM MQ Link for R/3 Header is not optional. It always appears in your input data and it must be included when mapping output data.

The **mq.mtt** type tree is organized so that you can drag the **MQSeries** root type to another type tree to easily use any of the IBM MQ-specific headers.

- [Mapping message data only](#)
- [Mapping the message descriptor](#)
- [Mapping the IBM MQ Link for R/3 header](#)

Mapping message data only

By default, the adapter does not use message descriptors. On input, it returns only the body of the received message, and on output, it treats the data that it receives as the message body.

Even if you do not need to use message descriptors, you can still use this same example tree to define the content of the message.

To use the type tree for mapping message data only:

1. Define the message content for your message (shown below as **Message** group under **CreditInfo** category).
2. Add it as a component of **MessageData**. By default, **MessageData** contains **Text** item that represents plain textual message content. The **Text** item should be replaced with your own message content type, such as **Message** group under **CreditInfo** category in this example.
3. Use **MessageData** as an input or output card type.

Mapping the message descriptor

If the Header (-HDR) or Default Header (-DH) adapter command is specified, the message descriptor appears in the data, immediately preceding the message data when used for a source.

For a target, you must provide the message descriptor (all components (for -HDR), or components of interest [for -DH]), which is included as the first part of your output data.

Mapping the IBM MQ Link for R/3 header

If you use IBM® MQ to send and receive data from an SAP R/3 system, you might want to use the IBM MQ Link for R/3. This link communicates with R/3 using inbound and outbound server queues.

When using the IBM MQ Link for R/3, message data is in IDoc format, prefixed by the IBM MQ Link for R/3 Header.

The **mq.mtt** type tree includes the definition of this header. You also need the IDoc Importer to automatically generate the types that describe the format of your IDoc data.

Troubleshooting

For information about error codes and messages returned by the adapters, see ["Return codes and error messages"](#).

- [Trace log](#)

Trace log

Use the Trace adapter command (-T) to create a trace file to report adapter activity information, recording events that occur while the adapter is retrieving and sending data. The trace command produces a log file with the default name **mqstrace.log** in the map execution directory.

Return codes and error messages

Return codes and messages are returned when the particular activity completes. Return codes and messages may also be recorded as specified in the audit logs, trace files, execution summary files, and so on.

- [Messages](#)
- [IBM MQ and error code relationships](#)

Messages

The following is a listing of all the codes and messages that can be returned as a result of using the IBM® MQ adapter for sources or targets.

In addition to these codes, the IBM MQ adapter can return IBM MQ error codes. Consult the IBM MQ documentation for descriptions of these codes.

Adapter return codes with positive numbers are warning codes that indicate a successful operation. Adapter return codes with negative numbers are error codes that indicate a failed operation.

Return Code	Message
-1001	Trace file could not be accessed
-1002	Adapter command is incorrect
-1003	Queue could not be opened
-1004	Message could not be stored on the queue
-1005	Message could not be retrieved from the queue
-1006	Connection could not be closed
-1007	Global transaction could not be started
-1008	Transaction could not be rolled back explicitly
-1009	Transaction could not be committed
-1010	Queue could not be closed
-1011	Invalid message format
-1012	Transaction monitor not supported
-1013	Error queue operation could not complete
-1014	Unable to Begin Commitment Control (HP NonStop ZLE)
-1015	Unable to Backout (HP NonStop ZLE)
-1016	Unable to Commit (HP NonStop ZLE)
-1017	Segmentation not supported (HP NonStop ZLE)

See [IBM MQ and error code relationships](#) for more information.

IBM MQ and error code relationships

The IBM® MQ adapter returns various error codes when it encounters a problem during code execution. Some of these error codes are related to the errors returned by the MQ calls (such as MQCONN, MQGET) and some of them are not.

The following table lists all MQ calls made by the adapter, and how errors from these calls are mapped to error codes:

IBM MQ Native Function

Action Taken by Adapter

MQCONN

If MQCONN fails while trying to connect to the queue manager specified in the -QMNN adapter command, the error code returned by the adapter is MPIRC_E_BAD_CONNECTION (-13). However, if MQCONN fails while trying to connect to the error queue manager specified by the -EQMNN adapter command, the adapter returns error code MPIRC_E_MQS_EQUEUE_OPERATION (-1013) to indicate that the error occurred while transferring messages to the error queue.

MQOPEN

If MQOPEN fails while trying to open the queue specified in the -QN adapter command, the error code returned by the adapter is MPIRC_E_MQS_OPEN_QUEUE (-1003). However, if MQOPEN fails while trying to open the error queue specified by the -EQMNN adapter command, the error code returned by the adapter is

`MPIRC_E_MQS_EQUEUE_OPERATION` (-1013) to indicate that the error occurred while transferring messages to the error queue. Additionally, if the `MQOPEN` call issued for opening the queue manager object fails (this is done as part of the connection validation process), the returned error code is `MPIRC_E_BAD_CONNECTION` (-13) to indicate to the Resource Manager that the connection is invalid.

`MQGET`

If `MQGET` call fails and this error cannot be corrected or handled by the adapter, the adapter returns error code `MPIRC_E_MQS_GET_MESSAGE` (-1005).

Additionally, if `MQGET` call fails inside of the listener thread, and the MQ reason code is `MQRC_CONNECTION_BROKEN` (2009L), the adapter returns error code `MPIRC_E_BAD_CONNECTION` (-13) to signal that the connection is broken and needs to be reestablished.

`MQPUT`

If `MQPUT` call fails and this error cannot be corrected or handled by the adapter, the adapter returns error code `MPIRC_E_MQS_PUT_MESSAGE` (-1004).

`MQCMIT`

If `MQCMIT` call fails while committing the messages upon successful map completion, the adapter returns error code `MPIRC_E_MQS_COMMIT` (-1009).

`MQBACK`

The adapter does not return an error code as a result of `MQBACK` failure. This is because if the `MQBACK` is called, that means that the map has already failed for another reason.

`MQBEGIN`

This function is called only when the global transactions are used (-GTX adapter command). If `MQBEGIN` fails, the adapter returns `MPIRC_E_MQS_BEGIN_GTX` (-1007). On NonStop ZLE, if the `BEGINTRANSACTION()` call fails, the adapter returns error code `MPIRC_E_MQS_BEGINUOW_ERR` (-1014).

`MQCLOSE`

A failure in `MQCLOSE` does not cause the map to fail (but a textual message indicating this will be traced). The only time a failure in `MQCLOSE` results in an error code returned by the adapter, is when the queue manager connection is being validated. If `MQCLOSE` fails for the previously opened queue manager object for validation, the adapter returns error code `MPIRC_E_BAD_CONNECTION` (-13).

`MQDISC`

If `MQDISC` fails, the returned error code is `MPIRC_E_MQS_DISCONNECT` (-1006).

`MQINQ`

If `MQINQ` fails, the adapter does not return any error.

`MQCONN`

If `MQCONN` fails while trying to connect to the remote queue manager specified in the (-QMN) adapter command, the error code returned by the adapter is `MPIRC_E_BAD_CONNECTION` (-13). Note that the `MQCONN` command is used only by the IBM MQ (client) adapter, and only if the (-CD) adapter command is specified.

`MQPUT1, MQSET`

Not used by the adapter.

Informix Adapter overview

Use the Informix Adapter to access and manipulate data contained in databases that are Informix® data sources. You can also install this adapter on additional systems for remote database connectivity.

You can use the Informix Adapter with the Command Server, Launcher, Software Development Kit, or with a map in a map rule.

- [System requirements](#)
- [Database columns and types](#)
- [Database Interface Designer settings](#)
- [Informix Adapter commands](#)
- [Adapter commands for a source](#)
- [Adapter commands for a target](#)
- [Binding values in DBLOOKUP/DBQUERY](#)
- [Configuring DSN for Launcher startup](#)
- [Restrictions and limitations](#)
- [Return codes and error messages](#)

System requirements

It is assumed that a Command Server has already been installed on the computer where the Informix® adapters are to be installed for runtime purposes. In addition, the following requirements exist for installing and running the Informix adapters on a supported Windows platform:

- Informix ODBC 3.30 32-bit driver or later. Verify that you have installed the required drivers and configured your data source(s). Refer to the documentation included with your driver(s) for more information.

Database columns and types

The Database Interface Designer and **mtsmaker** with the Type Tree Maker generate type trees for queries, tables, views, and stored procedures in an Informix-compliant RDBMS (Relational Database Management System). Item types will be created in a type tree that represent the data types of the columns of a query, table, view, or stored procedure.

The Database Interface Designer and **mtsmaker** obtain information about columns by calling the Informix® Driver Manager that calls the Informix driver for your RDBMS to describe the columns associated with a query, table, view, or stored procedure. The Informix driver for the RDBMS has mapped the data types in each column to Informix data types as appropriate. From the RDBMS, the Informix Driver Manager returns the data type, length, and other information to the Database Interface Designer and **mtsmaker**. The Informix data types are then mapped to types in a type tree by the Database Interface Designer and **mtsmaker**.

- [Item type properties](#)
- [Date and time formats](#)

Item type properties

The following table lists the Informix® data types and the values of the item type properties to which they correspond when the type tree is generated.

Informix Data Type	Interpret as	Item Subclass, Presentation	Length
BYTE	Text	Binary	*
CHAR(n) ¹	Text	Character	*
CHARACTER(n) ¹	Text	Character	*
CHARACTER VARYING(m,r) ²	Text	Character	*
DATE	Text	Character	10
DATETIME	Text	Character	19
DEC(p,s) ³	Number	Character, Decimal	*
DECIMAL(p,s) ³	Number	Character, Decimal	*
DOUBLE PRECISION	Number	Binary, Float	8
FLOAT	Number	Binary, Float	8
INT	Number	Character, Integer	11
INTEGER	Number	Character, Integer	11
INTERVAL	Text	Character	*
MONEY(p,s) ³	Number	Character, Decimal	*
NCHAR(n) ¹	Text	Character	*
NUMERIC(p,s) ³	Number	Character, Decimal	*
NVARCHAR(m,r) ²	Text	Character	*
REAL	Number	Binary, Float	4
SERIAL	Number	Character, Integer	11
SMALLFLOAT	Number	Binary, Float	4
SMALLINT	Number	Character, Integer	6
TEXT	Text	Binary	*
VARCHAR(m,r) ²	Text	Character	*

*The DBMS dictates the length of this type.

¹n=1 £ n £ 32, 67

²m=maximum size of the column: 1-255, r=minimum reserved space: 0-255; r £ m

³p=precision, s=scale

Date and time formats

When using columns defined as dates and times in Informix® databases, specify the input and output formats for date, time, and timestamp listed below.

Date

`yyyy-mm-dd`

Time

`hh:mi:ss`

Timestamp

`yyyy-mm-dd hh:mi:ss[.fff...]`

Formats

Format	Description
cc	two-digit century
yy	two-digit year
mm	two-digit month
dd	two-digit day
hh	two-digit hour
mi	two-digit minute

ss	two-digit second
.fff...	optional fractional seconds
fffff	six-digit fraction of a second
hh24	two-digit hour using a 24-hour day

If the group format is fixed, the field is padded to 26 with trailing spaces.

Example:

2001-08-27 00:00:00 specifies August 27, 2001.

In the Generate Type Tree from dialog, use the Represent date/time columns as text items check box to define whether to automatically format this information as Date & Time (which is with this check box disabled, the result of which is shown in the Item Subclass, Presentation column) as a text string. When generated as a text string, it might be necessary to use either the TEXTTODATE or TEXTTOTIME function in a map rule to convert the text string to the date and time format required by the database. If you are generating new type trees, it is recommended that you disable this check box.

This Represent date/time columns as text items check box is modal. After it has been disabled, it will remain disabled for all subsequent type tree generations, regardless of source. Therefore, you must be careful in determining this setting.

If you have a Date & Time type with a presentation format of month-to-hour, you should enable the **Represent date/time columns as text** items check box so that these values can be treated as text items. Otherwise, these types will be interpreted erroneously using an hour-to-second presentation format.

Note: Not all variations of Informix interval and date/time columns are supported.

Database Interface Designer settings

When you define an Informix® database in the Database Interface Designer, in addition to the common settings available for all of the database adapters in the Database Definition dialog box, enter the information specific to Informix.

- [Database Definition dialog box](#)
- [Stored procedures native call syntax](#)

Database Definition dialog box

The Informix Adapter-specific settings are described in the following table:

Setting	Description
Data Source	<p>Database Interface Designer The data source you defined in your Informix® development PC that is used by the Database Interface Designer to access the database information for design-time purposes.</p> <p>Runtime The data source you defined in your Informix database host platform to be used for access to the database for run-time (map execution) purposes both from the Map Designer and either a Command Server or an Launcher.</p>
Security	<p>User ID The user ID to connect to the database.</p> <p>Password The authorization password to connect to the database.</p>

Stored procedures native call syntax

A stored procedure can be accessed from within a map by specifying the native call syntax in the following:

- A query that is specified in an input card.
- The first argument in a DBQUERY or DBLOOKUP function.
This argument does not need to be a literal. The arguments to the stored procedure may be determined at map execution time.
- The SQL Statement adapter command (-STMT) in DBQUERY, DBLOOKUP, or GET functions.

For information about using DBLOOKUP or DBQUERY or about using the syntax for device-independent calls to access return values and output adapter commands see the Database Interface Designer documentation.

For example, a call to a stored procedure using Informix® in a DBLOOKUP might be:

```
DBLOOKUP("{call MyAddNameProc('" + Name:Column + "','-1')}",
          "mydb.mdq",
          "MyDB")
```

String literals must be contained within single quotation marks; adapter arguments must be separated by commas.

Note: There could be issues with running a stored procedure using data values that are bound at map runtime (for example, input or output card stored procedure calls). As an alternative, run these same stored procedures using DBLOOKUP or DBQUERY.

Informix Adapter commands

Use the adapter commands when specifying data sources and targets. The applicability of many of the commands depends upon whether you are specifying a source or target, whether a database/query file (.mdq) is used, and the situations in which the usage of the command applies.

Adapter commands can be used in GET->Source->Command or PUT->Target->Command settings in the Map Designer and Integration Flow Designer, using GET, PUT, DBLOOKUP, or DBQUERY function calls, or overriding a data source or target using execution commands in a RUN function or on the command line.

If connectivity to the mainframe-based DB2® is available, you can:

- View DB2 host database table names directly from Database Interface Designer
- Build type trees for DB2 host tables without using MTSMAKER
- Test maps on your workstation
- [Adapter-specific command list](#)
- [Database adapter type \(-DBTYPE\)](#)
- [Row count \(-ROWCNT\)](#)
- [Data source \(-SOURCE\)](#)

Adapter-specific command list

This documentation describes those adapter commands that are Informix-specific database parameters. For a complete listing of all adapter commands, see the Resource Adapters documentation.

Database adapter type (-DBTYPE)

Use the Database Adapter Type adapter command (-DBTYPE) to specify the database adapter type.

-DBTYPE IFMX

Option	Description
IFMX	The database adapter type is Informix®. Note: This command must be specified if the original card is not a database and no database is specified using the Database/Query adapter command (-MDQ) and the Database Name adapter command (-DBNAME).

Row count (-ROWCNT)

Use the Row Count adapter command (-ROWCNT) to specify the number of rows to be retrieved per fetch.

-ROWCNT row_count

Option

Description

row_count

The number of rows to be retrieved per fetch. The time to fetch is optimized in accordance with the increase in the number of rows to retrieve. Therefore, a significant performance improvement may be gained by retrieving as many rows as possible in a single fetch.

The factor limiting the number of rows that can be retrieved is the memory requirement because the return buffer must be large enough to hold the rows retrieved.

The default for the number of rows to be retrieved per fetch for the Informix Adapter is one row.

Data source (-SOURCE)

Use the Data Source adapter command (-SOURCE) to specify the Informix® data source.

-SOURCE datasource

Option	Description
datasource	Specify the Informix data source.

Adapter commands for a source

This summary shows the syntax of the adapter commands that can be used when defining a data source using an .mdq file or without using one, including both the required and optional adapter commands in the following situations:

- Using the GET > Source > Command setting in the Map Designer and Integration Flow Designer.
- Overriding a data source using the Input Source Override - Database execution command (-ID) using a RUN function or on the command line.
- Using a DBLOOKUP, DBQUERY, or GET function in map or component rules.
- [GET > Source > Command setting](#)
- [Database execution command: input source override \(-ID\)](#)
- [DBLOOKUP or DBQUERY functions](#)
- [GET function](#)

GET > Source > Command setting

In an input card, when you specify Database as the value for the GET > Source setting, the Command field is displayed where you can enter adapter commands.

Database command options when there is a database/query file

```
[-DBTYPE IFMX]
[-SOURCE datalink]
[-STMT SQL_statement]
[-FILE [directory]]
[-VAR name=value...]
[-USER user_ID]
[-PASSWORD password]
[-CSTMT [number]]
[-ROWCNT row_count]
[-AUDIT[G] [+]] [full_path]
[{-TRACE|-TRACEERR} [+]] [full_path]]
```

Database command options when there is no database/query file

```
-DBTYPE IFMX
-SOURCE datalink
-STMT SQL_statement
[-FILE [directory]]
[-VAR name=value...]
[-USER user_ID]
[-PASSWORD password]
[-CSTMT [number]]
[-ROWCNT row_count]
[-AUDIT[G] [+]] [full_path]
[{-TRACE|-TRACEERR} [+]] [full_path]]
```

Database execution command: input source override (-ID)

Use the Input Source Override - Database execution command (-ID) to designate a database as the source or you can override one or more of the adapter command settings or database definitions in a RUN function or on the command line.

The adapter commands are shown using a single quotation mark, which is the Windows syntax. For non-Windows platforms, use two single quotation marks followed by one single quotation mark and end with one single quotation mark followed by two single quotation marks.

Scenario: compiled map source is a database

Database/query file (.mdq):

```
'[-MDQ mdq_file -DBNAME database_name]
[-QUERY query_name] [-STMT SQL_stmt]
[-FILE [directory]]
[-VAR name=value...]
[-USER username]
[-PASSWORD password]
[-CCARD|-CSTMT [number]]
[-ROWCNT row_count]
[-AUDIT[G] [+]] [full_path]
[{-TRACE|-TRACEERR} [+]] [full_path]]'
```

No database/query file (.mdq):

```
'-DBTYPE IFMX
-SOURCE datalink
-STMT SQL_statement
[-FILE [directory]]
[-VAR name=value...]
[-USER user_ID]
[-PASSWORD password]
[-CCARD|-CSTMT [number]]
[-ROWCNT row_count]'
```

Scenario: compiled map source is not a database

Database/query file (.mdq):

```
'-MDQ mdq_file
-DBNAME database_name
-QUERY query_name|-STMT SQL_stmt
[-FILE [directory]]
[-VAR name=value...]
[-USER username]
[-PASSWORD password]
[-CCARD|-CSTMT [number]]
[-ROWCNT row_count]
[-AUDIT[G] [+| [full_path]]
[{-TRACE|-TRACEERR} [+| [full_path]]]
```

No database/query file (.mdq):

```
'-DBTYPE IFMX
-SOURCE datalink
-STMT SQL_statement
[-SOURCE datasource]
[-FILE [directory]]
[-USER user_ID]
[-PASSWORD password]
[-CCARD|-CSTMT [number]]
[-ROWCNT row_count]
[-AUDIT[G] [+| [full_path]]
[{-TRACE|-TRACEERR} [+| [full_path]]]
```

DBLOOKUP or DBQUERY functions

The DBLOOKUP and DBQUERY functions can be used in component rules in the schema designer and map rules in the map designer when creating a map that can be used with a database.

Using DBLOOKUP with a database/query file

```
DBLOOKUP ("SQL_statement",
"-MDQ mdq_file
-DBNAME database_name
[-USER username]
[-PASSWORD password]
[-CCARD|-CSTMT [number]]
[-ROWCNT row_count]
[-AUDIT[G] [+| [full_path]]
[{-TRACE|-TRACEERR} [+| [full_path]]])
```

Using DBQUERY without a database/query file

```
DBQUERY ("SQL_statement",
"-DBTYPE IFMX
-SOURCE datalink
[-USER username]
[-PASSWORD password]
[-CCARD|-CSTMT [number]]
[-ROWCNT row_count]
[-AUDIT[G] [+| [full_path]]
[{-TRACE|-TRACEERR} [+| [full_path]]])''
```

GET function

The GET function returns the data from the source adapter.

Using GET with a database/query file

```
GET ("DB",
"-MDQ mdq_file
-DBNAME database_name
-QUERY query_name|-STMT SQL_stmt
[-FILE [directory]]
[-VAR name=value...]
[-USER username]
[-PASSWORD password]
[-CCARD|-CSTMT [number]]
[-ROWCNT row_count]
[-AUDIT[G] [+| [full_path]]
[{-TRACE|-TRACEERR} [+| [full_path]]])
```

Using GET without a database/query file

```
GET "DB",
"-DBTYPE IFMX
-SOURCE datalink
-STMT SQL_stmt
```

```

[-SOURCE datasource]
[-FILE [directory]]
[-USER username]
[-PASSWORD password]
[-ROWCNT row_count]
[-CCARD|-CSTMT [number]]
[-AUDIT[G] [+][full_path]]
[{-TRACE|-TRACEERR} [+][full_path]]"

```

Adapter commands for a target

This summary shows the syntax of the adapter commands that can be used when defining a data target using an .mdq file or without using one, including both the required and optional adapter commands in the following situations:

- Using the PUT->Target->Command setting in the Map Designer and Integration Flow Designer.
 - Overriding a data source using the Output Source Override - Database execution command (-OD) using a RUN function or on the command line
 - Using the PUT function in map or component rules.
 - [PUT > Target > Command setting](#)
 - [Database execution command: output source override \(-OD\)](#)
 - [PUT function](#)
-

PUT > Target > Command setting

In an output card, when you specify Database as the value for the PUT > Target setting, the Command field is displayed so that you can enter adapter commands.

Database command options when there is a database/query file

```

[-DBTYPE IFMX]
[-SOURCE datalink]
[-PROC procedure_name| -TABLE table_name]
[-USER user_ID]
[-PASSWORD password]
[-CSTMT [number]]
[-DELETE]
[-UPDATE [OFF|ONLY]]
[-BADDATA[+] full_path]
[-AUDIT[G] [+][full_path]]
[{-TRACE|-TRACEERR} [+][full_path]]"

```

Database command options when there is no database/query file

```

-DBTYPE IFMX
-SOURCE datalink
-PROC procedure_name| -TABLE table_name
[-USER user_ID]
[-PASSWORD password]
[-CSTMT [number]]
[-DELETE]
[-BADDATA[+] full_path]
[-AUDIT[G] [+][full_path]]
[{-TRACE|-TRACEERR} [+][full_path]]"

```

Database execution command: output source override (-OD)

Use the Output Source Override - Database execution command (-OD) to designate a database as a target or you can override one or more of the adapter command settings or database definitions in a RUN function or on the command line.

The adapter commands are shown using a single quotation mark, which is the Windows syntax. For non-Windows platforms, use two single quotation marks followed by one single quotation mark and end with one single quotation mark followed by two single quotation marks.

Scenario: compiled map target is a database

Database/query file (.mdq):

```

' [-MDQ mdq_file -DBNAME database_name]
[-PROC procedure_name| -TABLE table_name]
[-USER username]
[-PASSWORD password]
[-CCARD|-CSTMT [number]]
[-DELETE]
[-UPDATE [OFF|ONLY]]
[-BADDATA[+] full_path]
[-AUDIT[G] [+][full_path]]
[{-TRACE|-TRACEERR} [+][full_path]]'

```

No database/query file (.mdq):

```
'-DBTYPE IFMX
-SOURCE datalink
-PROC procedure_name|-TABLE table_name
[-USER username]
[-PASSWORD password]
[-CCARD|-CSTMT [number]]
[-DELETE]
[-UPDATE [OFF|ONLY]]
[-BADDATA[+] full_path]
[-AUDIT[G][+] [full_path]]
[{-TRACE|-TRACEERR}[+] [full_path]]'
```

Scenario: compiled map target is not a database

Database/query file (.mdq):

```
'-MDQ mdq_file
-DBNAME database_name
-PROC procedure_name|-TABLE table_name
[-USER username]
[-PASSWORD password]
[-CCARD|-CSTMT [number]]
[-DELETE]
[-UPDATE [OFF|ONLY]]
[-BADDATA[+] full_path]
[-AUDIT[G][+] [full_path]]
[{-TRACE|-TRACEERR}[+] [full_path]]'
```

No database/query file (.mdq):

```
'-DBTYPE IFMX
-SOURCE datalink
-PROC procedure_name|-TABLE table_name
[-USER username]
[-PASSWORD password]
[-CCARD|-CSTMT [number]]
[-DELETE]
[-UPDATE [OFF|ONLY]]
[-BADDATA[+] full_path]
[-AUDIT[G][+] [full_path]]
[{-TRACE|-TRACEERR}[+] [full_path]]'
```

PUT function

Use the PUT function to pass data to the target adapter.

Using PUT with a database/query file

```
PUT ("DB",
"-MDQ mdq_file
-DBNAME database_name
-PROC procedure_name|-TABLE table_name
[-USER username]
[-PASSWORD password]
[-CCARD|-CSTMT [number]]
[-DELETE]
[-UPDATE [OFF|ONLY]]
[-BADDATA[+] full_path]
[-AUDIT[G][+] [full_path]]
[{-TRACE|-TRACEERR}[+] [full_path]])"
```

Using PUT without a database/query file

```
PUT ("DB",
"-DBTYPE IFMX
-SOURCE datalink
-PROC procedure_name|-TABLE table_name
[-USER username]
[-PASSWORD password]
[-CCARD|-CSTMT [number]]
[-DELETE]
[-BADDATA[+] full_path]
[-AUDIT[G][+] [full_path]]
[{-TRACE|-TRACEERR}[+] [full_path]])"
```

Binding values in DBLOOKUP/DBQUERY

When using a DBLOOKUP or DBQUERY function, use the Bind facility to submit similarly constructed SQL statements to the database server so that the statements are syntactically identical. By binding a value to a placeholder in the SQL statement, the actual syntax of the statement can be made static, which may improve performance. For more information about using bind values in database functions, see the Database Interface Designer documentation.

- [Limitation when using ODBC](#)
- [Specifying the data type](#)

Limitation when using ODBC

When binding values using Informix®, a limitation exists because Informix cannot determine the data type of the column to which the value is being bound at the time the SQL statement is executed. Therefore, the data type of the value being passed must be explicitly specified. Because of this Informix limitation, the Informix database adapter determines the data type of the value according to the following rule:

If the value contains only numeric characters or numeric characters with a decimal point, the value is assumed to be numeric; otherwise, the value is assumed to be text.

This rule provides the expected results in most cases; however, there may be some cases in which the rule may not apply and it may be necessary to specify the exact data type. For example, if the value is a text field consisting solely of numeric characters, the value will be incorrectly interpreted as numeric unless correctly specified.

Specifying the data type

To override the default behavior and to explicitly specify the data type, a data type indicator must precede the bind value. The syntax for this is:

```
:bind([T|N|D], value)
```

The following table shows the data type indicators and the correspondence between each data type indicator and the Informix® data types.

Data Type Indicator	Informix Data Type
T or TEXT	SQL_CHAR
N or NUM or NUMBER	SQL_NUMERIC
D or DATE	SQL_DATETIME

For information about item formats and interpretation, refer to ["Item properties"](#).

As an example of specifying a data type, if you want to bind a date value to a statement, it is always necessary to specify that the value is of data type DATETIME. The SQL statement would be entered as:

```
DBLOOKUP("select Age from Person where DateOfBirth=:bind([DATE], " +  
        DateItem:Row + ")", "DB.mdq", "MyDB")
```

Configuring DSN for Launcher startup

Note: For information about configuring data sources on UNIX systems, refer to your UNIX documentation.

When using the Informix Adapter on a Windows platform, you need to configure data sources. For information about how to do this, see the ODBC Adapter documentation.

Restrictions and limitations

The Database Interface Designer and database adapters offer options and functions for accessing and manipulating data contained within a database. However, there is one restriction for this adapter: database triggers are not supported. You cannot use a data source as an input event trigger for the Launcher.

Return codes and error messages

Return codes and messages are returned when the particular activity completes. Return codes and messages may also be recorded as specified in the audit logs, trace files, execution summary files, and so on.

For information about error codes and messages returned by database-specific adapters, see the database-specific adapter message section of Resource Adapters documentation.

Various troubleshooting tools are available in case you encounter problems while using the database adapters. For example, if you attempt to run a map that uses the adapter and encounter problems or do not receive the expected results, use the following adapter troubleshooting tools:

- adapter audit log (**.log**)
- adapter trace file (**.mtr**)

Java Class Adapter overview

Use the Java Class Adapter to instantiate a Java™ object and to invoke a sequence of methods on the object.

This adapter accesses Java classes from maps. This means, that maps can be created to instantiate objects of a specified class, set values of the public fields for that object and invoke the specified class methods with the specified arguments and retrieve the results that these methods return (as well as public field values that can be modified) so that they can be reused in the map.

You can use the adapter with a Command Server, Launcher, Software Development Kit, or map in a map rule.

- [System requirements](#)
- [Adapter capabilities](#)

- [Java Class Adapter limitations](#)
 - [Command alias](#)
 - [Java Class Adapter commands](#)
 - [Syntax summary](#)
 - [Java Class Importer](#)
 - [Return codes and error messages](#)
-

System requirements

The minimum system requirements and operating system requirements for the Java Class Adapter are detailed in the [system requirements](#). See also the [release notes](#). It is assumed that a Command Server has already been installed on the computer where the adapter is to be installed for runtime purposes.

Adapter capabilities

The following are Java Class Adapter capabilities:

- A card (or GET or PUT rule) represents a Java™ object. If multiple objects are required, multiple cards or rules must be specified.
 - The adapter provides access to public fields that can be set directly in the map.
 - Any (but naturally one) public constructor can be invoked.
 - Any public methods may be invoked on the object, in the order specified in the type tree.
 - Any public static methods may be invoked on the class (even if no object was constructed).
 - Objects created in one card can be used as parameters to methods in subsequent calls. These objects can be created either explicitly, or by invoking a method that returns an object.
 - Objects are identified by reference labels. The "this" attribute is used to assign a label to an explicitly created object. Objects that are returned from method calls are assigned a label by the adapter. This label is of the form "Ref.nn", where nn is a number which uniquely identifies the object.
 - Return values of any of the methods can be returned from a GET call. The importer provides a mechanism to allow the user to select which methods the return values should be provided. These return values can be primitive types (for example, int) or objects. If objects, the public fields are returned together with a reference to the object so that the object can be used in a subsequent call.
 - The public fields of the object after all methods have been invoked on it can be returned.
-

Java Class Adapter limitations

Multidimensional arrays are not supported.

Command alias

Adapter commands can be specified by using a command string on the command line or creating a command file that contains adapter commands. The execution command syntax is:

```
-IM[alias] card_num
-OM[alias] card_num
```

where -IM is the Input Source Override execution command and -OM is the Output Target Override execution command, alias is the adapter alias, and card_num is the number of the input or output card. The following table shows the adapter alias and its execution command.

Adapter	Alias	As Input	As Output
Java™ Class	JAVA	-IMJAVAcard_num	-OMJAVAcard_num

Java Class Adapter commands

The following table lists valid commands for the Java Class Adapter, the command syntax, and whether the command is supported (✓) for use with data sources, targets, or both.

This adapter does not support listening for events.

Note: A command line entry is required for this adapter. If you do not require any commands, enter a single space instead to override this restriction. (In this case the command line would appear empty.)

Command	Syntax	Source	Target
Deserialize (-DS)	-DS		✓
Serialize (-S)	-S		✓
Trace (-T)	-T[+] [full_path]		✓
Trace Error (-TE)	-TE[+] [full_path]		✓

- [Deserialize \(-DS\)](#)
- [Serialize \(-S\)](#)
- [Trace \(-T\)](#)
- [Trace error \(-TE\)](#)

Deserialize (-DS)

Use the Deserialize adapter command (-DS) to deserialize the input data. The adapter returns a deserialized object and stores it in the pool of objects.

-DS

Serialize (-S)

Use the Serialize adapter command (-S) to serialize the input object. Before serialization the adapter checks whether or not the object is in the pool. The object must be in the pool. The adapter then returns the serialized data.

-S

Trace (-T)

Use the Trace adapter command (-T) to produce a diagnostics file. This file contains detailed information about adapter activity. By default, the **javatrace.log** trace file is created in the directory where the map is located.

-T[+] [*full_path*]

Option

Description

+

Append trace information to the existing trace file.

full_path

Creates a trace file with the specified name in the specified directory

Trace error (-TE)

This adapter command produces a trace file containing only the adapter errors that occurred during map execution.

When this adapter command is specified, an adapter trace file is produced by default in the map directory, using the full name of the map file with an **.mtr** filename extension.

You can use this command in the output card (target) and in a GET or PUT function. Use the options listed below to append the trace information to an existing trace file or to specify a filename and path other than the default.

-TE[+] [*full_path*]

Value

Description

+

Append trace information to an existing trace file. (There is no space between TE and +.)

full_path

Create a trace file with the specified name in the specified directory. If you do not specify a directory (path) and filename, the adapter trace file (*map_name.mtr*) is created in the map directory by default.

The Trace Error adapter command (-TE) and the Trace adapter command (-T) are mutually exclusive.

Syntax summary

- [Data sources and targets](#)
- [Using the Java Class Adapter](#)

Data sources and targets

The Java Class Adapter cannot be used for data sources.

The following is the command syntax of the Java Class Adapter commands used for data targets:

[-DS]
[-S]
[-T[+] [*full_path*]]
[-TE[+] [*full_path*]]

Using the Java Class Adapter

Use the Java Class Adapter to instantiate a Java™ object and to invoke a sequence of methods on the object.

Adapter scenarios

The Java Class Adapter cannot be invoked from an input card. The adapter requires data to be passed to it (as defined by a type tree) to enable it to determine what to do. There are therefore two scenarios for using the adapter:

- If no data is required to be returned from the adapter: call the adapter in an output card. Typical scenarios include:
 - Creating an object (invoking its constructor, and optionally setting its fields) to be used in a subsequent card.
 - Invoking methods on an object that do not return data. The success of these methods is typically monitored through the use of exceptions.
 - Data is required to be returned from the adapter: call the adapter in a **GET** function call. The approach here is to build the data to be passed to the adapter in an output card, setting the output adapter to **sink**. Then, in an additional output card, map the root object of this card to the third parameter of the **GET** using the **PACKAGE** call to pass the XML data as a text item.
- [Simple Java class example](#)
[Complex Java class example](#)
[Return values](#)
-

Simple Java class example

The information that the user selects through the Java™ Class Importer is reflected in the type tree that is generated. The type tree contains several elements:

- Class name and options
- Constructor call
- Public fields
- Method calls

The following code is an example of the simple class SubClass (**SubClass.java**):

```
public class SubClass
{
    public int _intField;
    public byte _byteArrayField[];

    public SubClass(int intPar1)
    {
        _intField=intPar1;
        _byteArrayField = new byte[5];
    }
}
```

Install_dir\examples\adapters\javaclas\subclass.mtt reflects this simple example.

- [Type tree elements](#)
 - [Adapter action sequence](#)
-

Type tree elements

The **SubClass.mtt** type tree contains elements for the constructor and the public fields. When used in a map, the **SubClass** group is chosen as the root element.

The purpose of the identifier is so that a reference to this object can be used in subsequent cards (such as passing this object as a parameter to a method call of another object).

Adapter action sequence

This section describes the sequence of actions that the adapter is performing on the Java™ object.

The constructor is mapped followed by the public fields. This represents the order that the adapter considers these elements: it first invokes the constructor to create the object and then sets the public fields according to the values in the type tree.

When this card is executed, the object will be created by calling the constructor with 1 as the value of the constructor's single parameter. Then the public fields are set to the values defined in the map. The object is given the reference **id1**, which may be used in any subsequent card.

If this map is executed, and the output is directed to a file instead of to the Java Class Adapter, it can be seen that the type tree generates an XML representation of the object and its methods:

```
<SubClass inherited="no" returnFields="no" this="id1">
  <Constructor>
    <p1_int type="int">1</p1_int>
  </Constructor>
  <Fields>
    <_intField type="int">5</_intField>
    <_byteArrayField type="byte[]">
      <element>8</element>
      <element>9</element>
    </_byteArrayField>
  </Fields>
</SubClass>
```

Complex Java™ class example

A more complex example is given by the class MainClass (**Mainclass.java**):

```
public class MainClass
{
    public boolean _booleanField;
    public SubClass _SubClassField;
    public String _StringField;

    public MainClass(int intPar1)
    {
        _booleanField = intPar1 > 0 ? true : false;
        _SubClassField = new SubClass((short)0);
    }
    public MainClass(int intPar1, SubClass SubClassPar2)
    {
        _booleanField = intPar1 > 0 ? true : false;
        _SubClassField=SubClassPar2;
    }
    public int function1()
    {
        if (_SubClassField._byteArrayField == null)
            return 0;
        else
            return _SubClassField._byteArrayField.length;
    }
    public SubClass function2(int intPar1)
    {
        SubClass returnValue = new SubClass(2);
        for (int index = 0; index < 5; index++)
        {
            returnValue._byteArrayField[index] = (byte)intPar1;
        }
        return returnValue;
    }
    public char function2(short[] shortArrayPar1)
    {
        return 'A';
    }
    public void function3(int intPar1)
    {
        _StringField = "Value " + intPar1 + "was passed";
    }
    public byte[] function4(SubClass SubClassPar1, String StringPar2)
    {
        _StringField = StringPar2;
        return SubClassPar1._byteArrayField;
    }
}
```

Notice that this class has a field, a constructor parameter and a method parameter that are of the type **SubClass**.

In the type tree (*install_dir\examples\adapters\javaclasses\mainclass.mtt*), the following aspects are noteworthy:

- There is a field of type **SubClass**, the value of which is a reference. This object must be created in a previous card and its reference mapped to the **ref._SubClassField** item.
- The 2 parameter constructor was selected by the user. The second parameter is of type **SubClass** (item **ref._P2_SubClass**). Its value is a text item that is the object reference. This object must be created in an earlier card and assigned a reference value that is then mapped into this item.
- Methods **function1**, **function2**, and **function3** were selected by the user, **function4** was not.
- All the methods were selected by the user to be invoked by the adapter.
- **function1** has no parameters. Since it is represented as a group, and groups must have components, it has a single component called **#NullArgList**. This should be assigned NONE in the map.
- It is possible that two methods with the same name (but different parameter lists) can be invoked within the same card. To facilitate this, the importer will change the names of the groups to unique names by appending .nn, where .nn is a unique number.

For example, if we wanted to invoke two different methods named **function2**, they would appear in the type tree as **function2.1** and **function2.2**.

Return values

When the user chooses to return the return values from method calls, a category entitled **ReturnValues** is created in the type tree.

The **ReturnValues** category includes a group, also called **ReturnValues** which has as components the individual return values from the methods. An example follows:

This data reflects the following, when mapped to a file:

```
<ReturnValues>
<function1.int>2</function1.int>
<function2.1.char>A</function2.1.char>
<function2.2.SubClass>
<ref>Ref.977366</ref>
<intField type="int">2</intField>
<byteArrayField type="byte[]">
<element>3</element>
<element>3</element>
<element>3</element>
<element>3</element>
```

```

<element>3</element>
</_byteArrayField>
</function2.2.SubClass>
<function4.byte_array>
<element>8</element>
<element>9</element>
</function4.byte_array>
<this.MainClass>
<_booleanField type="boolean">true</_booleanField>
<_SubClassField type="SubClass">
<_intField type="int">5</_intField>
<_byteArrayField type="byte[]">
<element>8</element>
<element>9</element>
</_byteArrayField>
</_SubClassField>
<_StringField type="java.lang.String">Another</_StringField>
</this.MainClass>
</ReturnValues>

```

- [Return value rules](#)
-

Return value rules

The following rules apply to return values:

- When only one return value is selected in the importer, and that value is a primitive type (for example, int) or a string. That value is returned without any other syntax. This simplifies mapping, since that value does not require further parsing.
 - When more than one value is returned then the values are decorated with XML tags (as in the above example). To parse this generally requires passing it to another map through a RUN function call.
 - When an object is returned it will always be decorated with XML tags. Its public fields are output, and a unique object reference is returned as an attribute so that this object may be used in subsequent cards.
 - When in the importer the user opted to return the public fields of the object corresponding to the card (the top-level object), then these are returned in the **this** group of the return values.
-

Java Class Importer

Use the Java™ Class Importer to define the methods that will be executed on the object.

The Java Class Importer allows the user to create a type tree that can be used to manipulate Java objects from within a map using the Java Class Adapter. The structure of the type tree explains the sequence of operations that the adapter is performing: constructing the Java object, setting public fields and invoking public methods.

- [Configuring JVM options in the config.yaml file](#)
 - [Running the Java Class Importer](#)
-

Configuring JVM options in the config.yaml file

Importers with the Java™ native interfaces use the JVM options specified in the **config.yaml** (/runtime/JVM Options) file to create Java virtual machine (JVM). You can use the JVM options to tune the importer's behavior, such as setting up big initial or maximum heap size.

Running the Java Class Importer

Use the Java™ Class Importer to create a type tree in order to manipulate Java objects from within a map using the Java Class Adapter.

To run the Java Class Importer:

1. Open the Type Designer.
2. Select Import a type tree in the Startup window and click Next. Or select Import from the Tree menu at anytime while the Type Designer is running. The Importer Wizard dialog appears.
3. Select Java Class and click Next.
The **Java Class Importer Wizard** appears. Follow the steps to automatically generate a type tree for a java class.
4. Type a fully qualified class name of the class (in this example, **java.awt.Scrollbar**) that you want to import and click Next.
Optionally, select the **Include inherited members from the super classes** check box if you want to include inherited members.
5. Select a constructor from the dialog. Click Next.
Note: It is possible to proceed to the next step without selecting a constructor. In that case, only static methods can be involved, or you may use the class instance obtained in the preceding cards to invoke the non-static methods through the **this** item in the type tree.
6. Optionally, select the **Check here if you wish to return the public fields of the object** check box, if you want the adapter to return the public fields of the object. Click Next.
7. Select the methods that you want to be executed by the adapter. Click Next.
You must reorder the methods into the order that you want them to be executed. To execute a method more than once, right-click on it, select Copy, position your cursor where you want to place the method, right-click again and select Paste.

8. Select the methods for which you want return values to be returned by the adapter (in a **GET** call). Click Next.
 9. Enter the name of (or browse for) the type tree that you want to create. Click Next.
The type tree is created and displayed in the Importer Wizard.
 10. Click Finish. The following window appears.
 11. Click Yes if you want to open the generated type tree(s). Click No to close the Importer Wizard and return to the Type Designer application.
-

Return codes and error messages

Return codes and messages are returned when the particular activity completes. Return codes and messages may also be recorded as specified in the audit logs, trace files, execution summary files, and so on.

- [Java Class Adapter messages](#)
-

Java Class Adapter messages

The following is a listing of all the codes and messages that can be returned as a result of using the Java Class Adapter for sources or targets.

Note: Adapter return codes with positive numbers are warning codes that indicate a successful operation. Adapter return codes with negative numbers are error codes that indicate a failed operation.

Return Code	Message
-1001	<p>Java™ exception occurred: <i>exception_description</i> The variable <i>exception_description</i> represents the exception error returned by the Java language.</p>

JAXB Adapter

The JAXB adapter converts Java™ objects to XML and converts XML to Java objects. With the JAXB adapter, you can build Java objects that can be represented with XML, parse the objects, and use them in maps. The JAXB adapter supports all JAXB properties for marshalling and unmarshalling an XML document.

You can use the JAXB adapter on input cards and output cards, and in GET and PUT map rules. A card, GET rule, or PUT rule represents a single Java object. You must use multiple cards to specify multiple objects.

This documentation assumes that you are familiar with Java Architecture for XML Binding (JAXB).

- [System requirements](#)
The command server must be installed on the computer where the adapter is installed for runtime purposes.
 - [Java class requirements](#)
The generated Java classes must be serializable. The `@XmlRootElement` element must be specified in the Java classes that represent the schema.
 - [JAXB adapter command overview](#)
This table summarizes the adapter commands, syntax, and whether you can use the command with data sources, data targets, or both. You can specify the commands on the adapter command line or through the adapter properties.
-

System requirements

The command server must be installed on the computer where the adapter is installed for runtime purposes.

See the [system requirements](#) for details about minimum system requirements for the JAXB adapter.

Java class requirements

The generated Java™ classes must be serializable. The `@XmlRootElement` element must be specified in the Java classes that represent the schema.

You can use the URL location (-U command) to read and write Java objects only when your objects are serializable. To generate serializable objects, specify `<serializable uid="..._home_markdown_jenkins_workspace_Transform_in_SSVSD8_11.0.1_com.ibm.websphere.dtx.adapjaxb.doc_references_r_jaxb_Java_Classes_Requirements_1"/>` in a bindings file under `<globalBindings>` and pass it to the XJC binding compiler with the -b option. For example:

```
xjc -b bindings.xjb
```

The -C adapter command points to the root class. You can annotate your root class with the `@XmlElement`, or the XJC binding compiler can do it for you. To use the XJC binding compiler to annotate the root class, add `<xjc:simple/>` to your bindings file under `<globalBindings>`. Pass the bindings file to the XJC binding compiler by using the -b option.

An example bindings file follows:

```
<?xml version="1.0" encoding="UTF-8"?>
<bindings xmlns="http://java.sun.com/xml/ns/jaxb"
```

```

xmlns:xsi="http://www.w3.org/2000/10/XMLSchema-instance"
xmlns:xjc="http://java.sun.com/xml/ns/jaxb/xjc"
xsi:schemaLocation="http://java.sun.com/xml/ns/jaxb
http://java.sun.com/xml/ns/jaxb/bindingschema_2_0.xsd"
version="2.1">
<globalBindings>
<serializable uid="_home_markdown_jenkins_workspace_Transform_in_SSVSD8_11.0.1_com.ibm.websphere.dtx.adapjaxb.doc_references_r_>
<xjc:simple/>
</globalBindings>
</bindings>

```

The XJC binding compiler produces Java files (.java) from the schema. You manually package the Java files into Java Archive (JAR) files. You must modify the CLASSPATH environment variable for all Java-based adapters that rely on external JAR files. You can edit the CLASSPATH directly, or you can add the JAR files to the config.yaml configuration file. Add JAR file entries to the /runtime/External Jar Files path of config.yaml file and specify the full path to the JAR file. For example:

```

External Jar Files:
jar1: c:/J2EE/lib/j2ee.jar
jar2: c:/mypath/myjar.jar

```

Note: For Windows, use either a forward slash "/" or two back-slashes "\\\" for the directory delimiter in config.yaml.

JAXB adapter command overview

This table summarizes the adapter commands, syntax, and whether you can use the command with data sources, data targets, or both. You can specify the commands on the adapter command line or through the adapter properties.

The commands are not case-sensitive, but the command values are case-sensitive. Choices are separated by the OR symbol | . Optional parameters are enclosed in brackets []. The topics that follow provide the command syntax in railroad track format.

Table 1. JAXB adapter command summary

Command	Syntax	Use with Data Source	Use with Data Target
Class name	-C Java_class_name	Yes	Yes
File location	-U {URL file_name}	Yes	Yes
Object ID	-I ID	Yes	Yes
Encoding	-jaxb.encoding encoding_name	Yes	Yes
Schema location	-jaxb.schemaLocation URI [URI]	Yes	No
No namespace location	-jaxb.noNamespaceSchemaLocation location	Yes	No
Fragment	-jaxb.fragment {true false}	Yes	No
Trace	-T [+] [file_name]	Yes	Yes
Trace errors	-TE [+] [file_name]	Yes	Yes

- Command alias**

Use the JAXB adapter alias to use the JAXB adapter on input and output cards and in GET and PUT map rules.

- Class name (-C) command**

The -C command specifies the Java class name that the new context object must recognize.

- File location (-U) command**

The -U command specifies the URL or the name of a serialized Java object file. The file can be either a source or a target.

- Object ID (-I) command**

The -I command specifies the object ID of the Java object in the IBM Transformation Extender object pool. Use the object pool to save objects and retrieve them later for further map processing.

- Encoding (-jaxb.encoding) command**

The -jaxb.encoding command specifies the character encoding that the adapter is to use when it marshals the XML data.

- Schema location (-jaxb.schemaLocation) command**

The -jaxb.schemaLocation command specifies the xsi:schemaLocation attribute value to place in the marshalled XML output.

- No namespace schema location (-jaxb.noNamespaceSchemaLocation) command**

The -jaxb.noNamespaceSchemaLocation command specifies the xsi:noNamespaceSchemaLocation attribute value to place in the marshalled XML output.

- Generate document-level events (-jaxb.fragment) command**

The -jaxb.fragment command determines whether the marshaller generates document-level events in the XML data.

- Create trace file (-T) command**

The -T command produces a diagnostics file that contains detailed information about adapter activity. The trace file is overwritten each time the trace runs.

- Trace adapter errors (-TE) command**

The -TE command produces a trace file that contains only the adapter errors that occurred during map execution.

Command alias

Use the JAXB adapter alias to use the JAXB adapter on input and output cards and in GET and PUT map rules.

Specify adapter commands by using a command string on the command line or by creating a command file that contains adapter commands. The command syntax is:

```

-IA[alias]card_num
-OA[alias]card_num

```

where:

-IA

The Input Source Override execution command

-OA

The Output Target Override execution command

alias

The adapter alias

card_num

The number of the map card

To override an input card:

-IAJAXB*card_num*

To override an output card:

-OAJAXB*card_num*

Class name (-C) command

The **-C** command specifies the Java™ class name that the new context object must recognize.

-C command syntax

►► -C — *class_name* ►►

class_name

The Java class name that the new JAXBContext class must recognize.

File location (-U) command

The **-U** command specifies the URL or the name of a serialized Java™ object file. The file can be either a source or a target.

-U command syntax

►► -U — *file_name* — *URL* — ►►

file_name

The name of the serialized Java object file.

URL

The location of the serialized Java object file, specified with URL notation. If you omit the URL notation, the location is treated as a file name.

Object ID (-I) command

The **-I** command specifies the object ID of the Java™ object in the IBM Transformation Extender object pool. Use the object pool to save objects and retrieve them later for further map processing.

-I command syntax

►► -I — *object_ID* ►►

object_ID

The object ID of the Java object in the IBM Transformation Extender object pool.

Encoding (-jaxb.encoding) command

The **-jaxb.encoding** command specifies the character encoding that the adapter is to use when it marshals the XML data.

This command is optional. If you omit it, the adapter uses UTF-8 encoding.

-jaxb.encoding command syntax

►► — *encoding* — — *-jaxb.encoding* — — ►►

encoding

The character encoding that the adapter is to use when it marshals XML data. Specify the value as a java.lang.String object class. See your JAXB documentation.

Schema location (-jaxb.schemaLocation) command

The **-jaxb.schemaLocation** command specifies the xsi:schemaLocation attribute value to place in the marshalled XML output.

This command is optional.

-jaxb.schemaLocation command syntax



xsi:schemaLocation

The xsi:schemaLocation attribute to specify in the generated XML data. Specify the value as a java.lang.String object class. See your JAXB documentation.

No namespace schema location (-jaxb.noNamespaceSchemaLocation) command

The **-jaxb.noNamespaceSchemaLocation** command specifies the xsi:noNamespaceSchemaLocation attribute value to place in the marshalled XML output.

This command is optional.

-jaxb.noNamespaceSchemaLocation command syntax



xsi:noNamespaceSchemaLocation

The xsi:noNamespaceSchemaLocation attribute to specify in the generated XML data. Specify the value as a java.lang.String object class. See your JAXB documentation.

Generate document-level events (-jaxb.fragment) command

The **-jaxb.fragment** command determines whether the marshaller generates document-level events in the XML data.

This command is optional. If you omit it, the default is false. Document-level events are not generated in the XML data.

-jaxb.fragment command syntax



true

false

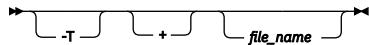
This value must be a java.lang.Boolean object class. The implications of the value depend on which marshaller you use. See your JAXB documentation.

Create trace file (-T) command

The **-T** command produces a diagnostics file that contains detailed information about adapter activity. The trace file is overwritten each time the trace runs.

This command is optional.

-T command syntax



+

Appends trace information to the existing trace file.

file_name

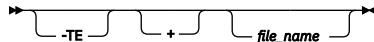
The name and path of the trace file. This parameter is optional. If you omit it, the adapter creates a trace file named m4jaxb.mtr in the IBM Transformation Extender map directory.

Trace adapter errors (-TE) command

The **-TE** command produces a trace file that contains only the adapter errors that occurred during map execution.

This command is optional.

-TE command syntax



+

Appends trace information to the existing trace file.

file_name

The name and path of the trace file. This parameter is optional. If you omit it, the adapter creates a trace file named m4jaxb.mtr in the IBM Transformation Extender map directory.

JDBC Adapter

Overview

Use the Java Database Connectivity (JDBC) adapter to query or update a database through the appropriate JDBC driver for the database.

- [Introduction](#)
Input and output cards and actions of JDBC connection type can be used to access relational database resources.
- [JDBC adapter command overview](#)
- [Authenticating Connections](#)
The adapter supports username/password-based authentication method.
- [Adapter properties and commands](#)
This section lists the properties supported by the adapter.
- [Examples](#)

Introduction

Input and output cards and actions of JDBC connection type can be used to access relational database resources.

JDBC adapter command overview

The following table lists the JDBC adapter commands by function, and specifies whether each command is valid on input and output cards and GET and PUT functions.

Table 1. JDBC adapter functions and commands

Function	Command	Input card	GET function	Output card	PUT function
Configure the JDBC connection and driver	-URL <i>connection_string</i>	✓	✓	✓	✓
	-USERNAME	✓	✓	✓	✓
	-PASSWORD	✓	✓	✓	✓
	-DRIVER	✓	✓	✓	✓
	-AUTOCOMMIT {default on off}	✓	✓	✓	✓
Identify a database table or columns or stored procedure	-CATALOG	✓	✓	✓	✓
	-SCHEMA	✓	✓	✓	✓
	-PROCEDURE	✓	✓	✓	✓
	-TABLE	✓	✓	✓	✓
	-COLUMNS column [, column [, ...]]	✓	✓	✓	✓
Define how to write to or fetch from a database	-DATEFORMAT	✓	✓	✓	✓
	-TIMEFORMAT	✓	✓	✓	✓
	-TIMESTAMPFORMAT	✓	✓	✓	✓
	-WRITEMODE {insert insert_first update update_first}			✓	✓
	-QUERY	✓	✓	✓	
	-QUERYFILE <i>query_file</i>	✓	✓	✓	
	-DML		✓		✓
	-DMLFILE <i>dml_file</i>		✓		✓
	-KEY		✓	✓	✓
	-BIND		✓	✓	✓
	-ROWLIMIT <i>row_limit</i>	✓	✓	✓	✓
	-NAMEDPARAMS	✓	✓	✓	✓
	-EXCLUDEREADONLY	✓	✓	✓	✓
	-SETCOLUMNS column [, column [, ...]]			✓	✓
	-USECOLUMNLABELS	✓	✓	✓	✓
Configure SQL statements to run before or after fetching records from the database	-TRUNCATE			✓	✓
	-MISSINGVALUemode {null_value empty_string}			✓	✓
	-LSN	✓	✓		
	-PRESQL	✓	✓	✓	✓
	-PRESQLFILE <i>pre_sql_file</i>	✓	✓	✓	✓

Function	Command	Input card	GET function	Output card	PUT function
Specify the number of rows to fetch from or send to the database	-FETCHSIZE	✓	✓		
	-BATCHSIZE			✓	✓
	-QTY	✓	✓		
Identify the table where the adapter stores invalid input records	-ERRORROWHANDLING {none error_table error_file}		✓	✓	✓
	-ERRORCATALOG		✓	✓	✓
	-ERRORSCHEMA		✓	✓	✓
	-ERRORTABLE		✓	✓	✓
	-ERRORFILENAME error_file_name		✓	✓	✓
	-ERRORROWDATA		✓	✓	✓
	-CHARSET	✓	✓	✓	✓
Specify character-set encoding	-SQLFILESCARSET	✓	✓	✓	✓
	-T[E V]?[+]? [file_name]	✓	✓	✓	✓
Configure adapter tracing					
Define how to fetch rows from a database (managed automatically by the adapter)	-READMODE {select_all lookup }		✓		
Specify the number of rows fetched from the database to concatenate and return as a single fetch unit (managed automatically by the adapter)	-ARRAYSIZE array_size	✓	✓		

Authenticating Connections

The adapter supports username/password-based authentication method.

JDBC adapter as a source

The JDBC adapter as a source supports:

- Reading rows from a table.
- Reading rows for a custom database query statement.
- Reading values and rows from a stored procedure.
- Running custom static SQL statements before and after reading rows.

JDBC adapter as a target

The JDBC adapter as a target supports:

- Writing rows to a table, in insert, update and delete modes.
- Writing rows using a custom data manipulation language (DML) statement.
- Invoking stored procedures with passing arguments from input data.
- Running custom static SQL statements before and after writing rows.
- Truncating table before writing rows.

Adapter properties and commands

This section lists the properties supported by the adapter.

URL

Defines the JDBC driver-specific connection to a database. See the JDBC driver documentation for details. Generally, the command format is:

`-URL jdbc:driver://host:port/database[?custom_settings]`

The corresponding adapter command is `-URL connection_string`.

User

Specifies the username for the connection. This is optional property. For many drivers, you can specify the username as part of the connection string. Some drivers might use authentication other than username and password, and some drivers might not require authentication at all. The corresponding adapter command is `-USR user_name` (or `-USER user_name`).

Password

Specifies the password for the connection. This command is optional for the reasons described in the `-USER` command. The corresponding adapter command is `-PWD password` (or `-PASSWORD password`).

Driver

Specifies the fully-qualified Java™ class name for the java.sql.Driver interface implementation in the JDBC driver, typically provided in the driver documentation. This command is optional. If you omit the `-DRIVER` command, the adapter searches all available drivers to locate the driver specified in the URL connection string. The corresponding adapter command is `-DRV driver_class` (or `-DRIVER driver_class`).

Autocommit

Instructs the adapter how to configure the driver in respect to automatic commit of SQL statements performed by the driver. Default is off. Use the default setting specific to the driver and database.

When set to on, the adapter explicitly requests the driver to enable and perform auto-commit.

When set to off, the adapter explicitly requests the driver to disable and not perform auto-commit.

The corresponding adapter command is -AC (or - AUTOCOMMIT).

Catalog

Specifies the name of the database catalog. It is case-sensitive. The adapter does not include quotation marks around the catalog name. This is an optional property.

The corresponding adapter command is -CAT *catalog_name* (or -CATALOG *catalog_name*).

Schema

Specifies the name of the database schema. It is a case-sensitive. The adapter does not include quotation marks around the schema name. This is an optional property.

The corresponding adapter command is -SCH *schema_name* (or -SCHEMA *schema_name*).

Procedure

Specifies the name of the stored procedure to execute. It is case-sensitive and does not include quotation marks around the procedure name. This command overrides the -TABLE command.

- When used in an input card, do not specify any input arguments. The procedure can return a result set or a single set of values through output arguments and a return value.
- When used in an output card and the procedure returns data, the card is automatically configured for a request and response exchange using schema types that represent input and output data.
- When used in a PUT function, any specified output arguments or return values are ignored, because there is no mechanism to provide those values back to the map.

The corresponding adapter command is -PROC *procedure_name* (or -PROCEDURE *procedure_name*).

Table

Specifies the name of the table or view to access. It is case-sensitive and does not include quotation marks around the table name. The corresponding adapter command is -TBL *table_name* (or -TABLE *table_name*).

Columns

Specifies the name of the columns to access. It is case-sensitive and does not include quotation marks around the column name. The corresponding adapter command is -COLUMNS *column [, column [, ...]]* where column is a comma separated list of columns. The corresponding adapter command is -COL (or -COLUMNS).

Date Format

Specifies the alternate string format for date columns. If not specified, the format is the JDBC default yyyy-mm-dd. The corresponding adapter command is -DF (or -DATEFORMAT).

Time Format

Specifies the alternate string format for time columns. If not specified, the format is the JDBC default hh:mm:ss.[fff...]. The corresponding adapter command is -TF (or -TIMEFORMAT).

Time Stamp Format

Specifies the alternate string format for timestamp columns. If not specified, the format is the JDBC default yyyy-mm-dd hh:mm:ss.[fff...]. The corresponding adapter command is -TSF (or -TIMESTAMPFORMAT).

Write Mode

Specifies the mode of operation to use in PUT map function and output card when the Table property is specified. The default value is Insert. The meaning of the values is as follows:

- Insert: Each input row is inserted to the target table. The corresponding adapter command value is *insert*.
- Update: Each input row is used to update the corresponding row in the target table. The corresponding adapter command value is *update*.
- Insert First: Each input row is inserted to the target table. If the operation fails due to an integrity constraint error (such as a primary key violation), use the input row data to update the matching row in the target table. The corresponding adapter command value is *insert_first*.
- Update First: Each input row is used to update the matching row in the target table. If the table does not contain a matching row, insert the input row into the target table. The corresponding adapter command value is *update_first*.
- Delete: Each input row is used to delete the matching row in the target table. The corresponding adapter command value is *delete*.

The corresponding adapter command is -WM (or -WRITEMODE).

Query

Specifies the text of the query statement to execute when fetching the data. When GET map function is used with the adapter, the query statement text may contain query bind parameters (as '?' characters), and the values for the parameters are provided as the third argument of the GET function, with '|' character used as the value separator. The corresponding adapter command is -QRY (or -QUERY *statement_text*).

Query File

This property is used as an alternative to provide query statements through a file. The SELECT statement for fetching rows from the database is provided through file. Generally, the command format is:

```
-URL jdbc_connection_url -USR username -PWD password -QUERYFILE file_path
```

The corresponding adapter command is -QRYF (or -QUERYFILE).

DML

Specifies the custom Data Manipulation Language (DML) statement to use for writing rows to the database. When specified, the schema, table and write mode properties are ignored. The number of bind parameters in the statement must match the number of fields in the records pushed to the adapter. The corresponding adapter command is -DML *statement_text*.

DML File

This property is used as an alternative to provide DML statements through a file. The custom DML statement for writing rows to the database is provided through file. Generally, the command format is:

```
-URL jdbc_connection_url -USR username -PWD password -DMLFILE file_path
```

The corresponding adapter command is -DMLF (or -DMLFILE).

Key

Specifies the list of column names to use in the WHERE clause of SELECT and UPDATE statements that the JDBC adapter generates automatically in the following scenarios:

- By a GET function when the -TABLE command is specified. The adapter assumes that the fields in the records match the number and order of columns in the generated WHERE clause.
- By a PUT function or output card when the -TABLE command is specified and the -WRITEMODE command specifies INSERT_FIRST, UPDATE, or UPDATE_FIRST. The adapter assumes that the fields in the records match the number and order of columns in the target table.

The -KEY command is ignored in all other scenarios. If not specified, the JDBC adapter inspects the table in the database (specified by the combination of catalog, schema and table properties), determines the columns that comprise the primary key of the table, and lists those columns in the WHERE clause. If a column name in the list contains the exclamation point (!) or comma (,), use the exclamation point (!) as the release character.

The corresponding adapter command is -KEY *name [,name[, ...]]*.

Bind

A list of comma-separated index [type] entries that specify how to bind input-record field values to parameters in the user-defined statement that the adapter performs on the database. The index value specifies the position of the respective parameter in the statement, starting with position 1. The type value specifies the data type to use for binding input data values to the statement parameter indicated by the index position, and is one of the following: TEXT, BINARY, INT8, INT16, INT32, INT64, DECIMAL, FLOAT32, FLOAT64, DATE, TIME, TIMESTAMP. The default value is TEXT. A single space character should be specified between each index value and the corresponding type value. No other whitespace characters are allowed in the list. The -BIND command applies only to the following scenarios:

- A GET function or output card that specifies the -QUERY command.
- A PUT function or output card that specifies the -DML command.

The corresponding adapter command is -BIND *index[type][,index[type][, ...]]*.

Row Limit

Specifies the maximum number of records to return from the database. The value 0 means all available records.

The corresponding adapter command is -RL (or -ROWLIMIT).

Named Parameters

If set parameters in the statement can be named instead of using ? placeholder. Named Parameters should be specified as {name}. This is applicable with QUERY and DML.

The corresponding adapter command is -NP (or -NAMEDPARAMS).

Exclude Read-Only Columns

Decides whether to exclude or include read-only columns, when to generate the schema or when to perform the table operations. It is enabled when source or target object is table. When specified, the read-only columns would be excluded while generating table queries or statements.

The corresponding adapter command is -ERO (or -EXCLUDEREADONLY).

Here is an example of the adapter command line that includes the Exclude Read-Only Columns command:

```
-URL jdbc_connection_url -USR username -PWD password -ERO -CAT catalog_name -SCH schema_name -TBL table_name
```

Set Columns

Specifies the comma-separated list of column names to be included in SET clause. Whitespace characters should not be included unless they are part of the actual column names.

The corresponding adapter command is -SETCOL (or -SETCOLUMNS).

Use Column Labels

Column labels are typically available when importing schemas for SELECT statements which include AS keyword with the columns in the list. This setting is applicable only when generating schemas.

The corresponding adapter command is -UCL (or -USECOLUMNLABELS).

Truncate

This property is applicable only on target action and can only be used with target as Table and Write Mode as Insert and Insert First. When the value is set to true, the adapter truncates the specified table prior to inserting rows to it.

The corresponding adapter command is -TRC (or -TRUNCATE).

Here is an example of the adapter command line that includes the Truncate command:

```
-URL jdbc_connection_url -USR username -PWD password -WM insert -TRC -MVM null_value -CAT catalog_name -SCH schema_name -TBL table_name -BS 1
```

Missing Value Mode

Determines whether to treat missing (empty) string values in the delimited input data as Null value or empty string value. The corresponding adapter command is -MVM (or -MISSINGVALUemode). The supported property values are Null Value and Empty String, and the corresponding adapter command values are *null_value* and *empty_string*.

-LSN {seconds | 0 | S}

Specifies the amount of time, in seconds, that the adapter is to wait to retrieve a database row.

0

The adapter does not wait. If no rows are available to retrieve (for example, if the database table is empty), the adapter returns a warning message. This is currently the only supported value and should always be set when fetching data from the database.

S

The adapter waits for an infinite period.

Pre SQL

Specifies one or more SQL statements to execute after connecting to the database and prior to fetching any rows from the database or writing any rows to the database. The statements are committed by the adapter after the last one is executed, or automatically if Autocommit is specified or the statements are non-transactional (e.g. DDL statements). Bind parameters are not supported – the statements must be static statements. The adapter assumes that individual statements are separated by a semicolon. Each semicolon and backslash character that is integral part of a statement needs to be escaped by a backslash character.

The corresponding adapter command is -PRESQL.

Pre SQL File

This property is used to provide Pre-SQL statements through a file. The file contains one or more SQL statements to execute after connecting to the database and before fetching any rows from the database or writing any rows to the database.

The corresponding adapter command is -PRESQLF (or -PRESQLFILE).

Here is an example of the adapter command line that includes the Pre SQL File command:

```
-URL jdbc_connection_url -USR username -PWD password -WM insert -MVM null_value -CAT catalog_name -SCH schema_name -TBL table_name -BS 1 -PRESQLF file_path
```

Post SQL

Specifies one or more SQL statement to execute after fetching or writing all rows and prior to disconnecting from the database. The statements are committed by the adapter after the last one is executed, or automatically if Autocommit is specified or the statements are non-transactional (e.g. DDL statements). Bind parameters are not supported – the statements must be static statements. The adapter assumes that individual statements are separated by a semicolon. Each semicolon and backslash character that is integral part of a statement needs to be escaped by a backslash character.

The corresponding adapter command is -POSTSQL.

Post SQL File

This property is used as an alternative to provide Post-SQL statements through a file. Specifies one or more SQL statement to execute after fetching or writing all rows and before disconnecting from the database, through file. The corresponding adapter command is -POSTSQLF (or -POSTSQLFILE).

```
-URL jdbc_connection_url -USR username -PWD password -WM insert -MVM null_value -CAT catalog_name -SCH schema_name -TBL table_name -BS 1 -POSTSQLF file_path
```

Fetch Size

Specifies the number of rows per fetch that the JDBC driver retrieves from the database. The fetch size must be a non-negative number. This is an optional property. If you omit it or specify a fetch size of 0, the JDBC driver determines the optimal fetch size.

The corresponding adapter command is -FS *fetch_size* (or -FETCHSIZE *fetch_size*).

Batch Size

Specifies the number of rows of data the adapter writes to the database in a single batch. If the Write Mode property specifies Insert First or Update First, the adapter ignores the specified batch size and uses a batch size of 1 row. The batch size must be a non-negative integer. When the batch size is 0, the adapter collects all records from the transformation engine and sends them to the database in a single batch. This command is optional.

- When batch size is not specified on an output card, the adapter uses a default value of 1 row.
- When batch size is not specified for a PUT function, the adapter uses a default value of 0 and sends all records from the transformation engine to the database.

The corresponding adapter command is -BS *batch_size* (or -BATCHSIZE *batch_size*).

Quantity

Specifies the number of rows to retrieve from a database table. This command is optional. If you omit it, the default is one row. A value of S retrieves all rows. The -QTY adapter command is used when fetching data from the database and it specifies the maximum number of rows to fetch from the database. The default value is 1, and the value S means all available rows. The -QTY command must always be accompanied by the -LSN 0 command. The -QTY and -LSN commands are managed automatically when the adapter is utilized in input card in the Design Server web UI. But when the adapter command line is specified directly, the -QTY and -LSN commands must be included in it manually.

Fetch As

Whether to read from the source in bursts or read the source in its entirety.

Fetch Unit

Specifies the fetch unit. 0 means Fetch All.

When using an API to create or overwrite an input JDBC card:

- When Fetch As is set to Integral mode, set the Fetch Unit to 1 and specify the following commands so that adapter reads and returns all available rows at one time:
 - -QTY 1
 - -LSN 0
 - -ARRAYSIZE 0
- When Fetch As is set to Burst mode, set the Fetch Unit to 1 and specify the following commands so that adapter reads and returns one row per burst:
 - -QTY S
 - -LSN 0

Note: The -ARRAYSIZE command instructs the adapter to concatenate rows that it fetched from the database and return them as a single fetch unit. The special value 0 means concatenate all fetched rows. This command is set automatically by the adapter and does not have a corresponding property in the Design Server.

Error Row Handling

Select the Error Row Handling to set the mode on how the error data would be handled. Three modes supported are:

- None - Error records will be ignored and will not be recorded.
- Error Table - Enables the parameters to record the errors in an error table.
- Error File - Enables the parameters to record the errors in an error file.

The corresponding adapter command are -EROWH (or -ERRORROWHANDLING).

The corresponding adapter command values are *none*, *error_table* and *error_file*.

Error Catalog

Specifies the name of the error-table database catalog. It is case-sensitive and does not include quotation marks around the name.

The corresponding adapter command is -ECAT *catalog_name* (or -ERRORCATALOG *catalog_name*).

Error Schema

Specifies the name of the error-table database schema. It is case-sensitive and does not include quotation marks around the name.

The corresponding adapter command is -ESCH *schema_name* (or -ERRORSCHEMA *schema_name*).

Error Table

Specifies the name of the error table where the JDBC adapter stores invalid input records. This is an optional property. When the error table is specified, the adapter stores the invalid input record as part of the current transaction and continues to process subsequent input records. When the adapter cannot insert an invalid record into the error table, the adapter reports the error and the map stops executing. When the error table is not specified, the adapter reports the first invalid record and the map stops executing.

The corresponding adapter command is -ETBL *table_name* (or -ERRORTABLE *table_name*).

Define the error table columns to accommodate the following data:

Error table column	Size	Content
1	Variable	The row number as an integer
2	Variable	The error code as an integer
3	16 characters	The SQLstate as string
4	1024 characters	The text of the error message reported by either the JDBC driver or the JDBC adapter.
5	Variable	The raw bytes of the entire invalid input record, including delimiters.

For example:

```
create table error_table (
    rowNum      number,
    vendorCode   number,
    SQLState    varchar2(16 char),
    reason      varchar2(1024 char),
    rawData     blob
);
```

Error File Name

Specifies the name of the Error File in which error records will be captured.

The corresponding adapter command is -EFILEN *error_file_name* (or -ERRORFILENAME *error_file_name*).

Error Row Data

Specifies the Error Row Data in the error file.

The corresponding adapter command is -EROWD (or -ERRORROWDATA).

Charset

Specifies the character-set encoding to convert character data between native (core engine) and Java (adapter) components. By default, the adapter uses the default system locale encoding. In most cases, you do not need to change the character-set encoding. Custom encoding is applicable only in advanced scenarios. For example, when you exchange binary data with the adapter directly, the data is interpreted as text data and is encoded in a non-default encoding.

The corresponding adapter command is -CS (or -CHARSET).

SQL Files Charset

Specifies the character set for converting contents of SQL statement files between native format and driver Java format. By default the current system locale charset is used.

The corresponding adapter command is -SFCS (or -SQLFILESCCHARSET).

Optional Fields Mode

Specifies the cardinality of field components of the row group in the imported schema. The default value is Always, resulting in fields being always marked as optional (0:1). When set to Mandatory, the fields are always marked as mandatory (1:1). When set to Nullable Columns, the fields are marked optional only if they are found to be nullable. This property applies only to schema generation operation in the Design Server UI. It is not applicable to read and write operations performed during a map run.

Logging

This property specifies the level of logging to use for the log (trace) file produced by the adapter. The default is Off. The value Informational means log informational, the value Errors Only means log error messages only, and the value Verbose means log debug and trace level messages along with the informational and error messages.

The corresponding adapter command is:

-T [E|V] [+] *[file_path]*

-T -> Log adapter informational messages.

-TE -> Log only adapter errors.

-TV -> Use verbose (debug) logging. The log file records all activity that occurs while the adapter is producing or consuming messages.

+ -> Appends the trace information to the existing log file. Omit this argument to create a new log file.

file_path -> The full path to the adapter trace log. If you omit this keyword, the adapter creates the m4jdbc.mtr log file in the map directory.

Log file path

Specifies the location of the log file to which to write log messages. If not specified, the default log file name m4jdbc.mtr is used, and the file is stored to the directory in which the executed compiled map resides.

Failure Action

Specifies the action to perform on the transaction when the transaction fails. The default value is Rollback.

The following actions are supported:

- Commit: This option will commit the transaction in case of transaction failure.
- Rollback: This option will roll back the transaction in case of transaction failure.

Examples

[JDBC adapter -KEY command examples](#)

These JDBC adapter examples illustrate the effect of the **-KEY** command in a **GET** function and in an output card.

[JDBC adapter -BIND command examples](#)

These JDBC adapter examples illustrate the effect of the **-BIND** command in a **GET** function and in an output card and custom Data Manipulation Language (DML) statement.

JDBC adapter -KEY command examples

These JDBC adapter examples illustrate the effect of the **-KEY** command in a **GET** function and in an output card.

Example of the -KEY command used in a GET function

Assume the table in the database is defined as follows:

```
table1(c1 int, c2 varchar(10), c3 date, primary key(c1, c3))
```

The adapter is used in the following **GET** function:

```
GET("JDBC", "-URL connection -USER username -PASSWORD password -TABLE table1", inputdata)
```

Because the **-KEY** command is not specified, the adapter uses columns **c1** and **c3** from the primary key of **table1**, and generates a **SELECT** lookup statement as follows. The fully qualified table name and quoted identifiers are omitted for clarity.

```
SELECT c1, c2, c3 FROM table1 WHERE c1 = ? and c3 = ?
```

The adapter assumes that the input data is a record with fields that match the order and names of the columns in the primary key, meaning the **c1** field followed by the **c3** field.

However, when the adapter command line includes the **-KEY** command as follows:

```
GET("JDBC", "-URL connection -USER username -PASSWORD password -TABLE table1 -KEY c2,c3", inputdata)
```

The adapter generates the **SELECT** lookup statement as follows. The fully qualified table name and quoted identifiers are omitted for clarity.

```
SELECT c1, c2, c3 FROM table1 where c2 = ? and c3 = ?
```

The adapter assumes that input data is a record with the fields that match the order and names of the columns in the **-KEY** command, meaning the **c2** field followed by the **c3** field.

Example of the -KEY command used in an output card

Assume the table in the database is defined as follows:

```
table1(c1 int, c2 varchar(10), c3 date, primary key(c1, c3))
```

The JDBC adapter is used in an output card with the adapter command line defined as follows:

```
-URL connection -USER username -PASSWORD password -WRITEMODE INSERT_FIRST -TABLE table1
```

Because the **-KEY** command is not specified, the adapter uses columns **c1** and **c3** as the primary key of **table1**, and generates **INSERT** and **UPDATE** statements as follows. The fully qualified table name and quoted identifiers are omitted for clarity.

```
INSERT INTO table1(c1, c2, c3) VALUES (?, ?, ?)
UPDATE TABLE table1 SET c2 = ? WHERE c1 = ? AND c3 = ?
```

However, if the adapter command line in the output card includes the **-KEY** command as follows:

```
-URL connection -USER username -PASSWORD password -WRITEMODE INSERT_FIRST -TABLE table1 -KEY c2,c3
```

The adapter generates the following **INSERT** and **UPDATE** statements. The fully qualified table name and quoted identifiers are omitted for clarity.

```
INSERT INTO table1(c1, c2, c3) VALUES (?, ?, ?)
UPDATE TABLE table1 SET c1 = ? WHERE c2 = ? AND c3 = ?
```

Regardless of whether the **-KEY** command is specified, the adapter assumes that input data consists of records with fields that match the order and names of the columns in the target table, meaning **c1**, followed by **c2** followed by **c3**.

JDBC adapter -BIND command examples

These JDBC adapter examples illustrate the effect of the **-BIND** command in a **GET** function and in an output card and custom Data Manipulation Language (DML) statement.

Example of the -BIND command used in a GET function

The JDBC adapter is used in a **GET** function as follows:

```
GET("JDBC", "-URL connection -USER username -PASSWORD password -QUERY ""SELECT C1, C2, C3 FROM table1 WHERE c4 = ? AND c1 = ?""", inputdata)
```

Because the **-BIND** command is not specified, the adapter assumes that the input data consists of a record with two fields:

- The first field is mapped to the first parameter. It corresponds to the column **c4** in the **WHERE** clause.
- The second field is mapped to the second parameter. It corresponds to the column **c1** in the **WHERE** clause.

The adapter interprets both fields and binds both parameters as text values.

When the **GET** function is defined with the **-BIND** command as follows:

```
GET("JDBC", "-URL connection -USER username -PASSWORD password -QUERY ""SELECT C1, C2, C3 FROM table1 WHERE c4 = ? AND c1 = ?"" -BIND ""2 INT32,1 FLOAT64""", inputdata)
```

The adapter assumes that the input data consists of a record with two fields:

- The first field is mapped to the second parameter. It corresponds to column **c1** in the **WHERE** clause.
- The second field is mapped to the first parameter. It corresponds to column **c4** in the **WHERE** clause.

The adapter interprets the first field as a 32-bit integer (in text representation). It interprets the second field as a double-precision floating-point number (value 8 in text representation, followed by the pipe (|) delimiter, followed by 8 bytes representing the number in binary format).

Example of the -BIND command used in an output card and custom DML

The JDBC adapter is used in an output card with the adapter command line defined as follows:

```
-URL connection -USER username -PASSWORD password -DML "INSERT INTO table1 VALUES (? + 10, TO_TEXT(?), ? + ?)"
```

Because the **-BIND** command is not specified, the adapter assumes that the input data consists of records with four fields each:

- The first field is mapped to the first parameter.
- The second field is mapped to the second parameter.
- The third field is mapped to the third parameter.
- The fourth field is mapped to the fourth parameter.

The adapter interprets all four fields and binds all four parameters as text values. It's likely that this statement will fail to run (depending on the driver) because the plus (+) operator used with some of the parameters might not work with text values.

Now assume the adapter command line is defined with the **-BIND** command as follows:

```
-URL connection -USER username -PASSWORD password -DML "INSERT INTO table1 VALUES (? + 10, TO_TEXT(?), ? + ?)" -BIND "1 INT8,2 DATE,4 INT16,3 INT64"
```

The adapter assumes that the input data consists of records with four fields each:

- The first field is mapped to the first parameter.
- The second field is mapped to the second parameter.
- The third field is mapped to the *fourth* parameter.
- The fourth field is mapped to the *third* parameter.

The adapter interprets the fields and binds the parameters as follows:

- The first field is bound to the first parameter as an 8-bit integer.
- The second field is bound to the second parameter as a date.
- The third field is bound to the fourth parameter as a 16-bit integer.
- The fourth field is bound to the third parameter as a 64-bit integer.

JMS Adapter overview

Use the Java™ Message Service (JMS) Adapter as a client to any JMS-compliant messaging system provider.

JMS is part of the Java™ 2 Platform Enterprise Edition (J2EE) standards. It defines a standard API for accessing enterprise-messaging systems. Its purpose is to insulate an application from the underlying messaging system in much the same way that Java Database Connectivity (JDBC) insulates an application from a specific database.

A number of JMS implementations exist for messaging-oriented middleware (MOM) systems. Some provide a JMS layer on top of existing systems, for example, MQ Series from IBM® and TIB Rendezvous from TIBCO, while others MOM systems are implemented in Java, and whose primary interface is JMS, for example, FioranoMQ from Fiorano. The JMS API standard allows Java applications to interact with MOM systems that have implemented the JMS-compliant interfaces.

- [System requirements](#)
- [Messaging modes](#)
- [Command Alias](#)
- [JMS Adapter commands](#)
- [Syntax summary](#)
- [Using the Adapter](#)
- [JMS Importer](#)

- [JMS Adapter dependent jars](#)
JMS Adapter dependent jars activemq-all-5.16.2.jar and log4j-1.2.17.jar have been added within the config.yaml file.
 - [Return Codes and Error Messages](#)
-

System requirements

The minimum system requirements and operating system requirements for the Java™ Message Service adapter are detailed in the release notes. It is assumed that a Command Server has already been installed on the computer where the adapter is to be installed for run-time purposes.

Messaging modes

In JMS, a connection factory is the precursor to a connection and this connection is the precursor to a session. It is in the session that messages, message producers and message consumers are created. The message producer sends messages to a destination while a message consumer receives messages from a destination. A destination can be either a topic or a queue.

JMS provides two messaging models as described in the following sections:

- [Point-to-Point \(PTP\)](#)
 - [Publish and Subscribe \(Pub/Sub\)](#)
 - [Point-to-Point \(PTP\)](#)
 - [Publish and subscribe \(Pub/Sub\)](#)
-

Point-to-Point (PTP)

When the destination is a queue, the messaging mode is said to be point-to-point, PTP or P2P, where it is possible to have one or more producers or senders but only one consumer (receiver).

In other words, the possible sender-receiver configurations in this mode are one-to-one and many-to-one. In the PTP mode, the receiver might or might not be active at the time the message is sent. The messaging provider will store the message until the intended consumer has retrieved the message or until the message expires. An important characteristic of this mode is that the consumer always acknowledges the successful processing of a message.

As designed, the JMS API classes are clearly grouped around messaging modes (PTP or Pub/Sub), which makes for easier programming and understanding of the model.

For example, if we know that the destination is a queue, the corresponding QueueConnectionFactory, QueueConnection, QueueSession, QueueSender and QueueReceiver classes must be available to perform messaging operations (send and receive) for the queue destination.

Publish and subscribe (Pub/Sub)

When the destination is a topic, the messaging mode is said to be publish and subscribe (or Pub/Sub) where each message sent by a producer (publisher) can have zero, one, or many consumers (subscribers). The possible publisher-subscriber configurations in this mode are one-to-one, one-to-many, many-to-many and many-to-one.

In the Pub/Sub mode, a message is deleted as soon as it expires or as soon as it has been delivered to all its active subscribers.

There is also a special mode in which we can have a durable subscriber. With this feature, an un-expired message will be delivered to a durable subscriber as soon as it becomes active as long as it was registered as a durable subscriber before the message was published.

Unlike the PTP mode, a provider is not obligated to store a message in the Pub/Sub mode if the intended destination has no active subscriber and no registered durable subscriber.

Successful processing of a message might or might not be acknowledged in this mode. As designed, the JMS API classes are clearly grouped around messaging modes (PTP or Pub/Sub), which makes for easier programming and understanding of the model.

For example, if we know that the destination is a topic, the corresponding TopicConnectionFactory, TopicConnection, TopicSession, TopicPublisher, TopicSubscriber classes must be available to perform messaging operations for the topic destination.

Command Alias

Adapter commands can be specified by using a command string on the command line or creating a command file that contains adapter commands. The execution command syntax is:

```
-IM[alias] card_num  
-OM[alias] card_num
```

where -IM is the Input Source Override execution command and -OM is the Output Target Override execution command, alias is the adapter alias, and card_num is the number of the input or output card. The following table shows the adapter alias and its execution command.

Adapter	Alias	As Input	As Output
JMS	JMS	-IMJMScard_num	-OMJMScard_num

JMS Adapter commands

The following table lists valid commands for the JMS Adapter, the command syntax, and whether the command is supported (✓) for use with data sources, targets, or any combination therein.

Commands can be specified on the adapter's command line or through adapter properties. Note that the commands are not case sensitive but that each value is case sensitive.

For ease of use, most commands have a short form and a long form that is described in the specific command text.

Note: The following commands are all valid in both JMS modes, Publish-and-Subscribe (Pub/Sub) and Point-to-Point (PTP) except where noted.
[1] PTP only

[2] Pub/Sub only

Command	Syntax	Source	Target
ConnectionFactoryName (-CFN)	-CFN <i>factory_name</i>	✓	✓
CorrelationID (-CID)	-CID <i>correlation_id</i>		✓
DeliveryMode (-DM)	-DM !P P		✓
DurableSubscriber (-DS)[2]	-DS <i>sub_name</i>	✓	
Expiration (-E)	-E <i>time_milliseconds</i>		✓
GroupID (-GID)	-GID <i>group_id</i>		✓
Header (-HDR)	-HDR	✓	
InitContextFactory (-ICTXF)	-ICTXF <i>factory_name</i>	✓	✓
InitContextFactoryURL (-ICTXFURL)	-ICTXFURL <i>factory_url</i>	✓	✓
JNDI Method (-JM)	-JM <i>sys env</i>	✓	✓
Listen (-LSN)	-LSN <i>timeout_value</i>	✓	
Priority (-PRI)	-PRI <i>msg_priority</i>		✓
Property (-PRP)	-PRP	✓	
Password (-PW)	-PW <i>password</i>	✓	✓
Queue (-QN)[1]	-QN <i>queue_dest</i>	✓	✓
Quantity (-QTY)	-QTY <i>num_of_messages</i>	✓	
Reply (-R)	-R <i>topic_name queue_name</i>		✓
Selector (-SEL)	-SEL "sel_expression"	✓	
Text (-TT)	-TT		✓
Topic (-TN) [2]	-TN <i>topic_dest</i>	✓	✓
Trace (-T)	-T[E][+] [<i>filename</i>]	✓	✓
UserName (-UN)	-UN <i>user_name</i>	✓	✓

- [ConnectionFactoryName \(-CFN\)](#)
- [CorrelationID \(-CID\)](#)
- [DeliveryMode \(-DM\)](#)
- [DurableSubscriber \(-DS\)](#)
- [Expiration \(-E\)](#)
- [GroupID \(-GID\)](#)
- [Header \(-HDR\)](#)
- [InitContextFactory \(-ICTXF\)](#)
- [InitContextFactoryURL \(-ICTXFURL\)](#)
- [JNDI Method \(-JM\)](#)
- [Listen \(-LSN\)](#)
- [Priority \(-PRI\)](#)
- [Property \(-PRP\)](#)
- [Password \(-PW\)](#)
- [Queue \(-QN\)](#)
- [Quantity \(-QTY\)](#)
- [Reply \(-R\)](#)
- [Selector \(-SEL\)](#)
- [Text \(-TT\)](#)
- [Topic \(-TN\)](#)
- [Trace \(-T\)](#)
- [UserName \(-UN\)](#)

ConnectionFactoryName (-CFN)

Use the ConnectionFactoryName adapter command (-CFN or -CONNECTIONFACTORYNAME) to indicate the Java™ Naming and Directory Interface (JNDI) lookup name of a valid connection factory in an administered object store.

In Point-to-Point (PTP) mode, the *factory_name* is a valid QueueConnectionFactory lookup name. In Publish-and-Subscribe (Pub/Sub) mode, the *factory_name* is a valid TopicConnectionFactory lookup name.

This is a required command.

```
-CFN factory_name
```

Option**Description**

factory_name

Required value. Specify the name of a valid connection factory name.

For example, to specify **PhoenixRising** as the connection factory:

```
-CFN PhoenixRising
```

CorrelationID (-CID)

Use the CorrelationID adapter command (-CID or -CORRELATIONID) to specify a string to send with the message.

This is an optional command. The default is a null CorrelationID.

```
-CID correlation_id
```

Option**Description**

correlation_id

Specify a correlation ID to send with the message.

For example, to specify **Washington** as the correlation ID:

```
-CID Washington
```

DeliveryMode (-DM)

Use the DeliveryMode adapter command (-DM or -DELIVERYMODE) to indicate the mode of delivery for messaging. For non-persistent messaging, the value is !P. For persistent messaging the value is P.

This is an optional command. The default is javax.jms.Message.DEFAULT_DELIVERY_MODE, making the behavior provider-specific.

```
-DM !P | P
```

Option**Description**

!P

Specifies non-persistent delivery mode for sent messages.

Indicates no guarantee that a sent message will be delivered and it is acceptable that intended receivers miss messages arising from provider failure.

P

Specifies once-only persistent delivery mode for sent messages. This guarantees once-only delivery of messages.

For example, to specify **persistent** as the delivery mode value:

```
-DM P
```

DurableSubscriber (-DS)

Use the DurableSubscriber adapter command (-DS or -DURABLESUBSCRIBER) to specify the name of the durable subscription. Useful only in Pub/Sub mode—that is, when the destination is a topic. It is ignored when the destination is a queue.

This is an optional command. The default is no durable subscriber (that the subscription spans the current session only).

```
-DS sub_name
```

Option**Description**

sub_name

Specify the durable subscription name for the current session.

For example, to specify **Reuters** as the durable subscription name:

```
-DS Reuters
```

Expiration (-E)

Use the Expiration adapter command (-E or -EXPIRATION) to specify the time-to-live in milliseconds for how long an undelivered message might be retained by the provider before it expires. The value is **t \$ 0**.

t = 0 specifies no expiration (message lives "forever").

t > 0 specifies that a sent message can live up to t milliseconds after it is sent before it expires.

This is an optional command. The default is javax.jms.Message.DEFAULT_TIME_TO_LIVE, which implies infinite message life in most cases.

-E time_milliseconds

Option

Description

time_milliseconds

Specify the time-to-live in milliseconds for how long an undelivered message might be retained by the provider.

For example, to specify **3000** milliseconds as the expiration time for an undelivered message to be retained:

-E 3000

GroupID (-GID)

Use the GroupID adapter command (-GID or -GROUPID) to specify the ID that marks multiple messages as belonging to a group.

This is an optional command. The default is a null GroupID.

The -GID adapter command only assigns a specified **group_id** value to the **JMSXGroupID** property of the message. If it is necessary to track the sequence numbers for the messages, the **JMSGroupSeq** property of Java™ **int** type must be utilized. In other words, a type tree with the **JMSXGroupSeq** property must be used.

Additionally, if it is necessary to specify the *last* message of the group, additional properties must be used, and this is defined by the message provider.

For example, to specify the last message in a group sequence in IBM® MQ, the IBM MQ JMS-specific property, **JMS_IBM_Last_Msg_In_Group** of Java **boolean** type must be used.

Note that if you need to use the type tree with JMS properties to specify grouped messages (for example, to specify sequence numbers), you have a choice of using the -GID adapter command or the **JMSXGroupID** property of **java.lang.String** type to specify the group identifier value.

-GID group_id

Option

Description

group_id

Specify the ID that marks multiple messages as belonging to a group.

For example, to specify an ID of **JMSTotal** for the group:

-GID JMSTotal

Header (-HDR)

Use the Header adapter command (-HDR or -HEADER) to extract standard header fields to include them in the output in addition to the body of the message. This is valid only in source context.

This is an optional command. Default is null, that is, header fields are not added to the retrieved message.

-HDR

For example, to extract the header fields and include them in the output:

-HDR

InitContextFactory (-ICTXF)

Use the InitContextFactory adapter command (-ICTXF or -INITIALCTXFACTORY) to indicate the initial context factory provider needed to build the starting point of all required JNDI services.

This is a required command.

-ICTXF factory_name

Option

Description

factory_name

Required value. Specify the factory name of the initial context provider. There are no restrictions on the factory name; it can be any string.

For example, to specify an LDAP provider:

```
-ICTXF com.sun.jndi.ldap.LdapCtxFactory
```

InitContextFactoryURL (-ICTXFURL)

Use the InitContextFactoryURL adapter command (-ICTXFURL or -INITIALCTXFACTORYURL) to indicate the initial context factory server URL needed to build the starting point of all required JNDI services.

This is a required command.

```
-ICTXFURL factory_url
```

Option

Option	Description
factory_url	Required value. Specify the factory URL name of the initial context server.

For example, to specify a file based directory:

```
-ICTXFURL file:///localhost/mydir/STORE
```

JNDI Method (-JM)

Use the optional JNDI (Java™ Naming and Directory Interface) Method adapter command (-JM or -JNDIMETHOD) to indicate how the JNDI environment properties are configured. For system properties the value is **sys**. For local environment properties, use the value **env** (default).

This is an optional command.

```
-JM sys|env
```

Option

Option	Description
env	Specify use of the environment properties for the Internet service provider. This is the default value.
sys	Specify use of the system properties (JNDI SPI is the J2EE reference server) for the provider.

For example, to specify **J2EE system properties** as the reference server:

```
-JM sys
```

Listen (-LSN)

Use the Listen adapter command (-LSN) to provide a timeout value in source contexts for retrieving messages.

This is an optional command. The default is infinity.

Note: The -LSN S and -QTY S commands cannot be used together. If you specify -LSN S, you need to specify a specific value for -QTY.

```
-LSN timeout_value
```

Option

Option	Description
timeout_value	Specify the time out value in seconds.

For example, to specify **60** seconds as the timeout value for retrieving messages:

```
-LSN 60
```

Priority (-PRI)

Use the Priority adapter command (-PRI or -PRIORITY) to specify the priority of a message. The value must be an integer between 0 (lowest priority) and 9 (highest priority).

This is an optional command. The default is **message.DEFAULT_PRIORITY**.

-PRI msg_priority

Option

Description

msg_priority

Specify the message priority integer.

For example, to specify **5** as the message priority for a message:

-PRI 5

Property (-PRP)

Use the Property adapter command (**-PRP** or **-PROPERTY**) to extract message properties to include them in the output in addition to the body of the message. This is valid only in source context.

This is an optional command. Default is null, that is, properties are not added to the retrieved message.

-PRP

For example, to extract the message properties and include them in the output:

-PRP

Password (-PW)

Use the Password adapter command (**-PW** or **-PASSWORD**) to indicate an acceptable password to be used for authentication when connecting to the service provider.

This is an optional command. The default is null.

-PW password

Option

Description

password

Specify the password for authentication purposes.

For example, to specify **wizards** as the password:

-PW wizards

Note: The **-PW** command can be used only if the JMS provider supports user authentication.

Queue (-QN)

Use the Queue adapter command (**-QN** or **-QUEUENAME**) to indicate the JNDI lookup name of a valid queue in the administered object store.

All messaging operations for this parameter are conducted in the PTP mode. The value is the JNDI name of an actual queue object (administered object).

This is a required command for PTP mode.

-QN queue_dest

Option

Description

queue_dest

Required value. Specify the JNDI name of an actual queue object (administered object).

For example, to specify **BigDog** as the actual queue object (administered object):

-QN BigDog

Quantity (-QTY)

Use the Quantity adapter command (**-QTY**) to provide the number of messages to return in source contexts.

This is an optional command. The default is **1**.

Note: The `-LSN S` and `-QTY S` commands cannot be used together. The default for `-LSN` is infinity. Therefore, if you specify `-QTY S`, you need to specify a specific value for `-LSN`.

`-QTY num_of_messages`

Option

Description

`num_of_messages`

Specify the number of messages to return in integer form.

For example, to specify **50** as the number of messages to return in source contexts:

`-QTY 50`

Note that when the JMS Adapter is used in the Map Designer to retrieve messages from the queue, every map that executes will leave the message cursor behind the last retrieved message.

For example, if `-QTY 3` is used in input card, and there are 5 message on the queue, the first execution of the map will receive three messages from the queue. If the same map is run again in the same Map Designer session, it will not complete since there will be only two messages left behind the cursor. Only when an additional message arrives to the queue will the map complete because the `-QTY 3` condition is then satisfied.

The second map would also complete if an `LSN` value was specified, in which case only two messages would be returned. For example, if `-LSN 10` was specified, the second map would try to receive the third message for 10 seconds in order to satisfy the `-QTY 3` condition, however, after 10 seconds it would complete and return only two messages because the `-LSN 10` condition was met first.

Reply (-R)

Use the Reply adapter command (`-R` or `-REPLY`) to specify the destination to send a reply.

In source context, the value is a string to send to the non-null destination found in the `JMSReplyTo` header.

In target context, the value is the topic name or queue name (an administered object) to which a receiver sends a reply.

This is an optional command. The default is a null Reply.

`-R topic_name|queue_name`

Option

Description

`topic_name`

Specify the name to which a receiver sends a reply.

`queue_name`

Specify the name (an administered object) to which a receiver sends a reply.

For example, to specify **irongate** as the topic name (for target context only) to which a receiver sends a reply:

`-R irongate`

Selector (-SEL)

Use the Selector adapter command (`-SEL`) to specify a valid message selector expression. The value is a valid selector expression. Since this might contain spaces and single quotes, the value should be surrounded by double quotes in order to not confuse the parser.

This is an optional command. The default is no selector that implies select all.

`-SEL "sel_expression"`

Option

Description

`"sel_expression"`

Specify the selector expression being used.

For example, to specify to selectively process all messages whose application-specific property is **Age minus 20 greater than 4** as the selector expression:

`-SEL "Age - 20 > 4"`

Trace (-T)

Use the Trace adapter command (`-T` or `-Trace`) to produce a diagnostics file that contains detailed information about JMS Adapter activity.

The default filename is **m4java.mtr** and it is created in the map directory unless otherwise specified.

-T [E] [+] [*filename*]

Option

Description

E

Produce a trace file containing only the adapter errors that occurred during map execution.

+

Append trace information to the existing trace file.

filename

Create a trace file with the specified name in the specified directory.

Note: You can override the adapter command line trace options dynamically using the Management Console. See "Dynamic Adapter Tracing" in the *Launcher* documentation for detailed information.

Text (-TT)

Use the Text adapter command (-TT or -TEXT) to instruct the adapter to treat the data that is passed as text and send it as a text message.

If this option is *not* specified and the data does not start and end with <Message> and </Message>, respectively (noting that the double quotes do not count, but the space character after the <Message> value does), the data is assumed to be a byte stream. The message is sent as a Bytes message. Otherwise, the operation fails. This command should be used only for a target (for sending messages).

This is an optional command.

-TT

For example, to treat the data that is passed as text and send it as a text message:

-TT

Topic (-TN)

Use the Topic adapter command (-TN or -TOPICNAME) to indicate the JNDI lookup name of a valid topic in the administered object store.

All messaging operations for this parameter are conducted in the Pub/Sub mode. The value is the JNDI name of an actual topic object (administered object).

This is a required command for the Pub/Sub mode.

-TN *topic_dest*

Option

Description

topic_dest

Required value. Specify the JNDI name of an actual topic object (administered object).

For example, to specify **BigDog** as the actual topic object (administered object):

-TN **BigDog**

UserName (-UN)

Use the UserName adapter command (-UN or -USERNAME) to indicate an acceptable user name to use for authentication when connecting to the service provider.

This is an optional command. The default is null.

-UN *user_name*

Option

Description

user_name

Specify the user name for authentication purposes.

For example, to specify **JamesBond** as an acceptable user name to be used for authentication purposes:

-UN **JamesBond**

Note: The -UN command can be used only if the JMS provider supports user authentication.

Syntax summary

- [Data sources](#)
 - [Data targets](#)
-

Data sources

The following is the command syntax of the JMS Adapter commands used for data sources:

```
-CFN factory_name
-ICTXF factory_name -ICTXFURL factory_url
-QN queue_dest| -TN topic_dest [-DS sub_name]
[-HDR]
[-JM sys|env]
[-LSN timeout_value]
[-PRP]
[-PW password]
[-QTY num_of_messages]
[-SEL "sel_expression"]
[-T[E][+] [filename]]
[-UN user_name]
```

- [Example 1](#)
-

Example 1

For the GET > Source setting in an input card, select JMS. In the GET > Source > Command field, enter:

```
-T -HDR -ICTXFURL file:///C:/jdkmq/STORE -ICTXF
com.sun.jndi.fscontext.RefFSContextFactory -CFN myQueueCF1 -QN
myQ1
```

This command creates a trace file and outputs a header. In addition, it specifies the following:

- the initial context factory URL as file-based object **STORE**
 - the initial context factory as **com.sun.jndi.fscontext.RefFSContextFactory**
 - the connection factory JNDI lookup name as **myQueueCF1**
 - sets the JNDI queue lookup name as **myQ1**
-

Data targets

The following is the command syntax of the JMS Adapter commands used for data targets:

```
-CFN factory_name
-ICTXF factory_name -ICTXFURL factory_url
-QN queue_dest| -TN topic_dest
[-CID correlation_id]
[-DM !PIP]
[-E time_milliseconds]
[-GID group_id]
[-JM sys|env]
[-PRI msg_priority]
[-PW password]
[-R topic_name|queue_name]
[-T[E][+] [filename]]
[-TT]
[-UN user_name]
```

- [Example 2](#)
-

Example 2

For the PUT > Target setting in an output card, select JMS. In the PUT > Target > Command field, enter:

```
-CID 12345 -ICTXFURL file:///C:/jdkmq/STORE -ICTXF
com.sun.jndi.fscontext.RefFSContextFactory -CFN
"cn=myQueueCF1" -QN "cn=myQ1"
```

This command sets the correlation ID value to **12345**. In addition, it specifies the following:

- the initial context factory URL as file-based object **STORE**
- the initial context factory as **com.sun.jndi.fscontext.RefFSContextFactory**
- the connection factory JNDI lookup name as **myQueueCF1**
- sets the JNDI queue lookup name as **myQ1**

Using the Adapter

Use the JMS Adapter to act as a client to any JMS-compliant messaging system provider.

- [Message content](#)
- [Message Selection](#)
- [Example Files](#)

Message content

A JMS message content is made up of the following three sections:

- Header ([Header](#))
 - Properties ([Properties](#))
 - Body or Payload ([Body or Payload](#))
-
- [Header](#)
 - [Properties](#)
 - [Body or Payload](#)

Header

The header fields are present in every message and common to all JMS implementations.

If the `-HDR` command is used in a source, the header fields are returned from the adapter. If the header group is used in the type tree of a target, then values can be mapped to the header fields.

Note that any adapter commands will override any values that are mapped to these fields and that all fields are passed between the engine and the adapter as UTF-8 data.

- [Header Fields Table](#)

Header Fields Table

The following table indicates how these fields are set and used:

Field	Set by	Significant Context
JMSDestination	Provider	Target
JMSDeliveryMode	Sender	Target
JMSMessageID	Provider	Source
JMSTimestamp	Provider	Source
JMSCorrelationID	Sender	Source, Target
JMSReplyTo	Sender	Source, Target
JMSRedelivered	Provider on receipt	Source
JMSType	Sender	Target
JMSExpiration	Provider ¹	Target
JMSPriority	Provider	Target

¹The **JMSExpiration** field is set by the Provider based on the current GMT time and the **TimeToLive** setting input by the sender.

Only those fields that can be set by the sender can be set in a target. If values are passed to other fields, they will be ignored.

Properties

Properties are a mechanism to add application and provider specific properties to extend the header. Properties enable the JMS provider or the user with the capability to attach more information to a message. Properties are subdivided into the following three categories:

- application-specific ("[Application-specific Properties](#)")
 - JMS-specific ("[JMS-specific Properties](#)")
 - vendor-specific ("[Vendor-specific Properties](#)")
-
- [Application-specific Properties](#)
 - [JMS-specific Properties](#)
 - [Vendor-specific Properties](#)
 - [Using Message Properties](#)

Application-specific Properties

This category includes properties set by the user using the application developer (the adapter). Users can set as many user-defined properties as permitted by resources and the provider. The user nomenclature for application-specific properties must avoid using the contiguous set of characters "JMSX" or "JMS_" that are reserved for JMS-specific and vendor-specific properties, respectively.

JMS-specific Properties

This category includes properties, both required and optional, that are predefined by the JMS specification. These properties are named using the format JMSX<property>, that is, each key begins with the prefix "JMSX".

Required properties must be supported by all JMS-compliant providers. The required properties include:

- JMSXGroupID
- JMSXGroupSeq

Optional properties are currently defined and can be set by the provider. The optional properties include:

- JMSXUserID
- JMSXAppID
- JMSXProducerTXID
- JMSXConsumerTXID
- JMSXRcvTimestamp
- JMSXDeliveryCount
- JMSXState

These properties are not required to be supported by providers.

JMS-specific Properties Table

The following table lists the properties and their significant contexts:

Field	Set by	Significant Context
JMSXUserID	Provider on send	Source
JMSXAppID	Provider on send	Source
JMSXDeliveryCount	Provider on receipt	Source
JMSXGroupID	Sender	Target
JMSXGroupSeq	Sender	Target
JMSXProducerTXID	Provider on send	Source
JMSXConsumerTXID	Provider on receipt	Source
JMSXRcvTimestamp	Provider on receipt	Source
JMSXState	Provider	None

Vendor-specific Properties

The third category of properties is the vendor specific properties. The JMS specification makes it easy to identify this category of properties as each, if any, must follow a nomenclature that begins with the prefix "JMS_".

A provider might define as many provider-defined "JMS_" type properties as its resources permit.

Using Message Properties

If the `-PRP` command is used in a source, all the available properties (application-specific, vendor-specific and JMS-specific) are returned from the adapter. If the properties group is used in the type tree of a target, then values can be mapped to the property fields.

Note that any adapter commands will override any values that are mapped to these fields and that all fields are passed between the engine and the adapter as UTF-8 data.

When a type tree is generated using the JMS Importer, the **Properties** group contains a dummy text item, since it is not possible to know which properties will be contained within the JMS message. Therefore, it is necessary to manually create the items in the property group.

The example type tree, `Text_in.mtt` (located in the examples folder) shows an example set of properties that can be used as a template.

Perform the following steps to add properties to a type tree.

To add properties to a Type Tree

1. Create an item in the **Properties** category. The **Item Subclass** should be set according to the following table:

Java" Type	Item Subclass	Type Attribute in Initiator
boolean	Number->Integer	"boolean"
byte	Number->Integer	"byte"
short	Number->Integer	"short"
int	Number->Integer	"int"
long	Number->Integer	"long"

Java™ Type	Item Subclass	Type Attribute in Initiator
float	Number->Decimal	"float"
double	Number->Decimal	"double"
java.lang.String	Text	"java.lang.String"

For all properties except for the properties of java.lang.String type, the initiator of the item that is representing it should be:

```
<Property_Name type="type">
```

and the terminator of the item type should be:

```
</Property_Name>
```

For the properties of java.lang.String type, the initiator should be:

```
<Property_Name type="type"><![CDATA[
```

and the terminator should be:

```
]]></Property_Name>
```

Property_Name is the actual name of the property, and type is one of the values listed in the above table.

<![CDATA[and]]> enable XML content to appear as a string value for the property of java.lang.String type. They ensure that the XML parser ignores the value of this property, that is, that it does not treat it as part of the XML document that represents the JMS message structure. Note further that the property value should not contain the "]]>" substring because it serves as a closing marker for the CDATA section.

2. Add the item to the **Properties** unordered group.

Body or Payload

The third section of a message is its body, also known as the payload. The JMS specification expects JMS providers to implement and support six message interface types. The adapter supports all six of the following types.

- Message ([Message](#))
- Bytes Message ([Bytes Message](#))
- Stream Message ([Stream Message](#))
- MapMessage ([MapMessage](#))
- TextMessage ([TextMessage](#))
- ObjectMessage ([ObjectMessage](#))

- [Message](#)
- [Bytes message](#)
- [Stream message](#)
- [MapMessage](#)
- [TextMessage](#)
- [ObjectMessage](#)

Message

This is a message with an empty body. It is the base interface for all the other message forms. Because it has no payload, it is useful as a control message.

If data of zero length is passed to the adapter the adapter will create a message with no payload.

A message with no payload can have header and property values.

When sending a message with no payload that has specified header and property values, the opening tag of the <Body> element must be specified in one of the following formats:

- <Body>
- <Body type="">
- <Body type="Message">

When the adapter receives a message with no payload, it always omits the "type" attribute of the <Body> element in the returned XML data.

Bytes message

The body is a stream or an array of uninterpreted bytes. The adapter is able to handle data to and from a Bytes message type.

When the adapter receives a Bytes message, it always base64 encodes the body of the message.

When sending a message:

- If -TEXT command is not specified and if the adapter fails to parse the received data as XML, and if this data does not start and end with "<Message " and "</Message>", respectively (the double quotes do not count, but the space character after the <Message value does), it interprets it as binary data, not base64 encoded, and dumps the data into a Bytes message.

- If the adapter locates the <Body> element in the data and it has the type attribute set to "Bytes", the content of the <Body> element is always interpreted as base64 encoded data.

To base64 encode a message to be sent in an output card, use the **TEXTTOBASE64** engine function.

To base64 decode a message received from an input card, use the **BASE64TOTEXT** engine function.

For more information about these engine functions, see the *Functions and Expressions* documentation.

Stream message

The body is a stream of Java™ primitive types that are filled and read sequentially. The essential characteristics is that the primitives must be read in the same order they were written. The adapter can handle data to and from a StreamMessage type.

MapMessage

Here, the body is a set of name-value pairs where the names are always strings but the values can be any Java™ primitive type, their wrapper object, or even strings. The values can either be accessed sequentially or randomly by name. The adapter is able to handle data to and from a MapMessage type.

TextMessage

The body is of type `java.lang.String`. Users can choose this type to process plain text messages such as XML documents. The adapter can handle data to and from a TextMessage type. A text message is represented in a map as UTF-8 data.

ObjectMessage

The payload is a serializable Java™ object. The adapter package can handle data to and from an ObjectMessage type.

Message Selection

JMS provides an SQL-like language to define "selectors". These selectors operate on the headers and properties (user-defined properties, JMS-defined properties and even provider-specific properties) of the message to select only messages that return **TRUE** for the given expression.

For example:

```
JMSMessageID='sj2j22j' AND MyType='payroll'
```

Note: See the [Selector \(-SEL\)](#) command on page [page 1-11](#) for more information.

Example Files

To access an example for this adapter, from the Start menu, select the **Examples** option for this product. In the Examples menu, select the **Resource Adapters** page from the Table of Contents listing and then select the link for this adapter.

JMS Importer

Use the JMS Importer to create type trees that represent the public fields and constructors of Java™ classes that are sent as objects using JMS messages. The generated type tree reflects the selections the user makes, and includes the fields required for the JMS header and properties.

The importer has no relevance for message types other than the *Object* message type. Metadata is not available for *Text*, *Bytes*, *Stream* and *Map* message types. Only template type trees are provided that can be manually modified to reflect the specific messages being processed.

For detailed information on the structure of the type tree as created by the JMS Importer, review the **readme.txt** file located in the JMS example folder. To access this file, from the Start menu, select the **Examples** option for this product. In the Examples menu, select the **Resource Adapters** page and then select the link for this adapter. Open the **readme.txt** file.

- [Configuring JVM Options in the config.yaml File](#)
- [Running the JMS Importer](#)

Configuring JVM Options in the config.yaml File

Importers with the Java™ native interfaces use the JVM options specified in the **config.yaml (/runtime/JVM Options)** file to create Java virtual machine (JVM). You can use the JVM options to tune the importer's behavior, such as setting up big initial or maximum heap size.

See the similarly named chapter in the *Type Tree Importers documentation* for detailed information.

Running the JMS Importer

Use the JMS Importer to create a type tree in order to manipulate Java™ objects from within a map using the JMS Adapter.

To run the JMS Importer:

1. Open the Type Designer application.
2. Select Import a type tree in the Startup window and click Next. Or select Import from the Tree menu at anytime while the Type Designer is running. The Importer Wizard dialog box appears.
3. Select Java Message Service (JMS) and click Next.
The **JMS Importer Wizard** appears. Follow the steps to automatically generate a type tree for a Java class.
4. Enter a fully qualified class name for the Java class file (in this example, **Book**) that you want to import. Click Next. The Java code for the **Book** class is:

```
public class Book implements java.io.Serializable { public String title; public String author; public float price; public boolean instock; public Book(String title, String author) { this.title = title; this.author = author; } public Book(String title, String author, float price, boolean instock) { this(title, author); this.price = price; this.instock = instock; } }
```

When using the JMS importer, the specified Java class (the one you are importing) must implement the `java.io.Serializable` interface; otherwise, the import process will fail.

Note: **Book** class is in the default package for this example. If it were in a named package, the name of the package would precede the name of the class. For example, `com.ibm.mypack.Book`.

JMS Adapter dependent jars

JMS Adapter dependent jars activemq-all-5.16.2.jar and log4j-1.2.17.jar have been added within the config.yaml file.

Return Codes and Error Messages

Return codes and messages are returned when the particular activity completes. Return codes and messages might also be recorded as specified in the audit logs, trace files, execution summary files, and so on.

- [Messages](#)

Messages

The JMS Adapter reports faults with intuitively annotated Java™ exceptions. Execution flow and error messages, if any, are in the ***.mal** file which the user can obtain via the "[Trace \(-T\)](#)" command on page [page 1-11](#).

Please see the ***.mal** file for error and execution flow information.

JNDI Adapter overview

Use the Java™ Naming and Directory Interface (JNDI) Adapter to enable maps to look up, add, modify and delete entries in external directories, and retrieve the results that these operations return so that they can be reused in a map.

The following JNDI service providers are supported:

- LDAP
- COS Naming
- RMI
- DNS
- File System
- WebLogic

Supporting a service provider involves the adapter recognizing it as a valid protocol, and the availability of a supported schema.

The adapter uses XML parsing to parse the stream passed to an output card. The JNDI Adapter uses a standard Java-based XML parser (Xerces).

See your [Oracle documentation](#) for an overview of the JNDI architecture.

- [System requirements](#)
- [Command alias](#)
- [JNDI commands](#)

- [Syntax summary](#)
- [Using the adapter](#)

System requirements

The JNDI Adapter requirements are detailed in the system requirements and the release notes. It is assumed that a Command Server has already been installed on the computer where the adapter is to be installed for runtime purposes.

Command alias

Adapter commands can be specified by using a command string on the command line or creating a command file that contains adapter commands. The execution command syntax is:

```
-IM[alias] card_num
-OM[alias] card_num
```

where **-IM** is the Input Source Override execution command and **-OM** is the Output Target Override execution command, *alias* is the adapter alias, and *card_num* is the number of the input or output card. The following table shows the adapter alias and its execution command.

Adapter	Alias	As Input	As Output
JNDI	JNDI	-IMJNDI <i>card_num</i>	-OMJNDI <i>card_num</i>

JNDI commands

This documentation describes the functions and use of the JNDI Adapter commands and their options.

- [List of commands](#)

List of commands

The following table lists valid commands for the JNDI Adapter, the command syntax, and whether the command is supported (✓) for use with data sources, targets, or both.

Command	Syntax	Source	Target
Authentication Method (-AM)	-AM SSL	✓	✓
InitContextFactory (-ICTXF)	-ICTXF <i>factory_name</i>	✓	✓
InitContextFactoryURL (-ICTXFURL)	-ICTXFURL <i>factory_url</i>	✓	✓
Max Hits (-MAXHITS)	-MAXHITS <i>hits</i>	✓	
Timeout (-TIMEOUT)	-TIMEOUT <i>milliseconds</i>	✓	
Trace (-T)	-T[E][+] [<i>file_name</i>]	✓	✓

- [Authentication Method \(-AM\)](#)
- [InitContextFactory \(-ICTXF\)](#)
- [InitContextFactoryURL \(-ICTXFURL\)](#)
- [Max Hits \(-MAXHITS\)](#)
- [Timeout \(-TIMEOUT\)](#)
- [Trace \(-T\)](#)

Authentication Method (-AM)

Use the Authentication Method adapter command (**-AM** or **-AUTHENTICATION_METHOD**) to use the secure sockets layer (SSL) security protocol for an LDAP connection. This command is optional.

-AM SSL

Option

Description

SSL

Use the SSL protocol for the LDAP connection.

InitContextFactory (-ICTXF)

Use the InitContextFactory adapter command (**-ICTXF** or **-INITIALCTXFACTORY**) to indicate the initial context factory provider needed to build the starting point of all required JNDI services. If the command is omitted, the adapter examines the URL and attempts to load the appropriate initial context factory provider.

This is a required command.

```
-ICTXF factory_name
```

Option

Description

factory_name

Required value. Specify the factory name of the service provider. There are no restrictions on the factory name; it can be any string.

For example, to specify an LDAP provider:

```
-ICTXF com.sun.jndi.ldap.LdapCtxFactory
```

InitContextFactoryURL (-ICTXFURL)

Use the InitContextFactoryURL adapter command (-ICTXFURL, -INITIALCTXFACTORYURL or -URL) to indicate the initial context factory server URL needed to build the starting point of all required JNDI services.

This is a required command.

```
-ICTXFURL factory_url
```

Option

Description

factory_url

Required value. Specify the factory URL name of the initial context provider.

For example, to specify a file-based directory:

```
-ICTXFURL file:///localhost/mydir/STORE
```

Max Hits (-MAXHITS)

Use the Max Hits adapter command (-MAXHITS) to specify the maximum number of hits to be returned in a search request. If not specified, all hits are returned.

```
-MAXHITS hits
```

Option

Description

hits

Specify the maximum number of hits to be returned in a search request.

For example, to specify **10** as the number of hits to be returned in a search request:

```
-MAXHITS 10
```

Timeout (-TIMEOUT)

Use the Timeout adapter command (-TIMEOUT) to specify the time limit for lookup operations in milliseconds. If not specified the timeout is infinite. Not all service providers support user-defined timeouts.

```
-TIMEOUT milliseconds
```

Option

Description

milliseconds

The time limit in milliseconds

For example, to specify **10,000** milliseconds as the time limit for lookup operations.

```
-TIMEOUT 10000
```

Trace (-T)

Use the Trace adapter command (-T) to produce a diagnostics file. This file contains detailed information about adapter activity. By default, the **m4jndi.mtr** trace file is created in the directory where the map is located.

```
-T[E] [+] [file_name]
```

Option	Description
E	Produce a trace file containing only the adapter errors that occurred during map execution.
+	Append trace information to the existing trace file.
file_name	Creates a trace file with the specified name in the specified directory

Trace (-T) and Trace Error (-TE) adapter commands are mutually exclusive.

Syntax summary

This section of the documentation discusses the JNDI syntax summary and how the command syntax is used.

- [Data sources](#)
 - [Data targets](#)
 - [Example 1](#)
 - [Example 2](#)
-

Data sources

The following is the command syntax of the JNDI Adapter commands used for data sources:

```
-ICTXF factory_name -ICTXFURL factory_url -MAXHITS hits -TIMEOUT milliseconds
[-T[E] [+]] [file_name]] [-AM SSL]
```

Data targets

The following is the command syntax of the JNDI Adapter commands used for data targets:

```
-ICTXF factory_name -ICTXFURL factory_url [-T[E] [+]] [file_name]] [-AM SSL]
```

Example 1

For the GET->Source setting in an input card, select JNDI. In the GET->Source->Command field, enter:

```
-TRACE+ myfile.mtr -Timeout 10000
-MAXHITS 10
-url "ldap://localhost/cn=Test, o=IBM Software,c=
US??base?objectclass=*"
```

This command specifies the following:

- enables a trace file called **myfile.mtr** in append mode
 - sets a time limit for lookup operations to 10,000 milliseconds
 - specifies the maximum number of hits to be returned in a search request to ten.
 - specifies the entries to search for on the LDAP server called localhost
-

Example 2

In a GET map function call enter:

```
=GET("JNDI",
"-T+ myfile.mtr
-ICTXF com.sun.jndi.rmi.registry.RegistryContextFactory
-ICTXFURL rmi:// localhost:2500 ", PACKAGE( MyPackage ))
```

This GET map function call will:

- enable a trace file called **myfile.mtr** in append mode
- load the **InitialContextFactory** *com.sun.jndi.rmi.registry.RegistryContextFactory*
- connect to the **RMI server** running on the local machine on port 2500
- perform the JNDI operations that have been defined in Output Card **MyPackage**

Using the adapter

Use the JNDI Adapter to enable maps to look up, add, modify and delete entries in external directories, and retrieve the results that these operations return so that they can be reused in a map.

- [Theory of operation](#)
- [Adapter capabilities](#)
- [Adapter limitations](#)
- [Adapter scenarios](#)
- [Return values](#)
- [Schemas](#)
- [JNDI service providers](#)
- [Interoperability between JNDI adapter and other adapters](#)
- [Binary data](#)
- [Example files](#)

Theory of operation

The adapter uses the Initial Context Factory, which might be explicitly or implicitly supplied on the command line, to initialize the JNDI API. The mandatory url supplied on the command line identifies the host and optionally the port, user, password and which domain to connect to.

There are two major types of JNDI service providers, Naming Services and Directory Services.

- Naming Services such as the RMI Registry are generic enough to share most of their functionality. They are capable of performing lookup, add, delete and modify operations; they do not use attributes; their lookups might return one, zero or many objects.
- Directory Services, such as LDAP and DNS require specific implementations due to their complexity.

The adapter command line can be used to load a specific service provider at runtime.

The adapter uses Java™ Reflection to dynamically load the service provider implementation. Communication between the adapter and the remote server is handled transparently by JNDI.

Adapter capabilities

The JNDI Adapter has the following capabilities:

- A card or `GET` or `PUT` rule represents a JNDI operation. JNDI operations include lookup, add, modify or delete.
- The adapter provides access to records and attributes that can be set in the map, assuming the underlying service provider supports add, modify and delete.
- RDN modification, including renaming, is not supported. However, RDNs can be deleted and added in a modified state, assuming the service provider supports delete and add. An RDN is analogous to the name of the file in a hierarchical file system in Windows or UNIX.
- Objects created in one card can be used as parameters to methods in subsequent calls. These objects must be created by invoking an operation that returns an object.
- Reference labels identify objects. Objects that are returned from method calls are assigned a label by the adapter. The label is of the form `Ref.nn`, where `nn` is a number which uniquely identifies the object.
- The return values of any of the methods can be returned from a `GET` call. The schemas provide a mechanism allowing the user to select which methods the return values should be provided for. These return values can be primitive types, for example, `int`, strings, arrays, or local or remote objects. In the case of remote objects, the reference label is returned so that the object can be used in a subsequent call.
- Objects can be shared between the JNDI, JMS, Java™ Class adapters.

Adapter limitations

The JNDI Adapter has the following limitations:

- Only simple authentication is supported in bind requests.
- Attributes are always retrieved in search requests.
- Aliases are always de-referenced.
- The JNDI Adapter does not support Compare Requests.
- Basic Encoding Rule (BER) values are not supported in modify requests.

Adapter scenarios

The JNDI Adapter can be used in an input or output card. In an input card, the adapter can only perform lookup operations. Data is passed to the adapter on the URL command line option. Each provider implementation uses a different URL format. When used as an output card, data is passed, as defined by a schema, to enable it to determine what to do.

Therefore, there are two scenarios for using the adapter.

- If no data is required to be returned from the adapter, call the adapter in an output card. Typical scenarios include:
 - Add operations

- Modify operations
- Delete operations
- If data is required to be returned from the adapter, call the adapter in a `GET` function call. Build the data to be passed to the adapter in an output card and set the output adapter to 'sink'. Map the root object of this card to the third parameter of the `GET` function using the `PACKAGE` call to pass the XML data as a text item. See ["Return Values"](#) for more information on the format of the data returned from the `GET` function call.

Return values

Lookup operations are expected to return values. The format of the return data is described below.

	No Hits	One Hit	Multiple Hits
Directories	DSML	DSML	DSML
Naming Repositories	No data	Raw data	DSML

Lookups on a specific name in a naming repository might only return one hit. Lookups on a general domain within a naming repository return a collection of hits.

Each object returned from any lookup naming and directories is examined by the adapter. If it is a simple object, its value is returned. If it is not a simple object, it is stored in the Object Pool and its reference id is returned.

Multiple values return data similar to the following:

```
<dsml:dsml xmlns:dsml="http://www.dsml.org/DSML">
<dsml:directory-entries>
  <dsml:entry dn="cn=Test, o=IBM Software, c=US">
    <dsml:objectclass>
      <dsml:oc-value>
        <![CDATA[ top ]]>
      </dsml:oc-value>
    </dsml:objectclass>
    <dsml:attr name="homepage">
      <dsml:value>
        <![CDATA[ www.ibm.com ]]>
      </dsml:value>
    </dsml:attr>
    <dsml:attr name="someObject">
      <dsml:value>
        <![CDATA[ Ref. 786542 ]]>
      </dsml:value>
    </dsml:attr>
  </dsml:entry>
</dsml:directory-entries>
</dsml:dsml>
```

Return data for a single value in a naming context is similar to the following:

Ref. 786542

or

www.ibm.com

Schemas

The following schemas are provided with the JNDI Adapter installation:

- [LDAP](#)
- [DNS](#)
- Generic Naming

The Generic Naming schema can be used to perform [RMI](#) Repository, [WebLogic](#) and [File System](#) operations. It can also be used to perform user-specific naming operations. If the URL command line option does not contain a *supported* protocol such as LDAP, FILE, RMI, IIOP, DNS, or WebLogic it is assumed that the adapter is performing a generic naming operation.

All protocols are processed by the JNDI Adapter, however, if an *unsupported* protocol is used and the adapter does not recognize it, an exception is thrown and the adapter fails with the following message:

```
"WARNING: unsupported protocol " + protocol_name
```

The use of schemas is required in add, modify and delete operations. Schemas can be used in lookup operations when the adapter is used in an output card (a three (3) parameter `GET`). If lookup parameters are supplied in both a schema and the command line's URL parameter, the parameters in the schema are used and the corresponding URL parameters are ignored.

JNDI service providers

This section provides examples of how the JNDI Adapter handles service provider scenarios.

- [LDAP](#)
- [COS naming](#)
- [RMI](#)

- [DNS](#)
 - [File system](#)
 - [WebLogic](#)
-

LDAP

The JNDI Adapter is designed to mimic the functionality of the existing LDAP adapter. The advantage of the JNDI Adapter is that it is Java-based, and therefore capable of running on UNIX platforms as well as Windows platforms.

The adapter treats LDAP as a directory implementation.

- [Operations supported](#)
 - [Command line options supported](#)
 - [URL format](#)
 - [Initial Context factory](#)
 - [LDAP schema](#)
-

Operations supported

The **LDAP** provider supports the following operations:

Lookup	Add	Delete	Modify
✓	✓	✓	✓

Command line options supported

The following command line options are supported for LDAP connections:

Command Option	Lookup	Add	Modify	Delete
AM	Optional	Optional	Optional	Optional
ICTXF	Optional	Optional	Optional	Optional
ICTXFURL	Required	Required	Required	Required
MAXHITS	Optional			
TIMEOUT	Optional			
TRACE	Optional	Optional	Optional	Optional

URL format

`LDAP://[user:pass@]host[:port]/[rdn[?attr[?scope[?filter]]]]`

Option	Description
<i>user</i>	Specify the user name to connect to the LDAP server.
<i>pass</i>	Specify the password that authenticates the user name.
<i>host</i>	Specify the LDAP server name or address. The default name is localhost .
<i>port</i>	Specify the LDAP server port number. The default port name is 389 .
<i>rdn</i>	Specify the LDAP distinguished names. <i>rdn</i> is used during LDAP search operations. For add, replace, and modify operations it is optional. You can find examples of distinguished name abbreviations and formats in RFC 1779. Examples of distinguished name abbreviations include CN (common name), O (organization), and C (country). The following examples are from the RFC: CN=Christian Huitema, O=INRIA, C=FR CN=Christian Huitema; O=INRIA; C=FR CN=James Hacker L=Basingstoke O=Widget Inc C=GB OU=Sales + CN=J. Smith, O=Widget Inc., C=US CN=L. Eagle, O="Sue, Grabbit and Runn", C=GB CN=L. Eagle, O=Sue, Grabbit and Runn, C=GB
<i>attr</i>	

Specify the search attributes. *attr* is used during LDAP search operations. For add, replace, and modify operations, it is optional. This might be a single value or a comma-separated list.

scope

Specify the scope of the search. The default scope is **base**. Use this option only during an LDAP search operation. It is only valid when it is used in the [GET...Source](#) [...Command setting](#).

The **scope** options are:

- **base** - indicates a base object search
- **one** - indicates a single-level search
- **sub** - indicates a subtree search

filter

Specify the search filter as defined in RFC 1558. For example:
(!(cn=Tim Howes))

returns any common name except **Tim Howes**.

Initial Context factory

If no Initial Context factory is specified on the command line, the following service provider is used:

com.sun.jndi.ldap.LdapCtxFactory

LDAP schema

When used as an input card, the **Results LDAP** group should be selected. For all operations, the **Results LDAP** group should be selected.

The meaning of each node is described below:

Name	Description
Field/Data	A single attribute value
Field/Name	A single attribute name
Field/Operation	Used to instruct the adapter to perform a modify, add or delete operation. Valid values are add , modify , delete and replace . Replace and modify are identical. If the adapter is being used in a PUT rule and no operation is specified, modify is assumed.
Field/Operation/U	During a modify operation on an entry, its individual attributes can be added, modified or deleted. The <i>U</i> node applies to specific attributes. Valid values are add , modify , delete and replace . Replace and modify are identical. If left blank, modify is assumed.
Field/RDN	Overrides the RDN (<i>ldap_DN</i>) on the URL
Group/Attribute/Attribute	Corresponds to the <i>attr</i> parameter on the URL
Group/Attribute/ObjectClass	Represents the LDAP special attribute <i>objectclass</i>
Group/Attribute/ObjectClassU	Represents the LDAP special attribute <i>objectclass</i> in a modify request
Group/Entry	Represents a single directory entry
Group/Entries	Represents a collection of directory entries
Group/Value/Attribute	Contains the value or reference id of one attribute
Group/Value/ObjectClass	Contains the value or reference id of the LDAP specific <i>objectclass</i> attribute
Results	Represents the output from a lookup operation.

COS naming

The Common Object Services (COS) Name Server is the name server for storing Common Object Request Broker Architecture (CORBA) object references. It can be accessed from CORBA applications using the COS Naming package (*org.omg.CORBA.COSNaming*).

The JNDI/IOP service provider implements the **javax.naming.Context** interface on top of the COS Naming package in order to allow applications to use JNDI to access the COS Name Server.

The adapter treats COS Naming as a naming implementation.

- [**Operations supported**](#)
- [**Command line options supported**](#)
- [**URL format**](#)
- [**Initial Context factory**](#)

- [Generic naming type tree \(COS Naming\)](#)

Operations supported

The **COS Naming** provider supports the following operations:

Lookup	Add	Delete	Modify
✓	✓	✓	✓

Command line options supported

The following command line options are supported:

	Lookup	Add	Modify	Delete
TRACE	Optional	Optional	Optional	Optional
ICTXF	Optional	Optional	Optional	Optional
ICTXFURL	Required	Required	Required	Required
MAXHITS				
TIMEOUT				

URL format

`IIOP://[host[:port]]/[cosnaming_name] [?rdn]`

Option	Description
host	Specify the CORBA Naming Server name or address. The default name is localhost .
port	Specify the LDAP CORBA Naming Server port number. The default port is 900 .
cosnaming_name	Specify the root naming context (if any).
rdn	The name of the object to lookup, add, delete or modify

Initial Context factory

If no Initial Context factory is specified on the command line, the following service provider is used:

`com.sun.jndi.cosnaming.CNCtxFactory`

Generic naming type tree (COS Naming)

The COS Naming implementation uses the generic **Naming Repository** type tree.

When used as an input card the **Results JNDI** group should be selected. For all operations, the **Results JNDI** group should be selected.

The meaning of each node is described below:

Name	Description
Field/Data	Stores arbitrary input or output data
Field/Name	A single entry name
Field/Operation	Used to instruct the adapter to perform a modify, add or delete operation. Valid values are add , modify , delete and replace . Replace and modify are identical. If the adapter is being used in a PUT rule and no operation is specified, modify is assumed.
Group/Entry	Represents a single directory entry
Group/Entries	Represents a collection of directory entries
Group/Value	Contains the value or reference id of one entry
Results	Represents the output from a lookup operation

RMI

The RMI registry service provider allows JNDI applications to access remote objects registered with the RMI registry. Given the location of a registry, the provider will create a naming context with bindings for the objects registered there.

With this service provider installed, JNDI subsumes the functionality of the **java.rmi.Naming** class.

The adapter treats RMI as a naming implementation.

- [Operations supported](#)
- [Command line options supported](#)
- [URL format](#)
- [Initial Context factory](#)
- [Generic naming type tree \(RMI\)](#)

Operations supported

The **RMI** provider supports the following operations:

Lookup	Add	Delete	Modify
✓	✓	✓	✓

Command line options supported

The following command line options are supported:

	Lookup	Add	Modify	Delete
TRACE	Optional	Optional	Optional	Optional
ICTXF	Optional	Optional	Optional	Optional
ICTXFURL	Required	Required	Required	Required
MAXHITS				
TIMEOUT				

URL format

`rmi://[host[:port]][/[object]][?rdn]`

Option

Description

host

Specify the RMI registry name or address. The default name is **localhost**.

port

Specify the RMI registry port number. The default port is **1099**.

object

If the object name is absent, then the URL names the registry at the given host and port. Otherwise, it names the remote object registered at that registry under the name provided.

rdn

The name of the object to lookup, add, delete or modify

Initial Context factory

If no Initial Context factory is specified on the command line the following service provider is used:

`com.sun.jndi.rmi.registry.RegistryContextFactory`

Generic naming type tree (RMI)

The RMI implementation uses the generic **Naming Repository** type tree.

When used as an input card, the **Results JNDI** group should be selected. For all operations, the **Results JNDI** group should be selected.

The meaning of each node is described below:

Name

Description
Field/Data Stores arbitrary input or output data
Field/Name A single entry name
Field/Operation Used to instruct the adapter to perform a modify, add or delete operation. Valid values are add , modify , delete and replace . Replace and modify are identical. If the adapter is being used in a PUT rule and no operation is specified, modify is assumed.
Group/Entry Represents a single directory entry
Group/Entries Represents a collection of entries
Group/Value Contains the value or reference id of one entry
Results Represents the output from a lookup operation

DNS

The DNS service provider allows JNDI applications to access information stored in the Internet Domain Name System. The provider presents the DNS namespace as a tree of JNDI directory contexts, and DNS resource records as JNDI attributes.

The adapter treats DNS as a directory implementation.

- [Operations supported](#)
- [Command line options supported](#)
- [URL format](#)
- [Initial Context factory](#)
- [DNS Type Tree](#)

Operations supported

The **DNS** provider supports the following operations:

Lookup	Add	Delete	Modify
✓			

Command line options supported

The following command line options are supported:

	Lookup	Add	Modify	Delete
TRACE	Optional			
ICTXF	Optional			
ICTXFURL	Required			
MAXHITS				
TIMEOUT				

URL format

`dns://[host[:port]][/domain][[?rdn]?attr]`

Option

Description
<i>host</i> Specify the RMI registry name or address. The default name is localhost .
<i>port</i> Specify the RMI registry port number. The default port is 53 .
<i>domain</i> The domain is the DNS domain name of the context, and is not necessarily related to the domain of the server; it defaults to ". ", the root domain.
<i>rdn</i> The IP host name of the object to lookup
<i>attr</i> DNS resource record class and type names are mapped onto JNDI attribute identifiers. If a record is in the internet class, the corresponding attribute ID is simply the record's type name. If the type is an unsupported one, its integer value is used instead. If the record is not in the internet class, the class name or integer class value is pre-pended to the attribute ID and separated by a space.

For example, the attribute identifier "AAAA" represents an IPv6 address record, and the attribute identifier "HS 97" represents a resource record of type 97 in the **Hesiod** class.

Superclass attribute identifiers are also defined. These can be useful when querying records using the `DirContext.getAttributes()` method. If an attribute name has "*" in place of a type name or class name, it represents records of any type or class. For example, the attribute identifier "IN *" might be passed to the `getAttributes()` method to find all **internet** class records.

The default attribute identifier "* *" represents records of any class or type.

Attribute identifiers are case-insensitive.

Initial Context factory

If no Initial Context factory is specified on the command line, the following service provider is used:

`com.sun.jndi.dns.DnsContextFactory`

DNS Type Tree

The DNS implementation uses the following type tree.

When used as an input card, the **Results DNS** group should be selected. For all operations, the **Results DNS** group should be selected.

The meaning of each node is described below:

Name	Description
Field/Data	A single attribute value
Field/Name	A single attribute name
Field/RDN	A single entry name
Group/Attribute	A single attribute
Group/Entry	Represents a single entry
Group/Entries	Represents a collection of entries
Group/Value	Represents a collection of attributes
Results	Represents the output from a lookup operation

File system

JNDI offers a universal naming API and a standardized way to store objects in a namespace. The file system service provider supports these features for a local or networked file system. Its purpose is two-fold. It allows clients to:

- traverse the file system namespace using JNDI's context interface.
- store Java™ objects in the file system.

These two separate functions are handled by two different providers, that is, classes. The first class handles accessing the file system and can be used independently of the second. The second class handles storage of Java objects and is used with the first. It is a subclass of the first.

The adapter treats the **File** provider as a naming implementation.

- [**Operations supported**](#)
- [**Command line options supported**](#)
- [**URL format**](#)
- [**Initial Context factory**](#)
- [**Generic naming schema \(file system\)**](#)

Operations supported

The **File** provider supports the following operations:

Lookup	Add	Delete	Modify
✓	✓	✓	✓

Command line options supported

The following command line options are supported:

	Lookup	Add	Modify	Delete
TRACE	Optional	Optional	Optional	Optional
ICTXF	Optional	Optional	Optional	Optional
ICTXFURL	Required	Required	Required	Required
MAXHITS	Optional			
TIMEOUT				

URL format

`file:///qualified_file_name[?rdn]`

Option

Description

`qualified_file_name`

Specifies the file to be used as the root context. It must be a file URL representing a directory in the file system.

For example:

`file:///home/kafka`

This URL causes the provider to use the directory named `kafka` in the `home` directory of the root directory as the base context. Performing a `list()` on the initial context would be equivalent to typing the UNIX command `ls` in the `/home/kafka` directory. If this property is not set, it defaults to the root of the local file system.

`rdn`

The name of the file or object to lookup, add, delete or modify.

Initial Context factory

If no Initial Context Factory is specified on the command line, the following service provider is used:

`com.sun.jndi.fscontext.RefFSContextFactory`

It is expected that users will typically specify a different Initial Context Factory (`com.sun.jndi.fscontext.FSContextFactory`) which performs slightly differently. The default provider stores Objects on the file system. The optional provider provides access to the file system itself.

Generic naming schema (file system)

The **File** implementation uses the generic **Naming Repository** schema.

When used as an input card, the **Results JNDI** group should be selected. For all operations, the **Results JNDI** group should be selected.

The meaning of each node is described below:

Name

Description

Field/Data

Stores arbitrary input or output data

Field/Name

A single entry name

Field/Operation

Used to instruct the adapter to perform a modify, add or delete operation. Valid values are **add**, **modify**, **delete** and **replace**. **Replace** and **modify** are identical. If the adapter is being used in a `PUT` rule and no operation is specified, **modify** is assumed.

Group/Entry

Represents a single directory entry

Group/Entries

Represents a collection of entries

Group/Value

Contains the value or reference id of one entry

Results

Represents the output from a lookup operation

WebLogic

The WebLogic service provider allows JNDI applications to access remote objects registered with the WebLogic registry.

The adapter treats WebLogic as a naming implementation.

- [Operations supported](#)
 - [Command line options supported](#)
 - [URL format](#)
 - [Initial Context factory](#)
 - [Generic naming type tree \(WebLogic\)](#)
-

Operations supported

The **WebLogic** provider supports the following operations:

Lookup	Add	Delete	Modify
✓	✓	✓	✓

Command line options supported

The following command line options are supported:

	Lookup	Add	Modify	Delete
TRACE	Optional	Optional	Optional	Optional
ICTXF	Optional	Optional	Optional	Optional
ICTXFURL	Required	Required	Required	Required
MAXHITS				
TIMEOUT				

URL format

`t3://[host[:port]][/[object]][?rdn]`

Option	Description
<code>host</code>	Specify the WebLogic registry name or address. The default name is localhost .
<code>port</code>	Specify the WebLogic registry port number. The default port is 1099 .
<code>object</code>	If the object name is absent, then the URL names the registry at the given host and port. Otherwise, it names the remote object registered at that registry under the name provided.
<code>rdn</code>	The name of the object to lookup, add, delete or modify

Initial Context factory

If no Initial Context factory is specified on the command line the following service provider is used:

`weblogic.jndi.WLInitialContextFactory`

Generic naming type tree (WebLogic)

The WebLogic implementation uses the generic **Naming Repository** type tree.

When used as an input card, the **Results JNDI** group should be selected. For all operations, the **Results JNDI** group should be selected.

The meaning of each node is described below:

Name	Description
Field/Data	Stores arbitrary input or output data
Field/Name	A single entry name
Field/Operation	Used to instruct the adapter to perform a modify, add or delete operation. Valid values are add , modify , delete and replace . Replace and modify are identical. If the adapter is being used in a PUT rule and no operation is specified, modify is assumed.
Group/Entry	

	Represents a single directory entry
Group/Entries	Represents a collection of entries
Group/Value	Contains the value or reference id of one entry
Results	Represents the output from a lookup operation

Interoperability between JNDI adapter and other adapters

The Object Pool used by the JNDI Adapter is shared by some other adapters, notably the Java™ Class and JMS adapters. Objects created in one adapter can be passed by reference to another adapter.

The Object Pool is also useful for creating local Objects, such as Referenceable Objects. A Java class can have a method that returns an Object which is a Referenceable Object. This locally generated Object can be stored, for example, in an RMI registry by passing its reference id to the JNDI Adapters add or modify operation.

Objects retrieved from JNDI operations and stored in the pool usually need to be narrowed before they can be used outside of JNDI. Narrowing objects is beyond the scope of the JNDI Adapter since it might require external third-party software or services. For example, to narrow the result of a COS Naming operation, you need a fully initialized ORB.

Adapters that aim to share Objects with the JNDI Adapter should take this into account.

Binary data

The JNDI Adapter will not process any text/binary conversion. If binary data needs to be passed from a map to a JNDI service provider, it can be done in any supported way. For example, using a sequence of octets, or using a Base64-encoded string.

Example files

You can find an example of this adapter in the `tx_install_dir\examples\adapters` directory.

Memory Link Adapter

The Memory Link adapter is only used by the Launcher, and provides additional workflow support for Launcher watches to be triggered without traditional transformation adapters (such as file, MQ, or database adapters) and therefore does not add any file system, messaging, or database overhead. The watches can be triggered based on the outcome of other watches, and can also be used in **PUT** rules to trigger other watches. The Memory Link adapter has the ability to run watches inline, when the transactional handling of a watch might depend on another watch that is triggered by a Memory Link event.

Overview

Table 1. Memory Link adapter overview. The following table provides an overview of the Memory Link adapter:

Characteristic	Definition
Adapter Name	Memory Link
Description	This adapter provides additional workflow support and allows Launcher watches to be triggered without traditional adapters (such as file, MQ, or database adapters). The watches can be triggered based on the outcome of other watches, and can also be used in PUT rules to trigger other watches.
Business usage	The Memory Link adapter can queue a memory link trigger immediately during a PUT rule in a map, using the -NOW option. Additionally, you can configure input and output cards of a Launcher watch to use the Memory Link adapter. For output, you can use the Memory Link adapter in target cards, PUT rules, RUN map target cards, and RUN map PUT rules. Memory link outputs can trigger multiple watches if the watches are configured with the same memory link name source event.
Preconfigured?	The the adapter is installed with Transformation Extender. However, you add information to the config.yaml file before using the Memory Link adapter.
Requires third-party files?	No. All required components are shipped with the product.
Platform availability	<ul style="list-style-type: none"> • Windows • Linux®
Application requirements	This adapter requires Transformation Extender version 9.0.0.1 or later.
Invocation	This adapter can be invoked via the command line.
Returned status values	If a map fails, the failure is reported to the top watch map, which fails with the new error code 79 – Linked map failure.

Characteristic	Definition
Restrictions	<ul style="list-style-type: none"> You must define a memory link name as a single string for input cards, output cards, and PUT rules. Wildcard characters are not allowed in the memory link name. Valid characters for a memory link name are: <ul style="list-style-type: none"> o 0 - 9 o a - z o A - Z o dot (.) o dash (-) o underscore (_) The maximum length is 64 bytes, and another 63 bytes are reserved for wildcard and index values. An input memory link must be a source event and contain only the valid memory link name.
Testing considerations	<ul style="list-style-type: none"> Triggered maps must run sequentially inline. Maps should be queued to run concurrently if MaxThreads allow. Maps should be queued to run concurrently when using the PUT rule.

Memory Link configuration in the config.yaml file

The following section in the config.yaml file controls the Memory Link adapter.

```
runtime:
  Memory Link:
    # How many simultaneous active watches that write to memory link. Consider Initialize Pending maps that are
    # queued up to trigger along with InitPendingHigh setting in Launcher section. Default is 2000.
    ActiveTableSize: 2000
    # How many memory link instances per an active watch (each output card / PUT instance). Default is 5.
    UniqueTableSize: 5
```

The use of the **ActiveTableSize** and **UniqueTableSize** properties depends on the workflow and map design, but they increase automatically if needed.

ActiveTableSize

The number of Memory Link source events that are queued to run, including Initialize Pending maps.

UniqueTableSize

The number of unique Memory Link targets in a single Transformation Extender map instance.

PUT rule command line options

Use the following command line options when using the **PUT** rule with this adapter:

-SCOPE

The scope of the **PUT** command. Valid values are MAP and CARD.

-NOW

Trigger the **PUT** rule immediately.

MIME Adapter

This documentation introduces the Multipurpose Internet Mail Extensions (MIME) Adapter. You can use the adapter with a Command Server, Launcher, Software Development Kit, or map in a map rule.

- [MIME Adapter overview](#)
- [System requirements](#)
- [Command alias](#)
- [MIME commands](#)
- [Syntax summary](#)
- [Using the adapter](#)

MIME Adapter overview

Use the MIME Adapter to package, on output, and decode, on input, generic MIME messages.

The adapter does not handle the transporting of messages. Typically, SMTP is the transport of choice, but HTTP is another alternative.

The MIME Adapter is an encode and decode adapter. That is, it receives a stream of data, performs encoding or decoding and returns a stream of data. It is designed to be used with other adapters to transport the data.

Typical uses for this adapter include:

- **RosettaNet** - Full MIME support, including nested multipart messages, encryption and enveloping. The MIME Adapter can either be chained with a transport adapter such as HTTP or invoked through a **GET** function.
- **Multiple attachments for e-mail** - The MIME Adapter, when chained with the E-mail adapter provides a means to support multiple attachments.
- General MIME encoding and decoding for use with other transports such as HTTP.
- Generating unique IDs, using the **-GENID** command, for content IDs and other purposes.
- Calculating message digests, using the **-DIGEST** command.

System requirements

The minimum system requirements and operating system requirements for the MIME Adapter are detailed in the release notes. It is assumed that a Command Server has already been installed on the computer where the adapter is to be installed for run-time purposes.

Command alias

Adapter commands can be specified by using a command string on the command line or by creating a command file that contains adapter commands. The execution command syntax is:

```
-IM[alias] card_num  
-OM[alias] card_num
```

where **-IM** is the Input Source Override execution command and **-OM** is the Output Target Override execution command, *alias* is the adapter alias, and *card_num* is the number of the input or output card. The following table shows the adapter alias and its execution command.

Adapter	Alias	As Input	As Output
MIME	MIME	-IMMIMEcard_num	-OMMIMEcard_num

MIME commands

This documentation describes the functions and use of the MIME Adapter commands and their options.

- [List of commands](#)

List of commands

The following table lists valid commands for the MIME Adapter, the command syntax, and whether the command is supported (✓) for use with data sources, targets, or both.

The MIME Adapter can only be used in an input card if it is chained with a transport adapter.

Command	Syntax	Source	Target
Decode (-DECODE)	-DECODE	✓	
Encode (-ENCODE)	-ENCODE		✓
Generate ID (-GENID)	-GENID [filename]	✓	
Message Digest (-DIGEST)	-DIGEST [OPGP SMIME] [MD5 SHA1]	✓	
Session (-SESSION)	-SESSION alias(commands)	✓	✓
Trace (-T)	-T[E][+][filename]	✓	✓
Transport (-TRANSPORT)	-TRANSPORT 'adapter_name (adapter_command_line)'	✓	✓

- [Decode \(-DECODE\)](#)
- [Encode \(-ENCODE\)](#)
- [Generate ID \(-GENID\)](#)
- [Message Digest \(-DIGEST\)](#)
- [Session \(-SESSION\)](#)
- [Trace \(-T\)](#)
- [Transport \(-TRANSPORT\)](#)

Decode (-DECODE)

Use the Decode adapter command (-DECODE) to instruct the adapter to decode the MIME data into XML format. This is performed implicitly if the MIME Adapter is chained with a transport adapter in an input card.

-DECODE

For example, to decode MIME data received over HTTP:

```
GET ("MIME", "-DECODE -TRANSPORT 'HTTP (-URL http://myurl/ )'"")
```

The -DECODE option is optional since it is implied by the invocation of the MIME Adapter in an input card or GET function.

For more information on encoding and decoding data, see the Encoding and Decoding Data section in the *Resource Adapters documentation*.

Encode (-ENCODE)

Use the Encode adapter command (-ENCODE) to instruct the adapter to encode the XML data into MIME format. This is performed implicitly if the MIME Adapter is chained with a transport adapter in an output card.

-ENCODE

For example, to encode MIME data before transporting over HTTP:

```
PUT("MIME", "-ENCODE -TRANSPORT 'HTTP (-URL http://myurl/)'"
  MimeRequest )
```

The -ENCODE option is optional since it is implied by the invocation of the MIME Adapter in an output card or PUT function.

For more information on encoding and decoding data, see the Encoding and Decoding Data section in the *Resource Adapters documentation*.

Generate ID (-GENID)

Use the Generate ID adapter command (-GENID) to return a *unique* ID. Its primary purpose being for Content-ID attributes.

The command uses a file to track the current ID, and serializes access to this file. The file contains a single number which is incremented with each call to the adapter with the -GENID command.

The -GENID cannot be used with any other adapter commands.

This is an optional command.

-GENID [filename]

where *filename* is the file used to track the current ID. The *filename* is optional and if not specified, defaults to *install_dir\genid.dat*.

For example, to generate a unique ID for a Content-ID:

```
ContentID = "ContextSensitivestem" + GET("MIME", "-GENID"
  c:\genid.dat")
```

Message Digest (-DIGEST)

Use the Message Digest adapter command (-DIGEST) to obtain the message digest for the specified message. The MD5 and SHA1 message digests can be calculated. The syntax is:

-DIGEST [OPGP|SMIME] [MD5|SHA1]

where SMIME is the default adapter to invoke for backwards compatibility, and MD5 is the default algorithm.

This is an optional command.

If this command is specified in an input card, the message digest will be calculated and returned back to the map. The rules for specifying the algorithm parameter are:

- If the original message is signed and the algorithm is not specified, the algorithm that was used in signing the message will be used to calculate the digest.
- If the original message is not signed and the algorithm is not specified, MD5 shall be used.
- If the algorithm is specified, this algorithm shall be used regardless of the algorithm used to sign the message.

Session (-SESSION)

Use the Session adapter command (-SESSION) to specify encryption session commands. These commands are appended to any encryption commands provided in the data.

This is an optional command.

-SESSION alias (commands)

where *alias* is the alias of the encoder to invoke to perform encryption or other operations, and *commands* contains the commands to be passed to that adapter.

For example:

```
-SESSION SMIME (-USER marcel -PASSWORD **** )
```

It is possible to have multiple -SESSION commands on the command line, if multiple encoders need to be invoked.

For example:

```
-SESSION SMIME(-USER marcel) -SESSION OPGP (-USER marcel)
```

In an output card, if the mapped data contains parameters for these encoders, the values of the mapped data override those of the command line.

The commands specified in the -SESSION command are appended to those specified in the data.

- [Example](#)

Example

If the data passed to the MIME Adapter contains the following message part:

```
<MIME-Message-Part>
<Header>
Content-Type: application/pkcs7-mime
Content-Transfer-Encoding: base64
Content-Disposition: attachment; filename=something.p7m
Encryption: SMIME (-ENCODE -ACTION ENCRYPT)
</Header>
```

```
</MIME-Message-Part>
```

and the MIME Adapter is called with the following command line:

```
-SESSION SMIME(-USER marcel -PASSWORD ****)
```

the command line is appended to any occurrence of *Encryption: SMIME* in the MIME data, and the command sent to the SMIME encoder is:

```
-ENCODE -ACTION ENCRYPT -USER marcel -PASSWORD ****
```

Trace (-T)

Use the Trace adapter command (-T) to produce a diagnostics file that contains detailed information about MIME Adapter activity.

The default filename is **m4mime.mtr** and it is created in the map directory unless otherwise specified.

```
-T[E][+] [filename]
```

Option

Description
E
Produce a trace file containing only the adapter errors that occurred during map execution.
+
Append trace information to the existing trace file.
filename
Creates a trace file with the specified name in the specified directory

You can override the adapter command line trace options dynamically using the Management Console. See "Dynamic Adapter Tracing" in the *Launcher* documentation for detailed information.

Transport (-TRANSPORT)

Use the Transport adapter command (-TRANSPORT) to specify which transport adapter to use and the command line for the transport adapter.

The syntax of the command must include the single quotes and parentheses as shown below.

```
-TRANSPORT 'adapter_name ( adapter_command_line )'
```

Option

Description
adapter_name
Specify the transport adapter to use.
adapter_command_line
Specify the command line for the transport adapter. You must enclose this option within parentheses () .

For example, to specify the **HTTP** adapter as the transport adapter, and to send the MIME request to the **www.myws.com/DoWS** URL, enter the following command:

```
-TRANSPORT 'http (-URL http://www.myws.com/DoWS -T)'
```

Syntax summary

This documentation discusses the MIME Adapter syntax summary and how it is used.

- [Data sources](#)
 - [Data targets](#)
 - [Example](#)
-

Data sources

The following is the command syntax of the MIME Adapter commands used for data sources:

```
-SESSION alias ( commands )
[-TRANSPORT 'adapter_name (adapter_command_line)']
[-DECODE]
[-DIGEST [OPGP|SMIME] [MD5|SHA1]]
[-GENID [filename]]
[-T[E] [+]] [filename]
```

If the -GENID command is specified, no other commands can be used.

Data targets

The following is the command syntax of the MIME Adapter commands used for data targets:

```
-SESSION alias ( commands )
[-TRANSPORT 'adapter_name (adapter_command_line)']
[-ENCODE]
[-T[E] [+]] [filename]
```

See the Encode/Decode Scenarios section in the *Resource Adapters documentation* for more information about the -DECODE, -ENCODE and -TRANSPORT commands.

Example

For the GET->Source setting in an input card, select MIME. In the GET->Source->Command field, enter:

```
-DECODE -SESSION SMIME(-U 1121 -PW cat12 -UE 1121@foo.com) -T+
-TRANSPORT 'HTTP (-URL http://myurl.com -METHOD GET)'
```

This command specifies that:

- the input stream is MIME and therefore should be decoded (-DECODE)
 - if the data contains SMIME encrypted or signed data, the session defined by the -SESSION command should be used
 - the adapter trace should be appended to the **m4mime.mtr** file
 - the data passed to the MIME Adapter originates from the transport adapter (HTTP) by getting the document from the URL *http://myurl.com*. See the Encoding and Decoding Data section in the *Resource Adapters documentation* for more information on the -TRANSPORT command.
- The MIME Adapter can only be used in an input card if it is chained with a transport adapter.
-

Using the adapter

Use the MIME Adapter to package, on output, and decode, on input, generic MIME messages. The adapter does not handle the transporting of messages. A transport adapter, such as HTTP, performs this action.

The MIME Adapter uses S/MIME, OpenPGP or another encoder to encode parts of the message and to handle signatures. Hence, the adapter directly invokes the S/MIME, OpenPGP or another encoder for this purpose.

The MIME Adapter gets its instructions for encryption, decryption, signing and verifying from within the header of the message parts, and additionally from the -SESSION command line.

The data mapped to an output card includes commands for encryption in the <Header> element of the message part. The MIME Adapter invokes the specified encoder and passes the specified commands to that encoder. Typically, this encoder is either the S/MIME or OpenPGP adapter.

- [MIME structure](#)
 - [Representing MIME data in type trees](#)
 - [Using the type tree example](#)
-

MIME structure

The following documentation describes the format of a MIME message. The specification RFC 2046 should be consulted for full description of the MIME format.

- [MIME header fields](#)

- [Multipart messages](#)
-

MIME header fields

This section defines the fields used in MIME headers. MIME headers consist of the following fields:

- MIME Version Header field
 - Content-Type Header field
 - Content-Transfer-Encoding Header field
 - Content-ID Header field
 - Content-Description Header field
 - [MIME version header field](#)
 - [Content-type header field](#)
 - [Content-transfer-encoding header field](#)
 - [Content-ID header field](#)
 - [Content-description header field](#)
-

MIME version header field

The header field must be included in the MIME message:

```
MIME-Version: 1.0<CR><LF>
```

A comment might be included in the header field.

For example:

```
MIME-Version: (produced by John Doe) 1.0<CR><LF>
```

This comment can appear anywhere.

Content-type header field

This header field contains media-type and subtype values. The media type is the general type of the data, and the subtype is a specific format.

Media type can have the following values:

- text
- image
- audio
- video
- application
- message
- multipart
- x-* - where * can be any token

The values are *not* case sensitive.

The Content-Type header field can be followed by any number of attributes.

For example:

```
Content-Type: text/plain; charset=us-ascii <CR><LF>
```

The value of the attributes can be enclosed in double quotes.

If the Content-Type is not specified it defaults to:

```
text/plain, charset=us-ascii
```

Content-transfer-encoding header field

This header field defines the content encoding of the message.

Valid values include:

- 7bit
- 8bit
- binary
- quoted-printable

- base64
- x-* - where * represents any token

If not specified, the Content-Transfer-Encoding defaults to 7bit.

7bit, 8bit and binary indicate that NO encoding has been applied. "x-token" permits additional encoding schemes, but its use is discouraged by the standard (RFC 2045).

If this field is used in the message header, it applies to the entire body of the message. If it appears in an entity header, then it applies only to that entity. If an entity is of type *multipart* then the encoding can only have values of 7bit, 8bit or binary.

Content-ID header field

This header field uniquely identifies MIME entities in several contexts.

For example:

```
Content-ID: 123132
```

Content-description header field

This header field permits descriptive text to be included in the header..

For example:

```
Content-Description: this is my message.
```

Multipart messages

Of special interest are messages with a media-type *multipart*. These are messages that contain one or more different sets of data in a single body.

A multipart message consists of the following parts in the order listed here:

- **Header** - The Content-Type is *multipart*. A required parameter is *boundary* that defines the boundary parameter value. The boundary value can be enclosed in quotes. This is always good practice, but not required. It can not exceed 70 characters. The boundary value is used to delimit each part. By making this a parameter, an application can ensure that there is no chance that the boundary value appears in any of the entities of the message.
- **Preamble** - This is optional and can be ignored.
- **Boundary Delimiter** - This is two hyphens, the boundary parameter, optional whitespace and a CRLF terminator.
- **Message Part (Entity)** - The entity might include a header. If it does, then this header can only include fields that describe the content and which begin with "Content-". The header is followed by a blank line, and then the body.
- **Closing Delimiter** - This is two hyphens, the boundary parameter, two more hyphens, optional whitespace and a CRLF terminator.
- **Epilogue** - This is optional and can be ignored.

Here is a sample multipart message:

```
From: Nathaniel Borenstein <nsb@bellcore.com>
To: Ned Freed <ned@innosoft.com>
Date: Sun, 21 Mar 1993 23:56:48 -0800 (PST)
Subject: Sample message
MIME-Version: 1.0
Content-type: multipart/mixed; boundary="simple boundary"

This is the preamble. It is to be ignored, though it
is a handy place for composition agents to include an
Explanatory note to non-MIME conformant readers.

--simple boundary

This is implicitly typed plain US-ASCII text.
It does NOT end with a linebreak.
--simple boundary
Content-type: text/plain; charset=us-ascii

This is explicitly typed plain US-ASCII text.
It DOES end with a linebreak.

--simple boundary--

This is the epilogue. It is also to be ignored.
```

Representing MIME data in type trees

The MIME Adapter accepts a datastream and produces a corresponding MIME stream for an output card, it then parses a MIME stream and creates a decoded datastream for an input card.

This datastream could contain binary data that needs encoding, for example, using Base64 or Quoted-Printable encoding. In addition, parts or all of the stream might need to be secured, either through S/MIME or PGP encoding.

Therefore, the input stream needs to contain not only the data, but the processing instructions that determine what to do with the various message parts.

The format of this data is largely XML-based, however, since the adapter can represent binary data, it is not well-formed XML.

- [Message](#)
 - [Header](#)
 - [Message Parts](#)
-

Message

A typical message can include several encrypted and non-encrypted parts.

The top level message contains one or more parts. A message is represented in the stream in the following example.

```
<MIME-Message>
<Header>
...
</Header>
<Preamble>
</Preamble>
<MIME-Message-Part>
...
</MIME-Message-Part>
...
<MIME-Message-Part>
...
</MIME-Message-Part>
<Epilogue>
</Epilogue>
</MIME-Message>
```

The message always contains a `Header`. See "[Header](#)" for more information. But some of the fields of the header are optional. Following this is the `Preamble`, which can be omitted in its entirety. Following this are one or more `Message Parts`, and finally an optional `Epilogue`.

If the message is a simple single-part MIME message, then no `Preamble` or `Epilogue` is permitted, and the single message part cannot have a `Header`.

If a `Boundary` `Header` field is specified in a multipart message, then this is used as the `Boundary` value in the final MIME message.

If the message is a simple single-part MIME message, then no `Preamble` or `Epilogue` is permitted, and the single message part cannot have a `Header`. The data presented to the adapter for a single-part MIME message should have the following form:

```
<MIME-Message>
<Header>
...
</Header>
<Message-Part-Data>
This is my data
</Message-Part-Data>
</MIME-Message>
```

Notice that there are no `<MIME-Message-Part>...</MIME-Message-Part>` tags. The type tree `install_dir\examples\adapters\mime\mime.mtt` provided with the product can be used to create single-part data.

Header

The header can appear in the top-level message or the individual message parts. It is represented in the stream as follows:

```
<Header>
Content-Type: type; attr1="val1"; attr2="val2" ...
Content-Transfer-Encoding: base64 | quoted-printable
Encryption: SMIME(commands) | PGP(commands)
Content-ID: id
Content-Description: description
Extra-Header1: value1; attr1="val1"; attr2="val2" ...
Extra-Header2: value2; attr1="val1"; attr2="val2" ...
Boundary: OptionalTerminatingString
</Header>
```

The `<Header>` contains information about the content and the chosen encoding. The `Content-Type`, `Content-ID`, `Content-Description` and any additional header values, shown above as `Extra-Header1` and `Extra-Header2`, are copied without processing to the resultant MIME message.

Note that these can have any number of attributes. `Content-Transfer-Encoding` specifies an encoding scheme which is either Base64 or Quoted-Printable. The specified `Content-Transfer-Encoding` will be applied to the message data before any `Encryption` commands are applied to it.

The `Encryption` field instructs the MIME Adapter as to which adapter to invoke to perform data encryption. The encoding scheme might optionally be followed by parameters, (these are adapter commands that are passed directly to the adapter).

If the `Boundary` value is specified, it is added as an attribute of the `Content-Type` field in the resulting MIME message, and is used in the part delimiter of the multipart messages.

The adapter looks specifically for the following Content-Types:

- `application/pkcs7-mime` (encrypted or signed SMIME data)
- `multipart/signed` (a multipart message where the last part is a detached signature)
- `application/pkcs7-signature` (the message part is the detached signature)
- `application/pgp-encrypted` (PGP encrypted data)
- `application/pgp-signature` (PGP signed data)

It also looks for any Content-Type that begins with `multipart/` to determine if the resulting MIME message should have a multipart structure with boundary delimiters.

In the case that Content-Type is `application/pkcs7-mime`, the adapter checks the attributes of the Content-Type header field and looks for `smime-type=signed-data` or `smime-type=enveloped-data`. **Signed-data** indicates that the message has an encapsulated signature. **Enveloped-data** indicates that the message is encrypted and possibly signed. Otherwise, the Content-Type is not interpreted by the MIME Adapter.

Message Parts

Each part of the message is represented in the stream as follows:

```
<MIME-Message-Part>
<Header>
...
</Header>
<Message-Part-Data>
Size: 1234
(the data - can be binary or text)
OptionalTerminatingString
</Message-Part-Data>
</MIME-Message-Part>
```

The header is identical to the header used in the top-level message, but generally only uses the `Content-*` fields.

The data of the part is enclosed in the `<Message-Part-Data>` tags. This might begin with a size if the data is binary. The value of the size is given by a character integer value. The size is followed by a `CR/LF`. The data then follows. It can be terminated by a terminator that is specified in the header of the message part by the `Boundary` field. If this is not specified, then the terminator is `</Message-Part-Data>`, which is always present regardless of whether there is an additional terminator or not.

It is possible to nest message parts, which is required in contexts such as RosettaNet messages.

Using the type tree example

To access the adapter example files, navigate to `install_dir/examples/adapters`.

The data stream presented to the MIME Adapter to be encoded, or returned from the adapter when decoding MIME data, must comply to the type tree `mime.mtt` provided in the `examples` directory. The document element in this tree is the group `Message` which has the following components:

The `MessageHeader` group is an unordered collection of MIME header items. If the MIME message is a multi-part message then there will be one or more `MessagePart` groups. If the message is not multi-part then there will be no `MessagePart` groups, and the message data will be contained within the `MessageData Content` choice group.

A `MessagePart` consists of the following components:

The `Header` group is similar to the `MessageHeader` group but contains less components, since there are fewer relevant MIME header fields in an individual message part.

The `MessageData Content` is a choice group: the data contained within is either binary or text. If binary, the data is sized and the item `Sizeof` contains the size of the following binary item.

When the MIME Adapter decodes MIME data and returns it, the adapter fills in the `Sizeof` item with the correct size so that subsequent validation can determine the limits of the binary data. When using the `mime.mtt` type tree to build XML formatted input for the MIME Adapter, it is not necessary to map a value for the size, since the adapter will look for the `</Message-Part-Data>` delimiter to determine the end of the binary data.

The `NestedPart` group is used to build nested MIME data (such as is constructed by e-mail clients when messages are forwarded and the prior e-mail is attached).

The example maps in the `examples` directory show how this type tree can be used to build data to be encoded by the MIME Adapter and to decode MIME data.

MongoDB Adapter

Use the MongoDB adapter to create, read, update, and delete single or multiple JavaScript Object Notation (JSON) documents in a MongoDB database.

- [Type tree requirements](#)
To validate JSON documents fetched from a MongoDB collection, a map must use the MongoDB adapter and specify a JSON-based type tree on the input or output card.
- [Instance document formats](#)
A JSON instance document for a MongoDB collection can represent a single document in a collection or many documents in a collection. The MongoDB adapter supports both formats.
- [MongoDB adapter commands](#)
Use MongoDB adapter commands on input and output cards or GET and PUT functions to configure the adapter to create, read, update, and delete single or

multiple JSON documents in a MongoDB database.

Type tree requirements

To validate JSON documents fetched from a MongoDB collection, a map must use the MongoDB adapter and specify a JSON-based type tree on the input or output card.

To specify the type tree:

- Optimally, specify the file name of a native JSON document as the Type Tree on the input or output card. The native JSON parser in Design Studio uses the JSON document as the type tree during map compilation.
- Alternatively, you can use the JSON Importer in Design Studio to create a type tree (.mtt file) from a native JSON document, then specify the .mtt file as the Type Tree on the input or output card.

To create documents, you can use either:

- A JSON-based type tree.
- Any type that provides the entire JSON instance document.

MongoDB is a NoSQL database, so the documents in a MongoDB collection can have different fields and structures. Ensure that the JSON document you use as a type tree represents the documents in your collection.

Any document you insert into a MongoDB collection must be a JSON document.

Instance document formats

A JSON instance document for a MongoDB collection can represent a single document in a collection or many documents in a collection. The MongoDB adapter supports both formats.

Single document format

The format of an instance document (.json file) that represents a single document is similar to the following:

```
{  
    JSON-document-data  
}
```

For example:

```
{  
    Customer : Company X,  
    Location : Europe  
}
```

Multiple document format

It's common to have more than one document in a MongoDB collection. In this case, an instance document (.json file) includes more than one document in a JSON array. Each document in a JSON array is saved as separate document in the MongoDB collection. The format of the instance document is similar to the following:

```
[  
    {  
        JSON-document-data  
    },  
    {  
        JSON-document-data  
    }  
]
```

For example:

```
[  
    {  
        Customer : Company X,  
        Location : Europe  
    },  
    {  
        Customer : Company Y,  
        Location : North America  
    }  
]
```

Managing document formats

When the type tree of an input card represents single instance document, the **-SDF** command is required. The -SDF command indicates that the adapter is to send a non-array style, single JSON instance document to the map, even if the adapter fetches multiple documents. A single-document type tree cannot validate an array-style, multiple-document instance.

When the type tree of an output card represents single instance document, the -SDF command is required to ensure the adapter treats the instance as single document; otherwise, adapter errors occur when inserting documents into the MongoDB collection.

When you specify the -SDF command, the MongoDB adapter cannot transmit documents in array-style, multiple-document format.

By default, the MongoDB adapter processes instance documents as multiple-document arrays (multiple-document format). There is no corresponding command option.

When the type tree of an input card represents an array-style (multiple-document) instance document, the Native JSON parser can successfully process documents that are in either single-document format or multiple-document format. When the type tree of an output document represents an array-style instance document, the adapter can insert only documents that are in multiple-document format into a MongoDB collection.

MongoDB adapter commands

Use MongoDB adapter commands on input and output cards or GET and PUT functions to configure the adapter to create, read, update, and delete single or multiple JSON documents in a MongoDB database.

MongoDB adapter command alias

Use **MONGO** as the adapter command alias on input and output card overrides and in **GET** and **PUT** rules. For example:

Input source override execution command	-IAMONGO card_num
Output target override execution command	-OAMONGO card_num

Adapter commands

-HOST {hostname | MongoDB_URI}

-H {hostname | MongoDB_URI}

The host name or MongoDB URI (for example, `mongodb://localhost:27017/`) of the MongoDB server.

When you use the MongoDB URI, you can specify additional options for MongoDB client communication, such as enabling the client to use security based on certificates. Also, you can secure the user ID and password by using resource aliases for the **-USER** and **-PSWD** commands, and specify the `$user` and `$pswd` tokens on the URI. The user ID and password are dynamically resolved in the MongoDB URI during map execution.

This command is optional. If omitted, the default server is localhost.

-PORT port_num

-P port_num

The port number of the MongoDB server. This command is optional. If omitted, the default port number is 27017.

The **-PORT** command is ignored when you specify a MongoDB URI on the **-HOST** command.

-DATABASE db_name

-DB db_name

The name of the database on the MongoDB server.

-USER userID

-U userID

The user name or identifier that authenticates to the MongoDB server. Omit this keyword if user security is disabled on the server for connectivity.

-PASSWORD pwd

-PSWD pwd

The password of the user that authenticates to the MongoDB server. Omit this keyword if user security is disabled on the server for connectivity.

-COLLECTION collection_name

-C collection_name

The name of the MongoDB collection in the database to operate on.

-QUERY {"query_text" | file_URL}

-Q {"query_text" | file_URL}

The text of the JSON query that selects the documents to operate on, or the URL of a file that contains the query (for example, `file:///C:/myjsonquery.txt`). If the query contains spaces, enclose it in quotation marks. A relative URL is relative to the map directory.

-QTY

Specifies the number of documents that the adapter is to retrieve. In burst mode, the total number of documents that the adapter retrieves does not exceed the number specified by the **-QTY** command, although a single burst might return fewer documents. The **-QTY** command is optional. If you omit it, the default number of documents is 1.

-LSN

Specifies the amount of time, in seconds, that the adapter waits for a document to arrive. This command is optional. If you omit it, the adapter waits for an infinite amount of time. If a document does not arrive within the specified time frame:

- In Command Server mode, the adapter returns a warning.
- In Launcher mode, the Launcher continues to retry.

-NOW

Commits the transaction immediately instead of as defined by the Transaction Scope setting on the input or output card.

-SDF

Indicates that the JSON document is in single document format, rather than an array of documents. When you specify **-SDF**:

- A **GET** transaction returns only one document to the map, even when the adapter gets multiple documents.
- A **PUT** transaction writes only one document to the collection in the database.

-UPDATE

- Updates the documents that match the query with the data in the third argument of the **GET** function, then returns the documents to the map for further processing.

- Updates the documents that match the query with either the data in the output card or the data in the third argument of the **PUT** function, then adds the documents to the collection.

-DELETE

Deletes the documents that match the query from the collection.

-T [E | V] [+] *file_path*

The adapter trace level and full path to the adapter trace log.

-T

Log adapter informational messages.

-TE

Log only adapter errors during map execution.

-TV

Use verbose (debug) logging. The log file records all activity that occurs while the adapter is fetching or inserting documents.

+

Append the trace information to the existing log file. Omit this keyword to create a new log file.

MSMQ Adapter

You can use the MSMQ Adapter with a Command Server, Launcher, the Software Development Kit, or with a map in a map rule.

- [Overview](#)
- [MSMQ message content](#)
- [System and configuration requirements](#)
- [Command alias](#)
- [MSMQ commands](#)
- [Syntax summary](#)
- [MSMQ message header properties](#)
- [Troubleshooting](#)
- [Return codes and error messages](#)

Overview

The adapter for Microsoft Message Queue (MSMQ) servers integrates mapping technology with reliable data transactions provided by MSMQ servers. Business applications can reliably exchange data on distributed corporate networks. The resulting data conforms to the data format requirements of the target application.

For example:

- Use the Command Server to use MSMQ messages as input to a map and/or to build MSMQ messages as a result of data transformation rules within a map.
- Use the Launcher to execute systems of maps based upon the arrival of a message in an MSMQ server, or at the presence of specific message events or properties. Individual maps within the system then read the message content as input data.
- Use the Platform API to tightly integrate an application with source MSMQ messages or to create MSMQ messages.

MSMQ message content

The messages matching the criteria set by the correlation ID adapter command (**-CID**) or the Label adapter command (**-LABEL**), or the top-most messages on the queue are returned.

System and configuration requirements

In addition to any requirements in the release notes, the following are requirements to install and run the MSMQ Adapters:

The system executing maps must be a member of a configured MSMQ enterprise. The workstation upon which the MSMQ Adapters are installed must be an Intel or Intel-compatible platform using a supported Microsoft Windows operating system. See the system requirements.

Command alias

Adapter commands can be specified by using an execution command string on the command line or by creating a command file that contains adapter commands dictating the desired execution settings. For information about all of the options you can use with the execution commands for adapters or how to create a command file, see the *Execution Commands* documentation.

Use the Input Source Override - Message execution command (**-IM**) and the Output Target Override - Message execution command (**-OM**) with the appropriate MSMQ Adapter-specific alias. The execution command syntax is:

```
-IM[alias] card_num
-OM[alias] card_num
```

where **-IM** is the Input Source - Override execution command and **-OM** is the Output Target - Override execution command, **alias** is the adapter alias, and **card_num** is the number of the input or output card, respectively. The following table shows the adapter alias and execution commands.

Adapter	Alias	As Input	As Output
MSMQ	MSMQ	-IMMSMQcard_num	-OMMSMQcard_num

When using an adapter alias with the execution command, the MSMQ Adapter commands can be issued on the command line or in a command file. Use the adapter commands to specify adapter functions such as:

- specifying a particular message identifier
- allowing output data to be broken up into multiple messages
- retrieving a logical message from a source queue with a correlation identifier

For example, to override the adapter commands defined in output card 1, the command string for the adapter in the MSMQ environment might be:

```
-IMMSMQ1 '-QN .\topqueue -T'
```

This sample execution command string specifies a local queue named **topqueue** as the source for this card's data. The Trace command (**-T**) indicates that a trace file will be created to report adapter activity information, recording the events that occur while the adapter is retrieving this data.

- [GET > Source settings](#)
- [PUT > Target settings](#)

GET > Source settings

Use the Map Designer **GET** settings to configure options for input map cards using the MSMQ Adapters. Use the **GET > Source** setting to select the data source. For the **Source** setting, select MSMQ to use the MSMQ Adapter.

- [GET > Source > Command setting](#)
- [Source > Transaction > OnSuccess setting](#)
- [Source > Transaction > OnFailure Setting](#)
- [Source > Transaction > Scope setting](#)
- [Source > Transaction > Warnings setting](#)

GET > Source > Command setting

Use the **Command** setting to enter adapter commands. For command syntax, see [List of Commands](#).

Source > Transaction > OnSuccess setting

Use the **OnSuccess** setting with adapter sources to prevent deleting the source messages from the server from which the messages originate. The default value is **Keep**.

Value

Description

Keep

This keeps source messages on the server.

Delete

This deletes source messages following a successful map completion.

Source > Transaction > OnFailure Setting

Use the **OnFailure** setting with adapter sources to select the rollback or commit behavior if the map does not successfully complete. The default value is **Rollback**.

Value

Description

Rollback

If the map does not complete successfully, messages retrieved through the adapter are not deleted and the targets are not stored.

Commit

The adapter behaves as if the map was successful and the action specified by the **OnSuccess** setting (**Keep**) is taken.

Rollback is supported only when using a transactional queue.

Source > Transaction > Scope setting

Use the **Scope** setting to specify when to check for success (**OnSuccess**) or failure (**OnFailure**) at the completion of the processing of either a map, card, or burst. This is so that the rollback and retry actions can be performed as specified. The default value is **Map**.

Value

Description

Map	Check for success or failure at the completion of each map. If the map completes successfully, use the OnSuccess setting. If the map fails, use the OnFailure setting.
Burst	Check for success or failure at the completion of each burst. If the burst is successful, use the OnSuccess setting. If the burst fails, use the OnFailure setting.
Card	Check for success or failure at the completion of each card. If the card is successfully processed, use the OnSuccess setting. If the card fails, use the OnFailure setting.

Source > Transaction > Warnings setting

Use the **Warnings** setting to choose to fail the map or ignore warning conditions for the specific input map card. The default value is Ignore.

Value	Description
Ignore	Ignore warnings for this map card.
Fail	Fail the map upon receiving a warning for this map card.

PUT > Target settings

Use the Map Designer **PUT** settings to configure options for output map cards that use the MSMQ Adapter. Use the **PUT > Target** setting to select the data target. Select **MSMQ** to use the MSMQ Adapter.

- [PUT > Target > Command setting](#)
- [Target > Transaction > OnSuccess setting](#)
- [Target > Transaction > OnFailure setting](#)
- [Target > Transaction > Scope setting](#)
- [Target > Transaction > Warnings setting](#)

PUT > Target > Command setting

Use the **Command** setting to enter adapter commands. For command syntax, see [List of Commands](#).

Target > Transaction > OnSuccess setting

Use the **OnSuccess** setting with adapter targets to prevent a message from being created unnecessarily during the adapter process. The default value is Create.

Value	Description
Create	Upon successful completion of the burst, map, card, or rule determined by Scope settings, create the message.
CreateOnContent	Upon successful completion of the burst, map, card, or rule determined by Scope settings, create the message if the message body is not empty.

Target > Transaction > OnFailure setting

Use the **OnFailure** setting with adapter targets to select the rollback, which restore a previous version of the message, or commit behavior if the map does not complete successfully. The default value is Rollback.

Value	Description
Rollback	If the map does not complete successfully, the targets are not stored.
Commit	The adapter behaves as though the map was successful and the action specified by the OnSuccess setting (Create or CreateOnContent) is taken.

Target > Transaction > Scope setting

Use the **Scope** setting to specify when to check for success (**OnSuccess**) or failure (**OnFailure**) of a map, burst, or card in order to perform the rollback and retry options. The default value is Map.

Value	Description
Map	

Map

Check for success or failure at the completion of each map. If the map completes successfully, use the **OnSuccess** setting. If the map fails, use the **OnFailure** setting.

Burst

Check for success or failure at the completion of each burst. If the burst is successful, use the **OnSuccess** setting. If the burst fails, use the **OnFailure** setting.

Card

Check for success or failure at the completion of each card. If the card is processed successfully, use the **OnSuccess** setting. If the card fails, use the **OnFailure** setting.

Target > Transaction > Warnings setting

Use the **Warnings** setting to determine whether to fail the map or ignore warning conditions for the specific output map card. The default value is **Ignore**.

Value**Description****Ignore**

Ignore warnings for this map card.

Fail

Fail the map upon receiving a warning for this map card.

MSMQ commands

This section describes adapter commands used to configure and operate the MSMQ Adapters. When used with the Launcher, the MSMQ Adapters for sources or targets are configured using the Integration Flow Designer.

- [MSMQ Adapter commands](#)
- [Wildcard support](#)

MSMQ Adapter commands

The following table lists valid commands for the MSMQ Adapter, adapter command syntax, and if the adapter command is supported (✓) for use with data sources, data targets, or both.

Name	Syntax	Source	Target
Correlation ID (-CID) ¹	-CID <i>value</i>	✓	✓
Global Transaction Management (-GTX)	-GTX	✓	✓
Header (-HDR)	-HDR	✓	✓
Label (-LABEL)	-LABEL <i>message_label</i>	✓	✓
Listen (-LSN)	-LSN {0 S <i>wait_time_in_sec</i> }	✓	
Queue Name (-QN) ²	-QN <i>machine_name\queue_name</i> -or- -QN .\ <i>queue_name</i>	✓	✓
Quantity (-QTY)	-QTY { <i>value S</i> }	✓	
Trace (-T)	-T[E S] [<i>full_path</i>]	✓	✓
Transactional Queue (-TQ)	-TQ	✓	✓

¹The use of wildcards with this MSMQ Adapter command is supported. For more information about this support, see [Wildcard Support](#).

²For more information about specifying different kinds of queues, see [Queue Name \(-QN\)](#).

Wildcard support

The MSMQ Adapter supports the use of wildcards. Wildcards might be specified for either the Correlation ID adapter command (-CID) or the Label adapter command (-LABEL) on a single command line. The following rule for wildcards in the MSMQ Adapters applies:

An asterisk (*) represents a multi-character wildcard. The entire retrieved message is returned. If the wildcard is w*z and wxyz is returned, the wildcard * is expanded to xyz.

- [Correlation ID \(-CID\)](#)
- [Global Transaction Management \(-GTX\)](#)
- [Header \(-HDR\)](#)
- [Label \(-LABEL\)](#)
- [Listen \(-LSN\)](#)
- [Queue Name \(-QN\)](#)
- [Quantity \(-QTY\)](#)
- [Trace \(-T\)](#)
- [Transactional Queue \(-TQ\)](#)

Correlation ID (-CID)

Use the Correlation ID adapter command (-CID) to specify a particular correlation identifier (CID) for a source or target. Use this adapter command for a data source to retrieve messages from a queue with a specific correlation ID. You can also use it for a data target to assign a correlation ID to a message when placing it on a queue. To select a message with a specific correlation identifier, or to assign a specified value to a message for a data target, use the following syntax:

-CID *value*

Option

Description

value

Specify the application-defined correlation identifier. The value can be an alphanumeric string containing between 1 and 20 characters. For more information about the content and usage of MSMQ message properties, see the Microsoft MSMQ property definition PROPID_M_CORRELATIONID.

The MSMQ Adapters do not support Boolean operations on message properties other than AND. However, when a message to be mapped cannot be identified using the Correlation ID adapter command (-CID) and the Label adapter command (-LABEL) to specify the message identification properties PROPID_M_CORRELATIONID and PROPID_M_LABEL, additional message selection criteria can be performed using standard mapping techniques such as conditional mapping expressions, runtime functions, or functional maps.

When the Correlation ID adapter command (-CID) is specified on the command line of an output map card, the message is generated with the specified property.

Wildcards can be specified as the value for the correlation ID when running under the Launcher. See "[Wildcard Support](#)".

Global Transaction Management (-GTX)

Use the Global Transaction Management adapter command (-GTX) to indicate that the transactions for this card, whether input or output, should be processed as global transactions.

-GTX

For more information about global transaction management, see the *Global Transaction Management* documentation.

Header (-HDR)

Use the Header adapter command (-HDR) to receive or include MSMQ message properties with the body of the message. When the message stream includes data other than the message body, the additional data is referred to as the header. To include message properties other than the body of the message, use the header adapter command (-HDR).

-HDR

When the MSMQ Adapter is started without the Header adapter command (-HDR), the data consists only of the body of the message. A sample MSMQ type tree, named **msmq_in.mtt**, is provided in the **examples\adapters\msmq** folder. This type tree represents the content of the MSMQ message header properties. This type tree is configured to receive or set all properties and uses property names as identifiers that partition them into proper type groups.

When the MSMQ Adapter is started with the Header adapter command (-HDR) for a data source, the input to the map is automatically formatted as an array of records that includes all properties present in the messages being received, including the body.

When the MSMQ Adapter is started with the Header adapter command (-HDR) for a data target, the adapter searches for property identifiers in the data produced by the map and includes these properties in the MSMQ message output.

When the Header adapter command is specified, the body of the message is treated the same as any other property.

Usage of the Header command (-HDR) requires additional execution time and memory for the adapter to receive or set all of the message properties. Therefore, specifying the header command can affect performance. Performance depends upon the number of properties set on the received message.

Label (-LABEL)

Use the Label adapter command (-LABEL) to select a message using a specified message label value for a data source, or to create a message with a specified message label value for a data target. Use the following syntax:

-LABEL *message_label*

Option

Description

message_label

Specify the application-defined message label, that is represented in MSMQ with the property definition PROPID_M_LABEL. The value can be between 1 and 250 alphanumeric characters. If the value contains spaces it must be enclosed in double quotes.

When the Label adapter command (-LABEL) is used for a data source, it is a filter that selects messages from the queue based upon the message label properties. The MSMQ Adapters do not support Boolean operations on message properties other than AND. However, when a message to be mapped cannot be identified using the two

message identification properties, additional message selection criteria can be performed using standard mapping techniques such as conditional mapping expressions, runtime functions, or functional maps.

When the Label adapter command (-LABEL) is used for a data target, the message is generated with the specified filter property. Double quotation marks are not allowed in the LABEL value character string.

Wildcards might be specified for the value of the *message_label* when running under the Launcher. See "[Wildcard Support](#)".

Listen (-LSN)

Use the Listen adapter command (-LSN) for data sources to dictate the length of time (in seconds) that the MSMQ Adapters wait for one or more messages to be received. Map execution is suspended until the message is received or until the specified time lapses. The default value is S, meaning that the adapter waits indefinitely for a message to arrive in the queue.

```
-LSN {0|S|wait_time_in_sec}
```

Option**Description**

0

Do not wait at all. If there is no input available, the adapter does not wait for more messages.

S

This is an infinite wait time.

wait_time_in_sec

This is the number of seconds the adapter waits for a message

For example, to specify a wait time of 30 seconds, the syntax would be:

```
-LSN 30
```

Queue Name (-QN)

The Queue Name adapter command (-QN) is required for both data sources and data targets. Use the Queue Name adapter command (-QN) to identify the name of a specific MSMQ queue using the conventions for Microsoft MSMQ machine name and queue name.

```
-QN {. | machine_name } \queue_name
```

Option**Description**

.

This is a replacement for the local machine name.

machine_name

Specify a machine name.

queue_name

Specify a queue name.

In addition to the functionality described above, use this adapter command to support private queues and direct format names. Specify private queues using the following syntax:

```
-QN {. | machine_name } \PRIVATE$ \queue_name
```

where PRIVATE\$ specifies the usage of private queues. Private queues are registered on the local computer and not in the directory service.

- For MSMQ1.0, private queues are not registered in the MQIS SQL Server database.
- For MSMQ2.0, private queues are not registered in the Active Directory Services, which means that if MSMQ2.0 is used without Active Directory Services (Workgroup mode), private queues are the only supported queues.

Direct format names are supported, allowing specification of the queue name in the form.

If *queue_name* is specified using the direct format name and the queue is transactional, it is recommended that you specify the Transactional Queue (-TQ) adapter command. This command indicates to the adapter that this queue is transactional. If this command is not specified and the queue is transactional, the map might fail with the following error message:
The queue is not transactional

For public queues, use the following syntax:

```
-QN DIRECT=AddressSpecification\queue_name
```

For private queues, use the following syntax for a transactional queue:

```
-QN DIRECT=AddressSpecification\PRIVATE$ \queue_name -TQ
```

or the following syntax for a non-transactional queue:

```
-QN DIRECT=AddressSpecification\PRIVATE$ \queue_name
```

where AddressSpecification is one of the following:

SPX:MachineSPXNetworkAddress
(if SPX network protocol is used)

TCP:MachineIPNetworkAddress

(if TCP/IP network protocol is used)
An IP address can be IPv4 (for example, 1.2.3.4) or IPv6 (for example, a:b:c:d:0:1:2:3) format.

OS:MachineName

(where *MachineName* is the name of the machine recognized by the underlying operating system)

As indicated above, specify direct format names using **DIRECT=** preceding **AddressSpecification**.

Following below are examples of this syntax. The first example:

-QN DIRECT=SPX:00000012;00A0234F7500\Queue1 -TQ

specifies the transactional queue named **Queue1** on the machine with the SPX address **00000012;00A0234F7500**.

The next example:

-QN DIRECT=TCP:157.18.3.1\Queue2

specifies the queue named **Queue2** on the machine with an IP address of **157.18.3.1**. The adapter attempts to determine if this queue is transactional. If this effort is unsuccessful, the adapter assumes that this queue is not transactional because the **-TQ** adapter command was not specified.

The final example:

-QN OS:Jones\Queue3

specifies the queue named **Queue3** on the machine named **Jones** in the underlying operating system.

The transactional dead letter queue is specified as:

-QN machine_name;DEADXACT

or

-QN .;DEADXACT

for the local machine.

The non-transactional dead letter queue is specified as:

-QN machine_name ;DEADLETTER

or

-QN .;DEADLETTER

for the local machine.

Quantity (-QTY)

Use the Quantity adapter command (**-QTY**) to specify the number of messages to retrieve from the source queue. If the Quantity adapter command (**-QTY**) is not specified, the default value is 1.

-QTY {value|S}

Option

Description

S

This value returns all messages on the queue.

value

This is a positive integer representing the number of concurrent multiple messages to be retrieved.

For example, to specify a quantity of ten messages, the syntax would be:

-QTY 10

The Quantity adapter command (**-QTY**) defines the maximum number of messages across all bursts for a map.

When the Quantity adapter command (**-QTY**) is used, the message cursor is not returned to the head of the queue for other source cards in the same map using the identical queue.

The number of bursts in a single map execution can be controlled with the combination of values set in the Map Designer SourceRule...>FetchAs...>FetchUnit setting and the Quantity adapter command (**-QTY**).

When using the Quantity adapter command (**-QTY**) with the **S** option through either the Launcher or the Command Server, the Listen adapter command (**-LSN**) is also required with a value specified other than **S**.

Trace (-T)

Use the Trace adapter command (**-T**) to produce a trace file that contains information about connections to MSMQ, getting and putting messages on the queues, and MSMQ error codes.

The default filename is **msmqtrace.log** and it is created in the map directory unless otherwise specified. However, it is recommended that you specify that the trace file is created in a root directory, such as c:\ on Windows, because the adapter logs some events before the full path of an optional destination directory is known.

If the trace file does not already exist, it is created. If it already exists it is automatically appended.

-T[E|S] [full_path]

Option

Option	Description
E	Error mode. Produce a trace file containing only the adapter errors that occurred during map execution.
S	Summary mode. Use this option to reduce the size of the trace file. The summary trace file excludes polling information and function entries and exits-reporting only minimal information, such as the activity that occurs within a function when applicable.
full_path	Create a trace file with the specified name in the specified directory. By default, the directory is where the map is located and the file name is msmqtrace.log .

You can override the adapter command line trace options dynamically using the Management Console. See "Dynamic Adapter Tracing" in the *Launcher* documentation for detailed information.

Transactional Queue (-TQ)

Use the Transactional Queue adapter command (-TQ) to identify that a queue specified using the Queue Name (-QN) adapter command is transactional when the adapter cannot obtain this information internally from within the queue. If the adapter cannot obtain this information externally, as in when a remote private queue is accessed, it will search for specification of the -TQ adapter command to determine if the queue is transactional.

-TQ

When the -TQ adapter command is specified, the adapter will assume that the queue is transactional.

The MSMQ Adapter does not support transactional read from non-local remote queues, because it is not implemented by MSMQ. However, Microsoft provides a workaround for this feature which can be found at:

<http://msdn.microsoft.com/library/default.asp?URL=/library/backgrnd/html/msmqtips.htm>

Syntax summary

This documentation displays the syntax of the MSMQ Adapter commands for data sources and targets.

- [Data sources](#)
- [Data targets](#)
- [MSMQ maps](#)
- [MSMQ type trees](#)
- [Using the MSMQ type trees](#)

Data sources

The following is the syntax of the MSMQ Adapter commands used for data sources:

```
-QN machine_name\queue_name [-CID value]
[-LABEL value]
[-LSN {0|S} wait_time_in_sec]
[-QTY {value|S}]
[-HDR]
[-GTX]
[-T[E|S] [full_path]
[-TQ]]
```

Data targets

The following is the syntax of the MSMQ Adapter commands used for data targets:

```
-QN machine_name\queue_name [-CID value]
[-LABEL value]
[-HDR]
[-GTX]
[-T[E|S] [full_path]
[-TQ]]
```

MSMQ maps

The map source file (*install_dir/examples/adapters/msmq/msmq.mms*) contains seven maps that show how the MSMQ Adapter works.

- [Map1](#)
- [Results1](#)
- [Change Map1](#)
- [Results2](#)
- [Map2](#)
- [Results1](#)
- [Change Map2](#)
- [Results2](#)
- [Map3](#)
- [Results](#)
- [Map4](#)
- [Results](#)
- [Map5](#)
- [Results1](#)
- [Put messages on the queue](#)
- [Results2](#)
- [Map6](#)
- [Results](#)
- [Map7](#)

Map1

The content of the **FileIn1.txt** source text file on the **TextFile** input card 1 maps to the message on the **.\queue1** MSMQ transactional queue (output card).

The **PUT->Target->Command** setting defined in the Map Designer for the **MSMQQueue** output card 1 is:

```
-QN .\queue1 -HDR -T
```

The Queue Name adapter command (-QN .\queue1) specifies that the data from the input card should be delivered as a message to the queue named **.\queue1**. The Header adapter command (-HDR) specifies inclusion of the message header properties other than the body of the message. These properties are LABEL (LABEL) and CORRELATION ID (CID). The Trace adapter command (-T) activates the adapter trace function to track function calls and to store them in a trace log file. The default log file name is **msmqtrace.log**.

Before you run the **Map1** map, be sure that the **.\queue1** transactional queue already exists or change the queue name in the **PUT->Target->Command** setting to some existing transactional queue. Purge the queue before you run the map.

The **MSMQQueue** output card 1, shown below, defines the message having the same content (message body) as the input file from the **TextFile** input card 1. The PROPID_M_CORRELATIONID message header property VT_CAB is defined as **CIDA**. The PROPID_M_LABEL message header property VT_LPSTR is defined as **LabelA**.

Run **Map1** and check the content of the **.\queue1** queue.

Results1

The queue now contains a message labeled **LabelA** with the following message body:

```
This sentence is a 46 bytes long message body.
```

Change Map1

Change the map rules for the **MSMQQueue** output card 1 for the PROPID_M_CORRELATIONID message header property VT_CAB component to CIDB and the PROPID_M_LABEL message header property VT_LPSTR component to **LabelB**. Run **Map1** again.

Results2

After you run **Map1** with the new map rules, the queue contains two messages. the first message with **LabelA** and **CIDA** and the new message with **LabelB** and **CIDB**.

Map2

Map2 maps a message from the **.\queue1** queue to both the **.\queue2** queue and a text file named **FileOut.txt**. **Map2** uses the two messages generated by **Map1**. These two messages are on the **.\queue1** queue. These messages have the PROPID_M_CORRELATIONID message header property VT_CAB = **CIDA** (first message) and **CIDB** (second message), and the PROPID_M_LABEL message header property VT_LPSTR = **LabelA** (first message) and **LabelB** (second message).

The GET>Source>Command setting for the **MSMQQueue1** input card 1 is:

```
-QN .\queue1 -LABEL LabelB -CID CIDB -HDR -T -LSN 5
```

The Queue Name adapter command (-QN .\queue1) specifies retrieval of messages on the queue named .\queue1. The Label adapter command (-LABEL LabelB) and the Correlation ID adapter command (-CID CIDB) specify retrieval of messages with the **LabelB** label and the **CIDB** correlation ID. The Header adapter command (-HDR) specifies retrieval of the message body and message header properties. The Trace adapter command (-T) enables tracing and tracks function calls and stores the trace information in the default file named **msmqtrace.log**. The Listen adapter command (-LSN 5) specifies a wait period of no more than five seconds for a message to arrive if that message is not already on the queue.

On the output side of the map are two cards. The first card specifies the **FileOut.txt** file. The second card specifies another MSMQ queue: .\queue2. The map specifies that all operations on the .\queue queue (input card) will belong to the transaction with a map scope (**OnFailure** = Rollback, **Scope** = Map) and that operations on the .\queue2 queue will not be in any transaction (**OnFailure** = Commit). This means that the .\queue2 queue can be a non-transactional queue.

Notice that the input card has an option (**OnSuccess** = Keep) that indicates that a copy of the message will remain on the source queue (.queue1).

The **MSMQQueue1** input card 1 has an **OnSuccess** setting of Keep, specifying that a copy of every message will remain on the .\queue1 source queue.

Run **Map2**.

Results1

Use the MSMQ Explorer to check the contents of .\queue2 queue. The message with the **LabelB** label is there. Both messages are still on the .\queue1 queue. In the **FileOut.txt** output file, notice that it contains the body of the received message.

Change Map2

Change the GET>Source>Command setting on the **MSMQQueue1** input card 1 to:

```
-QN .\queue1 -LABEL LabelA -HDR -T -LSN 5
```

Run **Map2** with this new GET>Source>Command setting.

Results2

The (**LabelA, CIDA**) message will be copied to the .\queue2 queue.

Map3

Map3 shows how to use the **PUT** function to place multiple messages on the queue. The **TextFile** input card 1 specifies a source text file named **FileIn2.txt** that contains the following structure:

```
This is Message 001  
This is Message 002  
This is Message 003  
This is Message 004  
...  
This is Message 099  
This is Message 100
```

Map3 uses these 100 text strings to create 100 messages on the queue named .\queue1 with labels entitled **Label001**, **Label002**, and so on, up to **Label100**.

The **MSMQQueue** output card 1 defines the **PUT>Target** setting as Sink, which means that there is no target. The Map Designer will execute the **PUT** function in the map rule of the **MSMQQueue** output card 1 for each occurrence of the **Body:TextFile** object. The Map Designer will call the MSMQ Adapter, which is specified as the first argument of the function "msmq". That means that this **PUT** function will execute 100 times-once for each message.

Non-printable characters are not displayed in the **Rule Bar**. They are represented in the Rule cell on the To window as a character.

The **File_In.MTT** type tree is specified for the **TextFile** input card as the data source. The type name is **File data**. The **Body** type is a component of **File** with a component range of S (some).

In the **TextFile** input card settings for **Map3**, the **GET>Source** setting is File. The **GET>Source>FilePath** setting is **FileIn2.txt**.

The first argument of the **PUT** function, "msmq", is the adapter alias that specifies the MSMQ Adapter.

The second argument of the **PUT** function is the adapter command plus the **RIGHT** function that returns a specified number of characters from a text expression beginning with the rightmost byte of a text item:

```
"-QN .\queue1 -T -LABEL Test001" + RIGHT(TEXT(Body:TextFile), 3)
```

The **RIGHT(TEXT(Body:TextFile), 3)** function takes the three right-most characters of the **Body** type on the input card named **TextFile**. For the first input string **This is Message 001**, the adapter command is:

```
-QN .\queue1 -T -LABEL Test001
```

For the second input string This is Message 002, the adapter command is:

```
-QN .\queue1 -T -LABEL Test002
```

And so on, concluding with the last message adapter command:

```
-QN .\queue1 -T -LABEL Test100
```

The third parameter of the `PUT` function specifies the content of the data message that is passed to the adapter. Because the Header adapter command (`-HDR`) is not specified, all data passed to the adapter should be treated as message bodies.

```
"<BODY>" + SYSTEM(13) + SYSTEM(10) + Body:TextFile + "</BODY>"  
+ SYSTEM(13) + SYSTEM(10)
```

The decimal value for a carriage return is 13. The decimal value for a line feed is 10. The carriage return and line feed are obtained using the `SYSTEM(13)` and `SYSTEM(10)` functions.

`<BODY><CR><LF>` is the literal initiator for the message body passed to the adapter and `</BODY><CR><LF>` is the literal terminator.

The `SYSTEM` function must be used because `<CR>` and `<LF>` are non-printable characters.

Before you run **Map3**, purge the `.\queue1` queue. Run **Map3**.

Results

The `.\queue1` queue will have 100 different messages.

Map4

Map4 uses burst mode to map multiple messages from one queue to another. Ten messages are transferred from the `.\queue1` queue to the `.\queue2` queue.

The `GET...Source...Transaction...OnSuccess` setting is `Delete`, specifying that the messages retrieved from the input queue will be deleted from that queue. The `SourceRule...FetchAs` setting is `Burst`, specifying retrieval of the messages in the burst units as specified in the `SourceRule...FetchUnit` setting. The `GET...Source...Command` setting is:

```
-QN .\queue1 -HDR -QTY 10 -LSN 3 -T
```

The Quantity adapter command (`-QTY 10`) specifies that ten messages should be retrieved from the `.\queue1` queue. The `FetchUnit` setting of 1 specifies that one message is retrieved per burst.

Before you run this map, `.\queue1` should contain at least 10 messages. You can run **Map3** to place 100 messages on it. These messages are needed to run **Map4**. Run **Map4**.

Results

Ten messages are transferred from the `.\queue1` queue to the `.\queue2` queue.

Map5

Map5 introduces some new features, including the ability to get more than one message per burst and using the `PUT` function to call adapter functions. **Map5** will attempt to map 105 messages from the `.\queue1` queue to the `.\queue2` queue. The `MSMQQueue` input card value for the `GET...Source` setting is `MSMQ`. The output card `Sink1` value for the `PUT...Target` setting is `Sink`, which is an empty output.

The Map Designer runs the `PUT` function on exit only for each appearance of the `MessageContainer` object on the input side, which is a component of the `ConcatenatedMessages` group type in the `Msmq_In.mtt` type tree.

The `Message` group type is a component of the `MessageContainer` group type. The `Message` group type contains one message with the literal message initiator value of `<MESSAGE><CR><LF>` and the literal message terminator value of `</MESSAGE><CR><LF>`. These initiators and terminators are for messages with more than one property, as opposed to the `<BODY><CR><LF>` initiators and terminators for messages with only body (content).

The `GET...Source...Command` setting for the `MSMQQueue` input card is:

```
-QN .\queue1 -HDR -QTY 105 -LSN 3 -T
```

The Quantity adapter command (`-QTY 105`) specifies retrieval of 105 messages from the `.\queue1` queue. The Listen adapter command (`-LSN 3`) specifies a wait period of not more than three seconds per burst. The Header adapter command (`-HDR`) specifies inclusion of all properties of the messages. The Trace adapter command (`-T`) specifies that function calls will be tracked in the `msmqtrace.log` file located in the map directory.

The map rule for the **Sink1** output card specifies the **PUT** function.

This map rule specifies the usage of the MSMQ Adapter. For each occurrence of the **MessageContainer** data object, extract the message label and message body from the input message and call the adapter function with the following adapter command:

```
-QN .\queue2 -T -LABEL label_constructed_from_input
```

Where *label_constructed_from_input* is the message label value constructed from the type values in the input card.

In the input card, the value of the **SourceRule**.>**FetchAs**.>**FetchUnit** setting is 10, specifying retrieval of ten messages per burst.

Before running **Map5**, purge the .\queue1 queue.

Results1

Attempt to run the map when the .\queue1 queue is empty. Because there are no messages on the .\queue1 queue, the adapter will wait for three seconds. After that, it will issue a warning. Because the input card **MSMQQueue** setting for the **Source**.>**Transaction**.>**Warnings** map setting is Fail, the warning will cause **Map5** to fail with the message **Source not available**. If you change the **Warnings** setting to Ignore, the map will run successfully.

Put messages on the queue

Run **Map3** to put 100 messages on the .\queue1 queue.

Run **Map5**.

Results2

Ten messages are retrieved from the .\queue1 queue ten times. These 100 messages are placed on the .\queue2 queue. An attempt will be made to retrieve five additional messages based upon the Quantity adapter command value (-QTY 105). Because there are no more new messages on the .\queue1 queue, the adapter will wait for three seconds (-LSN 3) and then it will give up. But this time, the warning is not issued. The map completes successfully, but only 100 messages are placed on the .\queue2 queue.

Map6

Map6 describes the use of response, administration, and dead letter queues. To run **Map6**, you must have three queues: **queue1**, **queue2**, and **queue3**. Note that **queue1** and **queue2** are transactional queues; **queue3** is non-transactional.

The **MSMQQueue** output card 1 defines placing a message on **queue1** having the same content (message body) as the input file from the **TextFile** input card 1. The PROPID_M_CORRELATIONID message header property VT_CAB is defined as **MyCID**. The PROPID_M_LABEL message header property VT_LPSTR is defined as **MyLabel**.

The output card uses the **Msmq_out_extended.mtt** type tree, which adds some message header properties and message header property variant types to the **Msmq_out.mtt** type tree used in previous map examples.

The following table describes properties for **Map6**:

Property

Specifies

PROPID_M_ACKNOWLEDGE

This is the type of acknowledgment messages that MSMQ posts in the administration queue when the message is sent:

MQMSG_ACKNOWLEDGMENT_NONE 0x00 MQMSG_ACKNOWLEDGMENT_POS_ARRIVAL 0x01 MQMSG_ACKNOWLEDGMENT_POS_RECEIVE 0x02
MQMSG_ACKNOWLEDGMENT_NEG_ARRIVAL 0x04 MQMSG_ACKNOWLEDGMENT_NEG_RECEIVE 0x08

PROPID_M_TIME_TO_REACH_QUEUE

This is the time limit, in seconds, for the message to reach the queue

PROPID_M_TIME_TO_BE_RECEIVED

This is the maximum length of time, in seconds, for a message to be received from the destination queue. This includes the time getting to the target queue and the time waiting in the queue before it is retrieved by an application.

PROPID_M_JOURNAL

This determines whether the message should be kept in a machine journal queue (on the originating machine), sent to a dead letter queue, or neither of the above.

PROPID_M_ADMIN_QUEUE

This is the queue used for MSMQ-generated acknowledgment messages.

PROPID_M_RESP_QUEUE

This is the queue to which application-generated response messages are returned.

PROPID_M_ACKNOWLEDGE is set to 8, representing the value MQMSG_ACKNOWLEDGMENT_NACK_RECEIVE. This means that notification is sent only when the message does not arrive at the **queue1** destination queue within the PROPID_M_TIME_TO_REACH_QUEUE-specified time of five seconds or that it arrives, but is not taken from the destination queue within the PROPID_M_TIME_TO_BE_RECEIVED-specified time of 50 seconds.

Notification messages are sent to **queue3**, which has been specified by PROPID_M_ADMIN_QUEUE as the administration queue. It is the user's responsibility to specify the queue to be used as the administration queue. The administration queue must be non-transactional.

The PROPID_M_JOURNAL property is set to 1—the value of the QMSG_DEADLETTER constant. This specifies that undelivered messages are to be placed on a dead-letter queue. MSMQ has the following two dead-letter queues:

- one for transactional messages (messages that are sent to a transactional queue) with a predefined name of **Xact Dead Letter**, and
- one for non-transactional messages (messages that are sent to a non-transactional queue) with a predefined name of **Dead Letter**.

Journal queues are used to track sent messages and to store report messages from MSMQ. Had PROPID_M_JOURNAL specified instead that the message be kept in a machine journal queue, MSMQ would have created a journal queue with the predefined name **Journal** in each machine added to the MSMQ enterprise and in each new queue.

The last property used in this type tree is PROPID_M_RESP_QUEUE. It is used to tell the receiving application to send a response message to the defined response queue (**queue2** in this example). The response queue can be either transactional or non-transactional.

Before you run the map, you must first create **queue1**, **queue2**, and **queue3**. Purge these queues to ensure they are clear of any old messages. You should also purge the **Xact Dead Letter** queue.

Run **Map6**.

Results

The message is placed on the **queue1** queue. If you go to the MSMQ Explorer quickly, you can see the **MyLabel** message on **queue1**. Right click on the message and select Properties. You will be able to see that **queue3** is the administration queue for this message and that **queue2** is the response queue.

After a few seconds, the message disappears from the queue. Because it was not received in 50 seconds, as specified in PROPID_M_TIME_TO_BE_RECEIVED, MSMQ will put the message onto the **Xact Dead Letter** queue. The **queue1** destination queue is transactional. If you right-click on this message in the **Xact Dead Letter** queue, the value of class property is Time to be received expired.

Because the administration queue has been specified, a copy of the **MyLabel** message will also be placed on the **queue3** administration queue.

Map7

Map7 explains how to use a map to read messages from dead letter queues. After **Map6** has been run, the **MyLabel** message has been placed on the **Xact Dead Letter** queue. **Map7** takes the message from this queue and stores its body to the **FileXact.txt** output file.

Similar to any other queue, dead letter queues are specified in adapter commands using the Queue Name adapter command (-QN). However, dead letter queues have a special naming convention. Each machine can have only one transactional dead letter queue and one non-transactional dead letter queue.

The transactional dead letter queue is specified as:

```
-QN machine_name;DEADXACT
```

or

```
-QN .;DEADXACT
```

for the local machine.

The non-transactional dead letter queue is specified as:

```
-QN machine_name;DEADLETTER
```

or

```
-QN .;DEADLETTER
```

for the local machine.

Because the **queue1** destination queue is transactional, the GET>Source>Command input card setting in **Map7** is:

```
-QN .;DEADXACT -HDR -T
```

Before you run **Map7**, you must first run **Map6**. **Map6** places the message on the dead letter queue that **Map7** is reading. Then run **Map7**.

The message from the **Xact Dead Letter** queue is mapped to the **FileXact.txt** file. Because the input card has the Source>Transaction>OnSuccess setting of Delete, the message will be removed from the queue after it has been received.

MSMQ type trees

Four MSMQ type trees are provided in the **examples\adapters\MSMQ** folder. These types describing different MSMQ message properties are provided so you can easily select the message types needed for input or output cards used in the Map Designer. Use a type tree appropriate for the data source. If the Header adapter command (-HDR) is not specified, the type tree representing the MSMQ message properties should not be used.

The **MSMQ_in.mtt** sample type tree should be used as a guide when creating new type trees to facilitate mapping of individual properties. This type tree is configured to receive or set all properties and it uses property names as identifiers that partition them into proper type groups. When setting or receiving individual message properties,

the type tree must define these properties exactly.

Using the MSMQ type trees

Depending upon how you want to map, either select one of the **Message** types as input or output or add a message content type as a **MessageData** component.

The type tree is arranged so you can integrate the definitions provided with any message data you want to define. The input card contains all message header properties. The output card contains only the message header properties you want. To optimize performance on the output card, select MSMQ_out.mtt and delete any unnecessary message header properties.

MSMQ message header properties

MSMQ message properties aid in identifying the message, routing the message, authenticating and signing the message, creating notifications, and tracing the message as it is being routed. MSMQ messages consist of message properties, each having its own name, type, value, and optional length.

Some MSMQ properties are of composite or aggregate type and thus have associated LEN (length) properties. Mapping of the LEN properties is not explicitly permitted. However, lengths are part of the record of the property to which the length belongs and are crucial to the mapping process.

An MSMQ property has a property ID that is an enumerated type representing the string of the property name. While MSMQ deals internally with the property IDs, property name strings are used to aid in mapping and tracing the mapping of properties. Each property has a variant type and a value.

- [MSMQ property variant type and values](#)
- [Message property definitions and values](#)

MSMQ property variant type and values

Each MSMQ property has a variant type and a value. There are five basic variant types:

- VT_CAB
 - VT_CLSID
 - VT_LPSTR
 - VT_UI1
 - VT_UI4
-
- [VT CAB](#)
 - [VT CLSID](#)
 - [VT LPSTR](#)
 - [VT UI1](#)
 - [VT UI4](#)

VT_CAB

This composite record type has a size field and a binary stream field. Except for PROPID_M_SECURITY_CONTEXT (cannot be received), PROPID_M_SENDERID, PROPID_M_MSGID and PROPID_M_SIGNATURE (cannot be set), all properties of this type can be used in mapping. MSMQ limits the size of PROPID_M_CORRELATIONID and PROPID_M_MSGID to 20 bytes. The adapter copies only the first 20 bytes in and out.

The following properties are in this group:

- PROPID_M_DEST_SYMM_KEY
- PROPID_M_SENDERID
- PROPID_M_BODY
- PROPID_M_MSGID (20 bytes fixed)
- PROPID_M_EXTENSION
- PROPID_M_SIGNATURE
- PROPID_M_CORRELATIONID (20 bytes fixed)
- PROPID_M_CORRELATIONID (20 bytes fixed)

VT_CLSID

This CLSID (GUID) record type is 16 bytes long. The adapter does not need a size field for this record because it is a fixed length field. It is critical that the data must be exactly 16 bytes long for proper input to MSMQ. The type tree should validate this.

The following properties are in this group:

- PROPID_M_CONNECTOR_TYPE
- PROPID_M_SRC_MACHINE_ID

VT_LPSTR

This is an ANSI string type. Properties of this type have an associated length property of type VT_UI4. The MSMQ Adapter performs conversions to UNICODE strings.

The MSMQ Adapter allows mapping of all properties of this type exclusive of PROPID_M_XACT_STATUS_QUEUE that cannot be set on an output card. The MSMQ Adapter expects the input data not to be null-terminated and depends upon the length record (four bytes). MSMQ limits the size of PROPID_M_LABEL to 250 characters and although the MSMQ Adapter does not place this limitation, the actual MSMQ property is set to the first 250 characters.

The following properties are in this group:

- PROPID_M_ADMIN_QUEUE
- PROPID_M_PROV_NAME
- PROPID_M_XACT_STATUS_QUEUE
- PROPID_M_DEST_QUEUE
- PROPID_M_RESP_QUEUE
- PROPID_M_LABEL (250 bytes max)

VT_UI1

This is an unsigned byte binary type. Properties of this type are assumed to be in their binary format when passed to and from the MSMQ Adapter. The proper type and conversion must take place. The adapter does not need a size field for this record because it is a fixed length field.

The following properties are in this group:

- PROPID_M_DELIVERY
- PROPID_M_ACKNOWLEDGE
- PROPID_M_TRACE
- PROPID_M_JOURNAL
- PROPID_M_PRIORITY

VT_UI4

This is an unsigned, four-byte binary type. Properties of this type are assumed to be in their binary format when passed to and from the MSMQ Adapter. The proper type conversion must take place.

The properties of this type that cannot be mapped are: PROPID_M_AUTH_LEVEL (cannot be received) and PROPID_M_SECURITY_CONTEXT (cannot be received). The adapter does not need a size field for this record because it is a fixed length field.

A number of time-stamping properties belong to this group. MSMQ time stamps are expressed in seconds since midnight of 01/01/1970 and the adapter returns data in this format. This release version includes runtime functions for conversion to and from this data format and a number of character and binary-based time formats. There are a few VT_UI2 properties that are returned as VT_UI4 (input card) and are property-casted to preserve the value.

When passing data on the output card, the user can create a new type with data that is two bytes long and the adapter correctly casts the data.

The following properties are in this group:

- PROPID_M_TIME_TO_BE_RECEIVED
- PROPID_M_AUTH_LEVEL
- PROPID_M_HASH_ALG
- PROPID_M_PROV_TYPE
- PROPID_M_APPSPECIFIC
- PROPID_M_TIME_TO_REACH_QUEUE
- PROPID_M_ENCRYPTION_ALG
- PROPID_M_PRIV_LEVEL
- PROPID_M_SECURITY_CONTEXT
- PROPID_M_BODY_TYPE

Message property definitions and values

The following general message properties can be set by the application. Length properties associated with another property are not included. The adapter converts normal strings to UNICODE when sending/receiving to/from MSMQ.

The following descriptions provide allowed values and, where appropriate, the hexadecimal values for the specific message property values. These values are present in the message header and are used when mapping message property data. This information is provided as a tool for mapping.

- [PROPID_M_ACKNOWLEDGE](#)
- [PROPID_M_ADMIN_QUEUE](#)
- [PROPID_M_APPSPECIFIC](#)
- [PROPID_M_BODY](#)
- [PROPID_M_BODY_TYPE](#)
- [PROPID_M_CONNECTOR_TYPE](#)
- [PROPID_M_CORRELATIONID](#)

- [PROPID_M_DELIVERY](#)
 - [PROPID_M_DEST_QUEUE](#)
 - [PROPID_M_EXTENSION](#)
 - [PROPID_M_JOURNAL](#)
 - [PROPID_M_LABEL](#)
 - [PROPID_M_PRIORITY](#)
 - [PROPID_M_RESP_QUEUE](#)
 - [PROPID_M_TIME_TO_BE_RECEIVED](#)
 - [PROPID_M_TIME_TO_REACH_QUEUE](#)
 - [PROPID_M_TRACE](#)
 - [PROPID_M_XACT_STATUS_QUEUE](#)
-

PROPID_M_ACKNOWLEDGE

This property specifies the type of acknowledgment messages that MSMQ posts in the administration queue when the message is sent.

Data format: Unsigned byte

Allowed Value	Hex Value
MQMSG_ACKNOWLEDGMENT_NONE	0x00
MQMSG_ACKNOWLEDGMENT_POS_ARRIVAL	0x01
MQMSG_ACKNOWLEDGMENT_POS_RECEIVE	0x02
MQMSG_ACKNOWLEDGMENT_NEG_ARRIVAL	0x04
MQMSG_ACKNOWLEDGMENT_NEG_RECEIVE	0x08

PROPID_M_ADMIN_QUEUE

This property specifies the queue used for MSMQ-generated acknowledgment messages.

Data format: Composite record with unsigned long integer size field, followed by zero-terminated UNICODE string up to 250 UNICODE characters long

Allowed values: Length of string in ULONG followed by string

PROPID_M_APPSPECIFIC

This property specifies an application specific index of values that can be used for sorting messages.

Data Format

Allowed Values

Unsigned long integer

Any application-specific index that fits in an unsigned long integer

PROPID_M_BODY

This property contains the body of the message and has PROPID_M_BODY_SIZE as an associated property.

Data Format

Allowed Values

Composite record with unsigned long integer size field, followed by variable-length byte stream of length given in size field

Length of byte stream in ULONG, followed by byte stream

PROPID_M_BODY_TYPE

This property indicates the type of body contained in the message.

Data format: Unsigned long integer

Allowed values:

- VT_EMPTY 0
- VT_NULL 1
- VT_I2 2
- VT_I4 3
- VT_R4 4
- VT_R8 5
- VT_CY 6
- VT_DATE 7

- VT_BSTR 8
- VT_DISPATCH 9
- VT_ERROR 10
- VT_BOOL 11
- VT_VARIANT 12
- VT_UNKNOWN 13
- VT_DECIMAL 14
- VT_I1 16
- VT_UI1 17
- VT_UI2 18
- VT_UI4 19
- VT_I8 20
- VT_UI8 21
- VT_INT 22
- VT_UINT 23
- VT_VOID 24
- VT_HRESULT 25
- VT_PTR 26
- VT_SAFEARRAY 27
- VT_CARRAY 28
- VT_USERDEFINED 29
- VT_LPSTR 30
- VT_LPWSTR 31
- VT_FILETIME 64
- VT_BLOB 65
- VT_STREAM 66
- VT_STORAGE 67
- VT_STREAMED_OBJECT 68
- VT_STORED_OBJECT 69
- VT_BLOB_OBJECT 70
- VT_CF 71
- VT_CLSID 72
- VT_VECTOR 0x1000
- VT_ARRAY 0x2000
- VT_BYREF 0x4000
- VT_RESERVED 0x8000
- VT_ILLEGAL 0xffff
- VT_ILLEGALMASKED 0xffff
- VT_TYPEMASK 0xffff

PROPID_M_CONNECTOR_TYPE

This property specifies the application connector type the message is using.

Data Format

Allowed Values

16-byte byte stream

16-byte byte stream

This must be exactly 16 bytes.

PROPID_M_CORRELATIONID

This property specifies the correlation identifier of the message.

Data Format

Allowed Values

Composite record with unsigned long integer size field, followed by variable-length byte stream of length given in size field up to 20 bytes

Length of byte stream in ULONG, followed by byte stream

PROPID_M_DELIVERY

This property specifies how the message is delivered.

Data Format	Allowed Values	
Unsigned byte	MQMSG_DELIVERY_EXPRESS	0
Unsigned byte	MQMSG_DELIVERY_RECOVERABLE	1

PROPID_M_DEST_QUEUE

This property specifies the target queue of the message and has PROPID_M_DEST_QUEUE_LEN as associated property.

Data Format

Allowed Values

Composite record with unsigned long integer size field, followed by zero-terminated UNICODE string up to 250 UNICODE characters
Length of string in ULONG, followed by string

PROPID_M_EXTENSION

This property provides a place to put additional information associated with a message and has PROPID_M_EXTENSION_LEN as associated property.

Data Format

Allowed Values

Composite record with unsigned long integer size field, followed by variable-length byte stream of length given in size field
Length of byte stream in ULONG, followed by byte stream

PROPID_M_JOURNAL

This property specifies if the message should be kept in a machine journal on the originating machine, sent to a dead letter queue, or no action should be taken.

Data Format	Keyword	Value
Unsigned byte	MQMSG_JOURNAL_NONE	0
Unsigned byte	MQMSG_DEADLETTER	1
Unsigned byte	MQMSG_JOURNAL	2

PROPID_M_LABEL

This property specifies a label of the message and has an associated PROPID_M_LABEL_LEN.

Data Format

Allowed Values

Composite record with an unsigned long integer size field, followed by zero-terminated UNICODE string up to 250 UNICODE characters
Length of string in ULONG, followed by string

PROPID_M_PRIORITY

This property specifies the message priority. A low number indicates low priority.

Data Format	Keyword	Values
Unsigned byte	MQ_MIN_PRIORITY	0 through 7
Unsigned byte	MQ_MAX_PRIORITY	0 through 7

PROPID_M_RESP_QUEUE

This property specifies the queue to which application generated response messages are returned. It has an associated PROPID_M_RESP_QUEUE_LEN property.

Data Format

Allowed Values

Composite record with unsigned long integer size field, followed by a zero-terminated UNICODE string up to 250 UNICODE characters
Length of string in ULONG, followed by string

PROPID_M_TIME_TO_BE_RECEIVED

This property specifies the total time, in seconds, that the message is allowed to live. This includes the time getting to the target queue and waiting in the queue before it is retrieved by an application.

Data Format

Allowed Values

Unsigned long integer
Time in seconds

PROPID_M_TIME_TO_REACH_QUEUE

This property specifies a time limit, in seconds, for the message to reach the queue.

Data Format

Allowed Values

Unsigned long integer
Time in seconds

PROPID_M_TRACE

This property specifies where report messages are sent when tracing a message.

Data Format	Keyword	Value
Unsigned byte	MQMSG_TRACE_NONE	0
Unsigned byte	MQMSG_SEND_ROUTE_TO_REPORT_QUEUE	1

PROPID_M_XACT_STATUS_QUEUE

This property identifies the transaction status queue on the source computer. This has an associated PROPID_M_XACT_STATUS_QUEUE_LEN property.

Data Format

Allowed Values

Composite record with unsigned long integer size field, followed by a zero-terminated UNICODE string up to 250 UNICODE characters
Length of string in ULONG, followed by string

Troubleshooting

For information about error codes and messages returned by the adapters, see [Return Codes and Error Messages](#).

Troubleshooting for the MSMQ Adapter includes the following:

- using the trace file generated with the Trace adapter command
- confirming your login permissions
- using transactional messages in a non-transactional queue
- [Trace log](#)
- [Error returned to engine \(-5\)](#)

Trace log

The Trace adapter command creates an internal trace file that records all of the input commands, called functions, found messages, and return values from all called functions. This data is recorded in the **msmqtrace.log** file located in the directory where the compiled map resides.

Error returned to engine (-5)

The following audit execution log file message

Error returned to engine: (-5) Could not open queue, MSMQ error MSMQerror_code

can be generated for different reasons:

- login permissions not established for you
Login permissions can be set at the server and corresponding queue object level. Whatever you have used as a login ID does not match the respective setting, disallowing your access to the queue.
The Launcher uses the `system` login ID as a default. You might need to change the Microsoft Windows control panel **Services** settings.
- transactional messages on non-transactional queues
You are trying to put or get a transactional message to or from a non-transactional queue.

If you are having trouble doing a `PUT` and the log file indicates that the properties are correctly set, make sure the property you are trying to set does not involve a resource that you do not have or that you are referencing incorrectly. For example, you might reference an encryption algorithm that is not supported on the other end or you might have a sender security certificate that must be validated against an external certificate authority that is not properly configured.

Return codes and error messages

Return codes and messages are returned when the particular activity completes. Return codes and messages might also be recorded as specified in the audit logs, trace files, execution summary files, and so forth.

- [Messages](#)

Messages

The following is a listing of all the codes and messages that can be returned as a result of using the MSMQ Adapter for sources or targets.

In addition to the following return codes, the MSMQ Adapter can return MSMQ error codes. Consult the MSMQ documentation for descriptions of these codes.

Adapter return codes with positive numbers are warning codes that indicate a successful operation. Adapter return codes with negative numbers are error codes that indicate a failed operation.

Return Code	Message
-1001	Trace file could not be accessed
-1002	Adapter command is incorrect
-1003	Failed to load and initialize MSMQ library
-1004	Queue could not be opened
-1005	Message could not be stored on the queue
-1006	Message could not be retrieved from the queue
-1007	Transaction could not be rolled back explicitly
-1008	Transaction could not be committed
-1009	Queue could not be closed
-1010	Invalid message format
-1011	Transaction monitor not supported
-1012	Problem occurred on the queue cursor
-1013	New transaction could not be started
-1014	Queue path could not be translated to the format name
-1015	Machine name could not be translated to the machine GUID
-1016	Transaction usage was required for a non-transactional queue

Microsoft SQL Server Adapter

This documentation discusses the Microsoft SQL Server Adapter. You can use the adapter with the Command Server, Launcher, Software Development Kit, or with a map in a map rule.

- [Overview](#)
- [System requirements](#)
- [Database columns and types](#)
- [Database Interface Designer settings](#)
- [Microsoft SQL Server Adapter commands](#)
- [Extensions for the Launcher](#)
- [Bulk copy example files](#)
- [Restrictions and limitations](#)
- [Return codes and error messages](#)

Overview

Use a Microsoft SQL Server Adapter for Windows to access and manipulate data contained in databases that are Microsoft SQL Server data sources. You can also install this adapter on additional systems for remote database connectivity.

System requirements

In addition to any requirements in the release notes, to install and run the Microsoft SQL Server database adapter on your Windows platform:

- Verify that you have installed Microsoft SQL Server or Server Client Utilities. For information about this, see your Microsoft SQL Server documentation.
- Set the following optional environment variables that can be used to satisfy the requirements for particular situations:
 - DTX_LOCALE_CHARSET
 - DTX_TEMPDB
 - DTX_SQLSVR_CONN_LIMIT
- [DTX_LOCALE_CHARSET](#)
- [DTX_TEMPDB](#)
- [DTX_SQLSVR_CONN_LIMIT](#)

DTX_LOCALE_CHARSET

Set the DTX_LOCALE_CHARSET environment variable to enable the usage of non-English characters in data. If not defined, the default character set of the locale is used. Generally, most western characters can be used if this environment variable is set to **ISO_1**, which is an alias for the **ISO88959-1** character set. For information about supported character sets, see your Microsoft SQL Server documentation.

DTX_TEMPDB

By default, the database adapter creates temporary stored procedures in the tempdb database. If you decide to use this default database, your database administrator should create a SQL script to be executed upon server startup, granting create and execute procedure permissions. If you decide to change this default database, you can set the DTX_TEMPDB environment variable to another database. For example, if you set it to mytmpdb, the database adapter will create temporary stored procedures in mytmpdb.

DTX_SQLSVR_CONN_LIMIT

By default, Microsoft SQL Server allows twenty-five simultaneous connections. If this is insufficient because of the need to simultaneously execute a large number of maps, the limit can be increased by defining the DTX_SQLSVR_CONN_LIMIT environment variable. For example, if you set it to 100 (DTX_SQLSVR_CONN_LIMIT=100), you would allow up to 100 connections.

Database columns and types

The Microsoft SQL Server Adapter is used with the Database Interface Designer to generate type trees for queries, tables, views, and stored procedures in a Microsoft SQL Server relational database management system (RDBMS). Item types are created in a type tree that represent the data types of the columns of a query, table, view, or stored procedure.

The Database Interface Designer gets information about columns by calling Microsoft SQL Server to describe the columns associated with a query, table, view, or stored procedure. Microsoft SQL Server returns the data type, length and other information to the Database Interface Designer. The Microsoft SQL Server data types are then mapped to types in a type tree.

- [Item type properties](#)
- [Date and time formats](#)

Item type properties

The following table lists the Microsoft SQL Server data types and the values of the item type properties to which they correspond when the type tree is generated.

Microsoft SQL Server Data Type	Interpret as	Item Subclass, Presentation	Length
BINARY	Binary	Text	*
CHAR	Character	Text	*
DATETIME	Character	Date & Time	23
DECIMAL(n) ¹	Character	Number, Integer	*
DECIMAL(n,m) ²	Character	Number, Decimal	*
FLOAT	Binary	Number, Float	8
IMAGE	Binary	Text	*
INT	Character	Number, Integer	11
MONEY	Character	Number, Decimal	17
NUMERIC(n) ¹	Character	Number, Integer	*
NUMERIC(n,m) ²	Character	Number, Decimal	*
REAL	Binary	Number, Float	4
SMALLDATETIME	Character	Date & Time	19
SMALLINT	Character	Number, Integer	6
SMALLMONEY	Character	Number, Decimal	17
TEXT	Character	Text	*
TIMESTAMP	Binary	Text	8
TINYINT	Character	Number, Integer	4
VARBINARY	Binary	Text	*
VARCHAR	Character	Text	*

¹ n ≥ 1

² n ≥ 1, m ≥ 0

* The DBMS dictates the length of this type.

Date and time formats

When using columns defined as **datetime** or **smalldatetime** in Microsoft SQL Server databases, the format for both input and output is:

datetime:

ccyy-mm-dd hh:mm:ss.fff

smalldatetime:

ccyy-mm-dd hh:mm:ss

where

cc	=	a two-year century
yy	=	a two-digit year
mm	=	a two-digit month
dd	=	a two-digit day
hh	=	a two-digit hour
mm	=	a two-digit minute
ss	=	a two-digit second
fff	=	fractional seconds

An example is 2001-08-27 00:00:00.000 which specifies August 27, 2001.

In the Generate Type Tree from dialog box, use the **Represent date/time columns as text items** check box to define whether to automatically format this information as Date & Time (which is with this check box disabled, the result of which is shown in the **Item Subclass, Presentation** column above) as a text string. When generated as a text string, it might be necessary to use either the **TEXTTODATE** or **TEXTTOTIME** function in a map rule to convert the text string to the date and time format required by the database. If you are generating new type trees, it is recommended that you disable this check box.

This **Represent date/time columns as text items** check box is modal. After it has been disabled, it will remain disabled for all subsequent type tree generations, regardless of source. Therefore, you must be careful in determining this setting.

Database Interface Designer settings

When you define a Microsoft SQL Server database in the Database Interface Designer, in addition to the common settings available for all the database adapters in the **Database** Definition window, you must enter the information specific to Microsoft SQL Server.

- [Database definition dialog](#)
- [Stored procedures native call syntax](#)

Database definition dialog

The Microsoft SQL Server Adapter-specific settings are described in the following table.

Setting			Description
Security			
	User ID		The user ID used to connect to the database
	Password		The authorization password to connect to the database
Data Source			
	Database Interface Designer		
		Server	The name of the Microsoft SQL Server that hosts the database to be accessed by the Database Interface Designer for design-time purposes
		Database	The name of the database to be accessed by the Database Interface Designer for design-time purposes
	Runtime		
		Server	The name of the Microsoft SQL Server that hosts the database to be accessed for runtime (map execution) purposes both from the Map Designer and a Command Server
		Database	The name of the database to be accessed for runtime (map execution) purposes both from the Map Designer and a Command Server

To specify a server, instance, and database combination (for example, to specify a Microsoft SQL Server 2005 database), enter *server_name\instance_name* in the Server field and *database_name* in the Database field.

Stored procedures native call syntax

A stored procedure can be accessed from within a map by specifying the native call syntax in the following:

- a query that is specified in an input card
- the first argument in a DBQUERY or DBLOOKUP function
This argument does not need to be a literal. The arguments to the stored procedure might be determined at map execution time.
- the SQL Statement adapter command (-STMT) in DBQUERY, DBLOOKUP, or GET functions

For information about using DBLOOKUP or DBQUERY, see database functions in the *Database Interface Designer* documentation. For information about using the syntax for device-independent calls to access return values and output adapter commands, see stored procedures in the *Database Interface Designer* documentation.

For example, a call to a stored procedure using Microsoft SQL Server in a DBLOOKUP might be:

```
DBLOOKUP      ("exec MyAddNameProc(' " + Name:Column + " ',-1)",
    "mydb.mdg",
    "MyDB")
```

String literals must be contained within single quotation marks. Adapter arguments must be separated by commas.

Microsoft SQL Server Adapter commands

This documentation describes the functions and usage of the Microsoft SQL Server Adapter commands and their options.

- [Adapter command summary](#)
- [Adapter commands for a source](#)
- [Adapter commands for a target](#)

Adapter command summary

The following is a summary of the adapter commands that can be used to specify data sources and targets. The applicability of many of the commands depends upon whether you are specifying a source or target, whether a database or query file (.mdq) is used, and the situations in which the usage of the command applies. Adapter commands can be used in GET,>Source,>Command or PUT,>Target,>Command settings in the Map Designer and Integration Flow Designer, using GET, PUT, DBLOOKUP, or DBQUERY function calls, or overriding a data source or target using execution commands in a RUN function or on the command line.

- [Adapter-specific command list](#)

Adapter-specific command list

The following sections describe those adapter commands that are Microsoft SQL Server-specific database parameters. For a complete listing of all adapter commands, see the *Resource Adapters* documentation.

- Database Adapter Type (-DBTYPE)
- Data Source Adapter Command (-SOURCE)
- Trigger Adapter Command (-TR or -TRIG)
- [Database Adapter Type \(-DBTYPE\)](#)
- [Data Source adapter command \(-SOURCE\)](#)
- [Trigger adapter command \(-TR or -TRIG\)](#)

Database Adapter Type (-DBTYPE)

Use the Database Adapter Type adapter command (-DBTYPE) to specify the database adapter type.

```
-DBTYPE SQLSVR
```

This command must be specified if the original card is not a database and no database is specified using the Database/Query adapter command (-MDQ) and the Database Name adapter command (-DBNAME).

Option	Description
SQLSVR	The database adapter type is Microsoft SQL Server.

Data Source adapter command (-SOURCE)

Use the Data Source adapter command (-SOURCE) to specify the Microsoft SQL Server data source.

```
-SOURCE server\\database
```

Option	Description
server\database	Specify the Microsoft SQL Server data source.

Trigger adapter command (-TR or -TRIG)

Use the Trigger adapter command (-TR or -TRIG) to specify the trigger string containing the options defined for a trigger specification.

-TRIG trigger_string

Option	Description
trigger_string	Specify the trigger string containing the options defined for a trigger specification.

Adapter commands for a source

This summary shows the syntax of the adapter commands that can be used when defining a data source using an **.mdq** file or without using one, including both the required and optional adapter commands in the following situations:

- using the **GET>Source>Command** setting in the Map Designer and Integration Flow Designer
- overriding a data source using the Input Source Override - Database execution command (-ID) using a **RUN** function or on the command line
- using a **DBLOOKUP**, **DBQUERY**, or **GET** function in map or component rules
- [GET>Source>Command setting](#)
- [-ID execution command](#)
- [DBLOOKUP or DBQUERY functions](#)
- [GET Function](#)

GET>Source>Command setting

Use the Map Designer or Integration Flow Designer to specify **Database** as the value for the **GET>Source** setting and enter the database adapter commands as desired for the **Command** setting.

With Database/Query File	Without Database/Query File
[-DBTYPE SQLSVR]	-DBTYPE SQLSVR
[-SOURCE server\database]	-SOURCE server\database
[-STMT SQL_statement]	-STMT SQL_statement
[-FILE [directory]]	[-FILE [directory]]
[-VAR name=value...]	[-VAR name=value...]
[-TRIG trigger_string]	[-TRIG trigger_string]
[-USER user_ID]	[-USER user_ID]
[-PASSWORD password]	[-PASSWORD password]
[-CSTMT [number]]	[-CSTMT [number]]
[-GTX]	[-GTX]
[-AUDIT[G][+] [full_path]]	[-AUDIT[G][+] [full_path]]
[[-TRACE]-TRACEERR[+][full_path]]	[[-TRACE]-TRACEERR[+][full_path]]

-ID execution command

Use the Input Source Override - Database execution command (-ID) to designate a database as the source or you can override one or more of the adapter command settings or database definitions in a **RUN** function or on the command line.

Adapter commands are shown using a single quotation mark, which is the Windows syntax. For non-Windows platforms, use two single quotation marks followed by one single quotation mark and end with one single quotation mark followed by two single quotation marks.

- [The source in the compiled map is a database](#)
- [The source in the compiled map is not a database](#)

The source in the compiled map is a database

With Database/Query File:

```
' [-MDQ mdq_file -DBNAME database_name]
 [-QUERY query_name|-STMT SQL_stmt]
 [-FILE [directory]]
 [-VAR name=value...]
```

```

[-TRIG trigger_string]
[-USER username]
[-PASSWORD password]
[-CCARD|-CSTMT [number]]
[-GTX]
[-AUDIT[G] [+][full_path]]
[{-TRACE|-TRACEERR} [+][full_path]]'

```

Without Database/Query File:

```

'-DBTYPE SQLSVR
-SOURCE server\\database
-STM SQL statement
[-FILE [directory]]
[-VAR name=value...]
[-TRIG trigger_string]
[-USER user_ID]
[-PASSWORD password]
[-CCARD|-CSTMT [number]]
[-GTX]
[-AUDIT[G] [+][full_path]]
[{-TRACE|-TRACEERR} [+][full_path]]'

```

The source in the compiled map is not a database

With Database/Query File

```

'-MDQ mdq_file
-DBNAME database_name
-QUERY query_name|-STM SQL_stmt
[-FILE [directory]]
[-VAR name=value...]
[-TRIG trigger_string]
[-USER username]
[-PASSWORD password]
[-CCARD|-CSTMT [number]]
[-GTX]
[-AUDIT[G] [+][full_path]]
[{-TRACE|-TRACEERR} [+][full_path]]'

```

Without Database/Query File

```

'-DBTYPE SQLSVR
-SOURCE server\\database
-STM SQL statement
[-FILE [directory]]
[-TRIG trigger_string]
[-USER user_ID]
[-PASSWORD password]
[-CCARD|-CSTMT [number]]
[-GTX]
[-AUDIT[G] [+][full_path]]
[{-TRACE|-TRACEERR} [+][full_path]]'

```

DBLOOKUP or DBQUERY functions

The DBLOOKUP and DBQUERY functions can be used in component rules in the Type Designer and map rules in the Map Designer when creating a map that can be used with a database.

With Database/Query File	Without Database/Query File
DBLOOKUP ("SQL_statement", "-MDQ mdq_file -DBNAME database_name [-USER username] [-PASSWORD password] [-CCARD -CSTMT [number]] [-GTX] [-AUDIT[G][+][full_path]] [{-TRACE -TRACEERR}[+][full_path]]")	DBQUERY ("SQL_statement", "-DBTYPE SQLSVR -SOURCE server\\database [-USER username] [-PASSWORD password] [-CCARD -CSTMT [number]] [-GTX] [-AUDIT[G][+][full_path]] [{-TRACE -TRACEERR}[+][full_path]]")

GET Function

The GET function returns the data from the source adapter.

With Database/Query File	Without Database/Query File
--------------------------	-----------------------------

With Database/Query File	Without Database/Query File
<pre>GET ("DB", "-MDQ mdq_file -DBNAME database_name -QUERY query_name -STMT SQL_stmt [-FILE [directory]] [-VAR name=value...] [-USER username] [-PASSWORD password] [-CCARD -CSTM [number]] [-GTX] [-AUDIT[G][+] [full_path]] [{-TRACE -TRACEERR}[+] [full_path]]")</pre>	<pre>GET "DB", "-DBTYPE SQLSVR -SOURCE server\database -STMT SQL_stmt [-FILE [directory]] [-USER username] [-PASSWORD password] [-CCARD -CSTM [number]] [-GTX] [-AUDIT[G][+] [full_path]] [{-TRACE -TRACEERR}[+] [full_path]]")</pre>

Adapter commands for a target

This summary shows the syntax of the adapter commands that can be used when defining a data target using an **.mdq** file or without using one, including both the required and optional adapter commands in the following situations:

- using the **PUT>Target>Command setting** in the Map Designer and Integration Flow Designer
- overriding a data source using the Output Source Override - Database execution command (**-OD**) using a **RUN** function or on the command line
- using the **PUT** function in map or component rules
- [**PUT>Target>Command setting**](#)
- [**-OD execution command**](#)
- [**PUT function**](#)

PUT> Target> Command setting

Use the Map Designer or the Integration Flow Designer to specify **Database** as the value for the **PUT>Target** setting and enter the adapter commands for the **Command** setting.

With Database/Query File	Without Database/Query File
<pre>[-DBTYPE SQLSVR] [-SOURCE server\database] [-PROC procedure_name -TABLE table_name] [-USER user_ID] [-PASSWORD password] [-CSTM [number]] [-DELETE] [-UPDATE [OFF ONLY]] [-BADDATA[+] full_path] [-GTX] [-AUDIT[G][+] [full_path]] [{-TRACE -TRACEERR}[+] [full_path]]</pre>	<pre>-DBTYPE SQLSVR -SOURCE server\database -PROC procedure_name -TABLE table_name [-USER user_ID] [-PASSWORD password] [-CSTM [number]] [-DELETE] [-UPDATE [OFF ONLY]] [-BADDATA[+] full_path] [-GTX] [-AUDIT[G][+] [full_path]] [{-TRACE -TRACEERR}[+] [full_path]]</pre>

-OD execution command

Use the Output Source Override - Database execution command (**-OD**) to designate a database as a target or you can override one or more of the adapter command settings or database definitions in a **RUN** function or on the command line.

The adapter commands are shown using a single quotation mark, which is the Windows syntax. For non-Windows platforms, use two single quotation marks followed by one single quotation mark and end with one single quotation mark followed by two single quotation marks.

- [**The target in the compiled map is a database**](#)
- [**The target in the compiled map is not a database**](#)

The target in the compiled map is a database

With Database/Query File

```
' [-MDQ mdq_file -DBNAME database_name]
[-PROC procedure_name|-TABLE table_name]
[-USER username]
[-PASSWORD password]
[-CCARD|-CSTM [number]]
[-DELETE]
[-UPDATE [OFF|ONLY]]
[-BADDATA[+] full_path]
[-GTX]
```

```
[-AUDIT[G] [+][full_path]
[{-TRACE|-TRACEERR} [+][full_path]]'
```

Without Database/Query File

```
'-DBTYPE SQLSVR
-SOURCE server\\database
-PROC procedure_name|-TABLE table_name
[-USER username]
[-PASSWORD password]
[-CCARD|-CSTMT [number]]
[-DELETE]
[-UPDATE [OFF|ONLY]]
[-BADDATA[+][full_path]
[-GTX]
[-AUDIT[G] [+][full_path]
[{-TRACE|-TRACEERR} [+][full_path]]'
```

The target in the compiled map is not a database

With Database/Query File

```
'-MDQ mdq_file
-DBNAME database_name
-PROC procedure_name|-TABLE table_name
[-USER username]
[-PASSWORD password]
[-CCARD|-CSTMT [number]]
[-DELETE]
[-UPDATE [OFF|ONLY]]
[-BADDATA[+][full_path]
[-GTX]
[-AUDIT[G] [+][full_path]
[{-TRACE|-TRACEERR} [+][full_path]]'
```

Without Database/Query File

```
'-DBTYPE SQLSVR
-SOURCE server\\database
-PROC procedure_name|-TABLE table_name
[-USER username]
[-PASSWORD password]
[-CCARD|-CSTMT [number]]
[-DELETE]
[-UPDATE [OFF|ONLY]]
[-BADDATA[+][full_path]
[-GTX]
[-AUDIT[G] [+][full_path]
[{-TRACE|-TRACEERR} [+][full_path]]'
```

PUT function

Use the `PUT` function to pass data to the target adapter.

With Database/Query File	Without Database/Query File
<code>PUT ("DB", "-MDQ mdq_file -DBNAME database_name -PROC procedure_name -TABLE table_name [-USER username] [-PASSWORD password] [-CCARD -CSTMT [number]] [-DELETE] [-UPDATE [OFF ONLY]] [-BADDATA[+][full_path] [-GTX] [-AUDIT[G][+][full_path]] [{-TRACE -TRACEERR}[+][full_path]]")</code>	<code>PUT ("DB", "-DBTYPE SQLSVR -SOURCE server\\database -PROC procedure_name -TABLE table_name [-USER username] [-PASSWORD password] [-CCARD -CSTMT [number]] [-DELETE] [-GTX] [-BADDATA[+][full_path] [-AUDIT[G][+][full_path] [{-TRACE -TRACEERR}[+][full_path]]")</code>

Extensions for the Launcher

Microsoft SQL Server database sources can be specified as input events for the Launcher. Insertions or updates to database tables can be the trigger mechanisms to execute a map.

If you are running the Launcher on Windows and want to use Microsoft SQL Server database sources as input events to execute a map, you will need to install the Microsoft SQL Server Extensions on your database host system.

- [Installing Microsoft SQL Server Extensions](#)

Installing Microsoft SQL Server Extensions

The Microsoft SQL Server Extensions are installed as one of the database components of the Database Interface Designer using the Design Studio installation program.

It is not necessary to install the Database Interface Designer or the IBM Transformation Extender Launcher on the same system hosting your Microsoft SQL Server database. The database adapter must be installed with the Database Interface Designer and also with the Launcher. The Microsoft SQL Server Extensions must be installed on the system hosting your Microsoft SQL Server database.

- [Adding extended stored procedures](#)

Adding extended stored procedures

To use the Microsoft SQL Server triggers, you must have your system administrator install and run the triggering installation script for Microsoft SQL Server: **m4sqlsvr.sql**. For more information about this procedure, see database triggers in the *Database Interface Designer* documentation.

Bulk copy example files

The bulk copy function transfers data between a table and an operation system file at high speeds. It is the most common way to load large amounts of data into a database table.

It is frequently used to transfer data between database vendors, spreadsheet applications and servers.

The example files are located in the following directory: *install_dir\examples\adapters\mssql*.

Restrictions and limitations

The Database Interface Designer and database adapter offer options and functions for accessing and manipulating data contained within a database. However, there is one restriction and limitation of a certain function when using the Microsoft SQL Server Adapter. This function is:

- **Bind Facility**
Using bind values in database functions is not supported.

Return codes and error messages

Return codes and messages are returned when the particular activity completes. Return codes and messages might also be recorded as specified in the audit logs, trace files, execution summary files, and so on.

For information about error codes and messages returned by database-specific adapters, see Return Codes and Error Messages> Database-specific Adapter Messages in the *Resource Adapters* documentation.

Various troubleshooting tools are available if you encounter problems while using the database adapters. For example, if you attempt to run a map that uses the adapter and encounter problems or do not receive the expected results, use the following adapter troubleshooting tools:

- adapter audit log (.log)
- adapter trace file (.mtr)

OData Adapter

The OData (Open Data Protocol) adapter provides support for performing read and write operations on data sources that expose OData interface. OData adapter is an ISO/IEC approved, OASIS standard that defines a set of best practices for building and consuming RESTful APIs.

OData adapter provides support for connecting to a compliant OData service, discovering entity sets exposed by the services and importing entity definitions in form of native JSON schemas. The adapter further supports fetching entities in managed mode, where the entity data complies with the imported schemas, as well as in raw mode where the OData service is accessed directly and the exchanged data is not constrained and validated by the adapter-provided schema. The adapter also supports inserting new entities into the specified entity set.

- [Authenticating connections](#)
The adapter supports Basic HTTP authentication, with the user provided username and password.
- [OData Adapter properties and commands](#)
- [GET function example](#)
- [PUT function example](#)

Authenticating connections

The adapter supports Basic HTTP authentication, with the user provided username and password.

OData adapter as a source

OData Adapter as a source supports:

- Filtering, projection, ordering and limiting of the fetched results
- Setting custom HTTP request headers
- Specifying data format

OData adapter as a target

ODATA adapter as a target supports:

- Inserting entities into the specified entity set
- Setting custom HTTP request headers
- Specifying data format

OData Adapter properties and commands

- **Root URL (-RU or -ROOTURL)**
Specifies the service root URL of the OData service. This is a mandatory property.
- **Authentication Type (-AT or -AUTHTYPE)**
Type of authentication to use for accessing the OData service. This is a mandatory property.
- **OData Version**
Specifies the version for OData adapter.
- **Username**
Specifies the username for the requests, when using Basic authentication method.
- **Password**
Specifies the password for the requests, when using Basic authentication method.
- **Entity Set**
Specifies the entity set in the service on which to perform the data operation. This is a mandatory property.
- **Headers**
Specifies custom request headers. This is an optional property.
- **Format**
Specifies the media type format of the data.
- **Filter**
Specifies the filter expression to apply when fetching the data.
- **Select**
Selects the required field. This is an optional property.
- **Top**
Specifies the limit for how many values to fetch. This is an optional property. The corresponding adapter command is -TOP.
- **Skip**
This property specifies the number of values to skip from the beginning. This is an optional property.
- **Order By**
Specifies an expression for determining what values are used to order the collection of entries. This is an optional property.
- **Read Mode**
Specifies the read mode of operation. Default is Managed. If Read Mode set to Managed, it generates schema directly.
- **Logging**
Specifies the level of logging to use for the log (trace) file produced by the adapter. The default is false.
- **Append Log**
Flag indicating what to do if the specified log file already exists.
- **Log File Path**
This is the location of the log file to which to write log messages.

Root URL (-RU or -ROOTURL)

Specifies the service root URL of the OData service. This is a mandatory property.

The corresponding adapter command is -RU (or -ROOTURL).

Authentication Type (-AT or -AUTHTYPE)

Type of authentication to use for accessing the OData service. This is a mandatory property.

There are two types of authentication:

- None: No authentication is performed.
- Basic: Basic HTTP authentication with username and password is performed.

The corresponding adapter command is -AT (or -AUTHTYPE).

OData Version

Specifies the version for OData adapter.

There are two versions for OData adapter: V4 and V2. The default is V4.

The corresponding adapter command is -OV.

Username

Specifies the username for the requests, when using Basic authentication method.

The corresponding adapter command is -U (or -USERNAME).

Password

Specifies the password for the requests, when using Basic authentication method.

The corresponding adapter command is -P (or -PASSWORD).

Entity Set

Specifies the entity set in the service on which to perform the data operation. This is a mandatory property.

The corresponding adapter command is -ES (or -ENTITYSET).

Headers

Specifies custom request headers. This is an optional property.

Each header is specified as name, value pair in the format name=value, and multiple headers are separated by a space character, for example: header1=value1 header2=value2 header3=value3. The corresponding adapter command is -H (or -HEADERS).

Format

Specifies the media type format of the data.

The corresponding adapter command is -FMT (or -FORMAT).

For additional information, see the OData website.

Filter

Specifies the filter expression to apply when fetching the data.

Filters the data as per your requirements. This is an optional property. The corresponding adapter command is -FIL (or -FILTER).

For more information, see the OData website.

Select

Selects the required field. This is an optional property.

Specifies the field selection expression for fetching the data. This is an optional property.

The corresponding adapter command is -SEL (or -SELECT).

Top

Specifies the limit for how many values to fetch. This is an optional property. The corresponding adapter command is -TOP.

For more information, see the OData website.

Skip

This property specifies the number of values to skip from the beginning. This is an optional property.

The corresponding adapter command is -SKP (or -SKIP).

For more information, see the OData website.

Order By

Specifies an expression for determining what values are used to order the collection of entries. This is an optional property.

The corresponding adapter command is -ORD (or -ORDERBY).

For more information, see the OData website.

Read Mode

Specifies the read mode of operation. Default is Managed. If Read Mode set to Managed, it generates schema directly.

In **Managed** mode, the imported JSON schema reflects the structure of the entities of the specified entity set. When fetching entities, only the properties captured in the schema are fetched and returned by the adapter.

In **Raw** mode, the imported schema contains just a plain text item. When fetching entities, the adapter returns the raw data returned by the service in unmodified form. To access individual properties and other pieces of information in the returned data, the plain text item would need to be manually replaced with a schema type that corresponds to the structure of the returned data.

Logging

Specifies the level of logging to use for the log (trace) file produced by the adapter. The default is false.

The value Information means log informational, the value Errors Only means log error messages only, and the value Verbose means log debug and trace level messages along with the informational and error messages.

Append Log

Flag indicating what to do if the specified log file already exists.

When set to true, the log messages are appended to the file. When set to false, the file is truncated, and the messages are written to the empty file. The default value is true.

Log File Path

This is the location of the log file to which to write log messages.

If not specified, the default log file name m4odata.mtr is used, and the file is stored to the directory in which the executed compiled map resides.

GET function example

In the following example, the GET map function is used to retrieve entities from the specified entity set, in managed mode, and with no authentication:

```
GET("ODATA", "-RU rootURL -AT none -RM managed -ES entitySetName")
```

PUT function example

In the following example, the PUT map function is used to insert a new entity (represented by the *entityPayload* data) to the specified entity set:

ODBC Adapter

You can use the Open DataBase Connectivity (ODBC) Adapter with a Command Server, Launcher, the Software Development Kit, or with a map in a map rule.

- [Overview](#)
- [System requirements](#)
- [Database columns and types](#)
- [Database Interface Designer settings](#)
- [ODBC Adapter commands](#)
- [Overriding column attributes](#)
- [Binding values in DBLOOKUP/DBQUERY](#)
- [Configuring DSN for Launcher startup](#)
- [Restrictions and limitations](#)
- [Return codes and error messages](#)

Overview

Use the ODBC Adapters to access and manipulate data contained in databases that are ODBC data sources. You can also install database adapters on additional systems for remote database connectivity.

This documentation discusses the requirements for installing and running the ODBC Adapters and provides the database-specific information required for using your specific adapters in your database host environment.

System requirements

For the latest information about how to install the drivers and configure your ODBC data sources, see IBM Transformation Extender product release notes and the ODBC documentation.

Database columns and types

The Database Interface Designer and **mtsmaker** with the Type Tree Maker generate type trees for queries, tables, views, and stored procedures in an ODBC-compliant RDBMS (Relational Database Management System). Item types are created in a type tree that represents the data types of the columns of a query, table, view, or stored procedure.

The Database Interface Designer and **mtsmaker** get information about columns by calling the ODBC driver for your RDBMS to describe the columns associated with a query, table, view, or stored procedure. The ODBC driver for the RDBMS has mapped the data types in each column to ODBC data types as appropriate. From the RDBMS, the ODBC Driver Manager returns the ODBC data type, length, and other information to the Database Interface Designer and **mtsmaker**. The ODBC data types are then mapped to types in a type tree. The Database Interface Designer and **mtsmaker** have no knowledge of the RDBMS-specific data types-only the ODBC data types.

- [Item type properties](#)
- [Date and time formats](#)

Item type properties

The following table lists the ODBC data types and the values of the item type properties to which they correspond when the type tree is generated.

ODBC Data Type	Interpret as	Item Subclass, Presentation	Length
SQL_VARCHAR	Character	Text	*
SQL_CHAR	Character	Text	*
SQL_DECIMAL	Character	Number, Decimal	*
SQL_NUMERIC	Character	Number, Decimal	*
SQL_SMALLINT	Character	Number, Integer	6
SQL_INTEGER	Character	Number, Integer	11
SQL_REAL	Binary	Number, Float	4
SQL_FLOAT	Binary	Number, Float	8
SQL_DOUBLE	Binary	Number, Float	8
SQL_LONGVARCHAR	Character	Text	*
SQL_BIT	Character	Number, Integer	*
SQL_TINYINT	Character	Number, Integer	4
SQL_BIGINT	Character	Number, Integer	20
SQL_BINARY	Binary	Text	*
SQL_VARBINARY	Binary	Text	*
SQL_LONGVARBINARY	Binary	Text	*

ODBC Data Type	Interpret as	Item Subclass, Presentation	Length
SQL_DATE	Character	Date & Time	*
SQL_TIME	Character	Date & Time	*
SQL_TIMESTAMP	Character	Date & Time	*
SQL_INTERVAL_HOUR	Character	Text	*
SQL_INTERVAL_MINUTE	Character	Text	*
SQL_INTERVAL_SECOND	Character	Text	*
SQL_INTERVAL_YEAR_TO_MONTH	Character	Text	*
SQL_INTERVAL_DAY_TO_HOUR	Character	Text	*
SQL_INTERVAL_DAY_TO_MINUTE	Character	Text	*
SQL_INTERVAL_DAY_TO_SECOND	Character	Text	*
SQL_INTERVAL_HOUR_TO_MINUTE	Character	Text	*
SQL_INTERVAL_HOUR_TO_SECOND	Character	Text	*
SQL_INTERVAL_MINUTE_TO_SECOND	Character	Text	*

* The DBMS dictates the length of this type.

Date and time formats

Precision can be specified on output. For example, a precision of 4 would change a date of format from ccyy-mm-dd to ccyy.

When using columns defined as dates and times in ODBC databases, the format for both input and output are:

date:

ccyy-mm-dd

time:

hh:mm:ss

timestamp:

ccyy-mm-dd hh:mm:ss[.ffff...]

If the group format is fixed, the field is padded to 26 with trailing spaces.

where

cc	=	a two-digit century
yy	=	a two-digit year
mm	=	a two-digit month
dd	=	a two-digit day
hh	=	a two-digit hour
mm	=	a two-digit minute
ss	=	a two-digit second
.ffff...	=	optional fractional seconds

An example is 2001-08-27 00:00:00 which specifies August 27, 2001.

In the Generate Type Tree from dialog, use the **Represent date/time columns as text items** check box to define whether to automatically format this information as Date & Time as a text string. When generated as a text string, it might be necessary to use either the TEXTTODATE or TEXTTOTIME function in a map rule to convert the text string to the date and time format required by the database. If you are generating new type trees, it is recommended that you disable this check box.

The **Represent date/time columns as text items** check box is modal. After it has been disabled, it will remain disabled for all subsequent type tree generations, regardless of source. Therefore, you must be careful in determining this setting.

Database Interface Designer settings

When you define an ODBC database in the Database Interface Designer, in addition to the common settings available for all of the database adapters in the Database Definition dialog, you need to enter the information specific to ODBC.

- [Database definition dialog](#)
- [Stored procedures native call syntax](#)

Database definition dialog

The ODBC Adapter-specific settings are described in the following table:

Setting	Description
Data Source	
Database Interface Designer	The data source you defined in your development PC that is used by the Database Interface Designer to access the database information for design-time purposes
Runtime	The data source you defined in your ODBC database host platform to be used for access to the database for runtime (map execution) purposes both from the Map Designer and a Command Server
Security	
User ID	The user ID used to connect to the database
Password	The authorization password to connect to the DB2® subsystem

Stored procedures native call syntax

A stored procedure can be accessed from within a map by specifying the native call syntax in the following:

- a query that is specified in an input card
- the first argument in a DBQUERY or DBLOOKUP function
This argument does not need to be a literal. The arguments to the stored procedure might be determined at map execution time.
- the SQL Statement adapter command (-SMT) in DBQUERY, DBLOOKUP, or GET functions

For information about using DBLOOKUP or DBQUERY, or about using the syntax for device-independent calls to access return values and output adapter commands, see the *Database Interface Designer* documentation.

For example, a call to a stored procedure using ODBC in a DBLOOKUP might be:

```
DBLOOKUP      ("{call MyAddNameProc('' + Name:Column + '' , -1)}",
                "mydb.mdq",
                "MyDB")
```

String literals must be contained within single quotation marks; adapter arguments must be separated by commas.

ODBC Adapter commands

- [Adapter command summary](#)
- [Adapter commands for a source](#)
- [Adapter commands for a target](#)

Adapter command summary

The following is a summary of the adapter commands that can be used to specify data sources and targets. The applicability of many of the commands depends upon whether you are specifying a source or target, whether a database or query file (.mdq) is used, and how the command is being used. Adapter commands can be used in GET->Source->Command or PUT->Target->Command settings in the Map Designer and Integration Flow Designer, using GET, PUT, DBLOOKUP, or DBQUERY function calls, or overriding a data source or target using execution commands in a RUN function or on the command line.

- [Adapter specific commands](#)

Adapter specific commands

This section lists and describes those adapter commands that are ODBC specific database parameters. For a complete listing of all database adapter commands, see the *Resource Adapters* documentation.

- [Database Adapter Type \(-DBTYPE\)](#)
- [Row Count \(-ROWCNT\)](#)
- [Data Source \(-SOURCE\)](#)
- [Database Adapter Type \(-DBTYPE\)](#)
- [Row Count \(-ROWCNT\)](#)
- [Data Source \(-SOURCE\)](#)

Database Adapter Type (-DBTYPE)

Use the Database Adapter Type adapter command (-DBTYPE) to specify the database adapter type.

-DBTYPE ODBC

Option	Description
ODBC	<p>The database adapter type is ODBC.</p> <p>This command must be specified if the original card is not a database and no database is specified using the Database/Query adapter command (-MDQ) and the Database Name adapter command (-DBNAME).</p>

Row Count (-ROWCNT)

Use the Row Count adapter command (-ROWCNT) to specify the number of rows to be retrieved per fetch.

-ROWCNT row_count

Option	Description
<i>row_count</i>	<p>This is the number of rows to be retrieved per fetch.</p> <p>The time to fetch is optimized in accordance with the increase in the number of rows to retrieve. Therefore, a significant performance improvement might be gained by retrieving as many rows as possible in a single fetch.</p> <p>The number of rows retrieved is limited only by the amount of memory installed on the querying PC.</p> <p>The default for the number of rows to be retrieved per fetch for the ODBC Adapter is one row.</p>

Data Source (-SOURCE)

Use the Data Source adapter command (-SOURCE) to specify the ODBC data source.

-SOURCE datasource

Option	Description
<i>datasource</i>	Specify the ODBC data source.

Adapter commands for a source

This summary shows the syntax of the adapter commands that can be used when defining a data source using an .mdq file or without using an .mdq file, including both the required and optional adapter commands in the following situations:

- using a GET>Source>Command setting in the Map Designer and Integration Flow Designer
- overriding a data source using the Input Source Override - Database execution command (-ID) using a RUN function or on the command line
- using a DBLOOKUP, DBQUERY, or GET function in map or component rules
- [GET> Source> Command setting](#)
- [-ID Execution command](#)
- [DBLOOKUP or DBQUERY functions](#)
- [GET function](#)

GET> Source> Command setting

Use the Map Designer or Integration Flow Designer to specify **Database** as the value for the GET>Source setting and enter the adapter commands for the **Command** setting.

With Database/Query File	Without Database/Query File
[-DBTYPE ODBC]	-DBTYPE ODBC
[-SOURCE <i>datasource</i>]	-SOURCE <i>datasource</i>
[-STMT <i>SQL_statement</i>]	-STMT <i>SQL_statement</i>
[-FILE [directory]]	[-FILE [directory]]
[-VAR <i>name=value...</i>]	[-VAR <i>name=value...</i>]
[-USER <i>user_ID</i>]	[-USER <i>user_ID</i>]
[-PASSWORD <i>password</i>]	[-PASSWORD <i>password</i>]
[-CSTMT [<i>number</i>]]	[-CSTMT [<i>number</i>]]
[-ROWCNT <i>row_count</i>]	[-ROWCNT <i>row_count</i>]
[-AUDIT[G][+] [<i>full_path</i>]]	[-AUDIT[G][+] [<i>full_path</i>]]
[-TRACE -TRACEERR [:] [<i>full_path</i>]]	[-TRACE -TRACEERR [:] [<i>full_path</i>]]

-ID Execution command

Use the Input Source Override - Database execution command (-ID) to designate a database as the source. Or, you can override one or more of the adapter command settings or database definitions in a RUN function or on the command line.

The following adapter commands are shown using a single quotation mark, which is the Windows syntax. For non-Windows platforms, use two single quotation marks followed by one single quotation mark and end with one single quotation mark followed by two single quotation marks.

- [The source in the compiled map is a database](#)
- [The source in the compiled map is not a database](#)

The source in the compiled map is a database

With Database/Query File

```
'[-MDQ mdq_file -DBNAME database_name]
[-QUERY query_name|-STMT SQL_stmt]
[-FILE [directory]]
[-VAR name=value...]
[-USER username]
[-PASSWORD password]
[-CCARD|-CSTMT [number]]
[-ROWCNT row_count]
[-AUDIT[G] [+| [full_path]]
[(-TRACE|-TRACEERR)[+| [full_path]]]
```

Without Database/Query File

```
'-DBTYPE ODBC
-SOURCE datasource
-STM SQL_statement
[-FILE [directory]]
[-VAR name=value...]
[-USER user_ID]
[-PASSWORD password]
[-CCARD|-CSTMT [number]]
[-ROWCNT row_count]
[-AUDIT[G] [+| [full_path]]
[(-TRACE|-TRACEERR)[+| [full_path]]]
```

The source in the compiled map is not a database

With Database/Query File

```
'-MDQ mdq_file
-DBNAME database_name
-QUERY query_name|-STMT SQL_stmt
[-FILE [directory]]
[-VAR name=value...]
[-USER username]
[-PASSWORD password]
[-CCARD|-CSTMT [number]]
[-ROWCNT row_count]
[-AUDIT[G] [+| [full_path]]
[(-TRACE|-TRACEERR)[+| [full_path]]]
```

Without Database/Query File

```
'-DBTYPE ODBC
-SOURCE datasource
-STM SQL_statement
[-FILE [directory]]
[-USER user_ID]
[-PASSWORD password]
[-CCARD|-CSTMT [number]]
[-ROWCNT row_count]
[-AUDIT[G] [+| [full_path]]
[(-TRACE|-TRACEERR)[+| [full_path]]]
```

DBLOOKUP or DBQUERY functions

The DBLOOKUP and DBQUERY functions can be used in component rules in the Type Designer and map rules in the Map Designer when creating a map that can be used with a database.

With Database/Query File	Without Database/Query File
--	---

With Database/Query File	Without Database/Query File
DBLOOKUP ("SQL_statement", "-MDQ mdq_file -DBNAME database_name [-USER username] [-PASSWORD password] [-CCARD -CSTMT [number]] [-ROWCNT row_count] [-AUDIT[G][+] [full_path]] [{-TRACE -TRACEERR}{+} [full_path]]")	DBQUERY ("SQL_statement", "-DBTYPE ODBC -SOURCE datasource [-USER username] [-PASSWORD password] [-CCARD -CSTMT [number]] [-ROWCNT row_count] [-AUDIT[G][+] [full_path]] [{-TRACE -TRACEERR}{+} [full_path]]")

GET function

The `GET` function returns the data from the source adapter.

With Database/Query File	Without Database/Query File
GET ("DB", "-MDQ mdq_file -DBNAME database_name -QUERY query_name -STMT SQL_stmt [-FILE [directory]] [-VAR name=value...] [-USER username] [-PASSWORD password] [-CCARD -CSTMT [number]] [-ROWCNT row_count] [-AUDIT[G][+] [full_path]] [{-TRACE -TRACEERR}{+} [full_path]]")	GET "DB", "-DBTYPE ODBC -SOURCE datasource -STMT SQL_stmt [-FILE [directory]] [-USER username] [-PASSWORD password] [-ROWCNT row_count] [-CCARD -CSTMT [number]] [-AUDIT[G][+] [full_path]] [{-TRACE -TRACEERR}{+} [full_path]]")

Adapter commands for a target

This summary shows the syntax of the adapter commands that can be used when defining a data target using an `.mdq` file or without using an `.mdq` file, including both the required and optional adapter commands in the following situations:

- using a `PUT>Target>Command` setting in the Map Designer and Integration Flow Designer
- overriding a data source using the Output Source Override - Database execution override (`-OD`) using a `RUN` function or on the command line
- using the `PUT` function in map or component rules
- [PUT> Target> Command setting](#)
- [-OD execution command](#)
- [PUT function](#)

PUT> Target> Command setting

Use the Map Designer or the Integration Flow Designer to specify **Database** as the value for the `PUT>Target` setting and enter the adapter commands for the **Command** setting.

With Database/Query File	Without Database/Query File
[-DBTYPE ODBC] [-SOURCE datasource] [-PROC procedure_name -TABLE table_name] [-USER user_ID] [-PASSWORD password] [-CSTMT [number]] [-DELETE] [-UPDATE [OFF ONLY]] [-BADDATA[+] full_path] [-AUDIT[G][+] [full_path]] [{-TRACE -TRACEERR}{+} [full_path]]	-DBTYPE ODBC -SOURCE datasource -PROC procedure_name -TABLE table_name [-USER user_ID] [-PASSWORD password] [-CSTMT [number]] [-DELETE] [-UPDATE [OFF ONLY]] [-BADDATA[+] full_path] [-AUDIT[G][+] [full_path]] [{-TRACE -TRACEERR}{+} [full_path]]

-OD execution command

Use the Output Source Override - Database execution command (`-OD`) to designate a database as a target or you can override one or more of the adapter command settings or database definitions in a `RUN` function or on the command line.

The following adapter commands are shown using a single quotation mark, which is the Windows syntax. For non-Windows platforms, use two single quotation marks followed by one single quotation mark and end with one single quotation mark followed by two single quotation marks.

- [The target in the compiled map is a database](#)

- [The target in the compiled map is not a database](#)

The target in the compiled map is a database

With Database/Query File

```
'[-MDQ mdq_file -DBNAME database_name]
[-PROC procedure_name|-TABLE table_name]
[-USER username]
[-PASSWORD password]
[-CCARD|-CSTMT [number]]
[-DELETE]
[-UPDATE [OFF|ONLY]]
[-BADDATA[+] full_path]
[-AUDIT[G][+] [full_path]]
[{-TRACE|-TRACEERR}[+] [full_path]]'
```

Without Database/Query File

```
'-DBTYPE ODBC
-SOURCE datasource
-PROC procedure_name|-TABLE table_name
[-USER username]
[-PASSWORD password]
[-CCARD|-CSTMT [number]]
[-DELETE]
[-UPDATE [OFF|ONLY]]
[-BADDATA[+] full_path]
[-AUDIT[G][+] [full_path]]
[{-TRACE|-TRACEERR}[+] [full_path]]'
```

The target in the compiled map is not a database

With Database/Query File

```
'-MDQ mdq_file
-DBNAME database_name
-PROC procedure_name|-TABLE table_name
[-USER username]
[-PASSWORD password]
[-CCARD|-CSTMT [number]]
[-DELETE]
[-UPDATE [OFF|ONLY]]
[-BADDATA[+] full_path]
[-AUDIT[G][+] [full_path]]
[{-TRACE|-TRACEERR}[+] [full_path]]'
```

Without Database/Query File

```
'-DBTYPE ODBC
-SOURCE datasource
-PROC procedure_name|-TABLE table_name
[-USER username]
[-PASSWORD password]
[-CCARD|-CSTMT [number]]
[-DELETE]
[-UPDATE [OFF|ONLY]]
[-BADDATA[+] full_path]
[-AUDIT[G][+] [full_path]]
[{-TRACE|-TRACEERR}[+] [full_path]]'
```

PUT function

Use the `PUT` function to pass data to the target adapter.

With Database/Query File	Without Database/Query File
<pre>PUT ("DB", "-MDQ mdq_file -DBNAME database_name -PROC procedure_name -TABLE table_name [-USER username] [-PASSWORD password] [-CCARD -CSTMT [number]] [-DELETE] [-UPDATE [OFF ONLY]] [-BADDATA[+] full_path] [-AUDIT[G][+] [full_path]] [{-TRACE -TRACEERR}[+] [full_path]]")</pre>	<pre>PUT ("DB", "-DBTYPE ODBC -SOURCE datasource -PROC procedure_name -TABLE table_name [-USER username] [-PASSWORD password] [-CCARD -CSTMT [number]] [-DELETE] [-UPDATE [OFF ONLY]] [-BADDATA[+] full_path] [-AUDIT[G][+] [full_path]] [{-TRACE -TRACEERR}[+] [full_path]]")</pre>

Overriding column attributes

When using the Database Interface Designer to generate a type tree for a query, table, view, or stored procedure, you might want to specify how columns are to be interpreted, rather than relying upon the interpretation of a particular database driver.

These column overrides are saved in the **.mdq** file. When generating a type tree with the ODBC Adapter and enabling the **Override Column Definitions** check box in the Generate Type Tree from dialog for a query, table, view, or stored procedure, the Column Datatype Specification dialog appears. Any overrides you specify in these dialogs are saved when the **.mdq** file is saved.

The generated type tree with overrides is represented in the Navigator with an icon having black spheres.

If a tree is regenerated at a later date for the same query, table, view, or stored procedure within a database, you are prompted to confirm whether the previously defined overrides should be applied to the new tree.

The following steps and dialogs show how to define column override attributes for a table or view. However, the steps and dialogs for a stored procedure are very similar and should follow the same process.

- [Defining column attribute overrides](#)

Defining column attribute overrides

The following procedure describes how to define column attribute overrides for a:

- table or stored procedure
- [To define column definition overrides for a query](#)

To define column attribute overrides for a table or stored procedure

1. In the Database Interface Designer, open the **.mdq** file.
2. In the Navigator, select the database containing the table for which you want to generate a type tree with column attribute overrides.
3. From the Database menu, select either Generate Tree From **> Table** or Generate Tree From **> Procedure**, depending upon the one you are generating. Either the **Generate Type Tree from Tables** or Generate Type Tree from Procedures dialog appears, depending upon your selection. An example of the Generate Type Tree from Tables dialog follows. For more information about the type tree generation dialogs, see the *Database Interface Designer* documentation.
4. Select one or more tables, views, or procedures for which you want to generate a type tree.
5. Enter a name for the type tree file (**.mtt**)
or
6. Enable one of the **Type options** to specify how you want the newly generated types to be saved.
7. From the **Row group format** list, enable **Delimited** or **Fixed**.
8. Accept the default values for the **Group options** fields (**Delimiter**, **Terminator**, and **Release**) or change them.
9. To specify how columns are to be interpreted, select the **Override column definitions** check box.
10. Click **Generate**.
The Column Datatype Specification dialog appears.
11. From the **Tables/Views** list, select the table or view containing the column or columns for which you want to specify an override.
12. From the **Columns** list, select the column for which you want to specify an override.
13. Specify the **Subclass**, **Interpret as**, **Presentation**, and **Length** values for the selected column. For more information about setting these attributes, see the *Type Designer* documentation. In most cases, the specified length should not be shorter than the default length because this might result in data truncation errors.
14. Repeat Steps 11-13 for each column for which you want to specify an override.
15. When finished, click **OK**.

To define column definition overrides for a query

1. In the Database Interface Designer, open the **.mdq** file.
2. In the Navigator, select the query for which you want to generate a type tree.
3. From the **Queries** menu, choose **Generate Tree**.
The Generate Type Tree from Query dialog appears, an example of which follows. For details on the type tree generation dialog, see the *Database Interface Designer* documentation.
4. Enter a name for the type tree file (**.mtt**)
or
5. To specify the way you want to save the newly generated types, enable one of the **Type options** check boxes.
6. From the **Row group format** list, enable either **Delimited** or **Fixed**.
7. Accept the default values for the **Group options** fields (**Delimiter**, **Terminator**, and **Release**) or change them.
8. To specify how columns are to be interpreted, select the **Override column definitions** check box.
9. Click **OK**.
The Column Datatype Specification dialog appears.
10. From the **Columns** list, select the column for which you want to specify an override.
11. Specify the **Subclass**, **Interpret as**, **Presentation**, and **Length** values for the selected column as desired. For more information about setting these attributes, see the *Type Designer* documentation. In most cases, the specified length should not be shorter than the default length because this might result in data truncation errors.
12. Repeat Steps 10-11 for each column for which you want to specify an override.
13. Click **OK**.

Binding values in DBLOOKUP/DBQUERY

When using a `DBLOOKUP` or `DBQUERY` function, use the Bind facility to submit similarly constructed SQL statements to the database server so that the statements are syntactically identical. By binding a value to a placeholder in the SQL statement, the actual syntax of the statement can be made static, which might improve performance. For more information about using bind values in database functions, see the *Database Interface Designer* documentation.

- [Limitation when using ODBC](#)
- [Specifying the data type](#)

Limitation when using ODBC

When binding values using ODBC, a limitation exists because ODBC cannot determine the data type of the column to which the value is being bound at the time the SQL statement is executed. Therefore, the data type of the value being passed must be explicitly specified. Because of this ODBC limitation, the ODBC Adapter determines the data type of the value according to the following rule:

If the value contains only numeric characters or numeric characters with a decimal point, the value is assumed to be numeric. Otherwise, the value is assumed to be text.

This rule provides the expected results in most cases. However, there might be some situations in which the rule might not apply and it might be necessary to specify the exact data type. For example, if the value is a text field consisting solely of numeric characters, the value is incorrectly interpreted as numeric unless correctly specified.

Specifying the data type

To override the default behavior and to explicitly specify the data type, a data type indicator must precede the bind value. The syntax for this is:

```
:bind([T|N|D], value)
```

The following table lists the data type indicators and the relationship between each data type indicator and the ODBC data types.

Data Type Indicator	ODBC data type
T or TEXT	SQL_CHAR
N or NUM or NUMBER	SQL_NUMERIC
D or DATE	SQL_DATETIME

For information about item formats and interpretation, see [Item Type Properties](#).

As an example of specifying a data type, to bind a date value to a statement, you must specify that the value be of data type DATETIME. The SQL statement would be entered as:

```
DBLOOKUP("select Age from Person where DateOfBirth=:bind([DATE] , " +
          DateItem:Row + ")",
          "DB.mdg",
          "MyDB")
```

If the `[DATE]` data type indicator is not specified, the trace file from the adapter displays the following error message:

```
Message: [Microsoft][ODBC Microsoft Access 97 Driver]
        Data type mismatch in criteria expression.
```

Configuring DSN for Launcher startup

This information also applies to establishing a Windows-based data source name for the Informix® adapter. For information about how to configure a DSN for UNIX, see your UNIX documentation.

Using ODBC, data sources are configured in the **ODBC Data Source Administrator** applet in the **Control Panel** as system sources (**System DSN**) that are visible to all users and NT services on the system, or to user sources (**User DSN**) that are visible only to you, the current user.

For Windows operating systems, access this applet by selecting Administrative Tools in the **Control Panel**, then **Data Sources (ODBC)**.

Based upon each data source configuration, the data source is allowed access only if the logon account is correctly specified. A system data source can be accessed only when using a system account logon, which is the default. A user data source can be accessed only when using a user account logon that must be specified.

When using the Launcher and access to ODBC data sources is required, the Launcher must be started from the **Services** applet with the correct logon, either system or user account, that allows access to the data sources required by the Launcher during runtime.

For example, if a data source is defined as a system source in the **System Data Sources** list on the **System DSN** tab, the Launcher must be started using the **System Account** logon, which is the default.

However, if your ODBC data sources are defined as user data sources and are listed in the **User Data Sources** list on the User DSN tab of the **ODBC Data Source Administrator** applet, you must change the Launcher **Startup** options so that the logon is specified for the user account. This allows the Launcher to have access to the data sources associated with the user account that are required during Launcher runtime.

The following steps are based on the Windows operating system.

To change Launcher Startup Options:

1. From the Start menu, select **Settings > Control Panel**.
2. Select **Administrative Tools**.
3. Open the **Services** applet.
The Services dialog appears.
4. In the **Name** column, double-click IBM Transformation Extender Launcher.
The IBM Transformation Extender Launcher Properties dialog appears.
5. Click the **Log On** tab.
The Log On page appears.
6. Enable **This account** and click **Browse** to select the user account under which the DSN was created.
7. Enter values for the selected user account in the **Password** and **Confirm password** fields. The Launcher will use these values to log on to the system and access the user data sources.
8. Click **OK**.

For more information, see the *Launcher* documentation.

Restrictions and limitations

The Database Interface Designer and database adapters offer options and functions for accessing and manipulating data contained within a database. However, there is one restriction for this adapter: database triggers are not supported. You cannot use a data source as an input event trigger for the Launcher.

Return codes and error messages

Return codes and messages are returned when the particular activity completes. Return codes and messages might also be recorded as specified in the audit logs, trace files, execution summary files, and so on.

For information about error codes and messages returned by database-specific adapters, see the *Resource Adapters* documentation.

Various troubleshooting tools are available in case you encounter problems while using the database adapters. For example, if you attempt to run a map that uses the adapter and encounter problems or do not receive the expected results, use the following adapter troubleshooting tools:

- adapter audit log (.log)
- adapter trace file (.mtr)

OLE DB Adapter overview

Use the OLE DB Adapter to access and manipulate data contained in databases that are OLE DB data links. You can install database adapters on additional systems for remote database connectivity.

This documentation discusses the requirements for running the OLE DB Adapter and provides the database-specific information required for using your specific adapters in your database host environment.

- [System requirements](#)
- [Creating a universal data location file](#)
- [Database columns and types](#)
- [Database interface designer settings](#)
- [OLE DB Adapter commands](#)
- [Restrictions and limitations](#)
- [Return codes and error messages](#)

System requirements

Currently, there are three supported OLE DB providers to be used with the OLE DB Adapter:

- Microsoft OLE DB Provider for SQL Server
- Microsoft OLE DB Provider for Oracle
- Microsoft Jet OLE DB Provider (for Access)

Verify that you have installed one of the above listed OLE DB providers and have configured your OLE DB data links. The data link populates a Universal Data Connection (**.udl**) file that you can create. For more information about how to configure and manage these data links, see the OLE DB documentation included with your OLE DB providers.

Creating a universal data location file

Before you can access data from OLE DB, you must provide specific connection information such as:

- the type of data to be accessed
- the server upon which the data resides
- the database in which the data is stored

For example, to connect to an Oracle database, you must specify the Microsoft OLE DB provider for Oracle and provide a server name.

A connection string is a string version of this connection information that you can save and reuse in your applications. You can build this string in the Data Link Properties dialog and associated tabbed pages. Using this interface, you can save this connection string as a data link in the **.udl** file.

An alternative to using the Microsoft Data Link menu option as described in the following procedure, because some operating systems might not have this option, is to create a **.txt** file and change the file extension to **.udl**.

- [Creating a .udl file](#)

Creating a .udl file

The following procedure outlines how to create the **.udl** file necessary to use the OLE DB Adapter.

1. In Explorer, either create a new directory or select an existing directory in which to create a data link.
2. Highlight the directory. From the File menu, select New>Microsoft Data Link.
A file with the **.udl** extension is created.
3. Name the file and double-click it.
The Data Link Properties dialog appears.
4. On the Provider tab, select the OLE DB provider for your platform from the **OLE DB Providers** list.
5. Click Next.
The Connections tab appears.
6. Enter or select from the list the name of the data source.
7. Enter appropriate values in the **User name** and **Password** fields that will be used only to test the connection, not at runtime.
8. In the **Select initial catalog to use** field, enter the database to be used.
9. Click Test Connection. If successful, you can go to the Database Interface Designer Database Definition dialog.

Now you are ready to use this **.udl** file for the **Data Link** settings in the Database Definition dialog.

Database columns and types

The Database Interface Designer and **mtsmaker** with the Type Tree Maker generate type trees for queries, tables, views, and stored procedures in an OLE DB-compliant RDBMS (Relational Database Management System). Item types are created in a type tree that represent the data types of the columns of a query, table, view, or stored procedure.

The Database Interface Designer and **mtsmaker** get information about columns by calling the OLE DB Adapter that calls the OLE DB provider for your RDBMS to describe the columns associated with a query, table, view, or stored procedure. The OLE DB provider for the RDBMS has mapped the data types in each column to OLE DB data types as appropriate. From the RDBMS, the OLE DB Adapter returns the OLE DB data type, length, and other information to the Database Interface Designer and **mtsmaker**. The OLE DB data types are then mapped to types in a type tree. The Database Interface Designer and **mtsmaker** have no knowledge of the RDBMS-specific data types, only the OLE DB data types.

- [Item type properties](#)
- [Date and time formats](#)

Item type properties

The following table lists the OLE DB data types and the values of the item type properties to which they correspond when the type tree is generated.

OLE DB Data Type	Interpret as	Item Subclass, Presentation	Length
DBTYPE_UI1	Character	Number, Integer	3
DBTYPE_UI2	Character	Number, Integer	5
DBTYPE_UI4	Character	Number, Integer	10
DBTYPE_UI8	Character	Number, Integer	20
DBTYPE_I1	Character	Number, Integer	4
DBTYPE_I2	Character	Number, Integer	6
DBTYPE_I4	Character	Number, Integer	11
DBTYPE_I8	Character	Number, Integer	20
DBTYPE_R4	Binary	Number, Float	4
DBTYPE_R8	Binary	Number, Float	8
DBTYPE_CY	Character	Number, Decimal	*

OLE DB Data Type	Interpret as	Item Subclass, Presentation	Length
DBTYPE_NUMERIC	Character	Number, Decimal	*
DBTYPE_BOOL	Character	Text (ASCII)	1
DBTYPE_STR	Character	Text (ASCII)	*
DBTYPE_WSTR	Character	Text (UNICODE)	*
DBTYPE_BYTES	Binary	Text (Byte stream)	*
DBTYPE_DBDATE	Character	Date & Time	10
DBTYPE_DBTIME	Character	Date & Time	8
DBTYPE_DBTIMESTAMP	Character	Date & Time	19-26

* The DBMS dictates the length of this type.

Date and time formats

When using columns defined as dates and times in OLE DB databases, the format for both input and output is:

date

yyyy-mm-dd

time:

hh:mi:ss

timestamp:

yyyy-mm-dd hh:mi:ss [.fff...]

If the group format is fixed, the field is padded to 26 with trailing spaces.

where

yyyy	=	a four-digit year
mm	=	a two-digit month
dd	=	a two-digit day
hh	=	a two-digit hour
mi	=	a two-digit minute
ss	=	a two-digit second
.fff...	=	an optional fractional second

An example is 2001-08-27 00:00:00 which specifies August 27, 2001.

In the Generate Type Tree from dialog, use the **Represent date/time columns as text items** check box to define whether to automatically format this informations as Date and Time, which is with this check box disabled, as shown in the **Item Subclass, Presentation** column above, as a text string. When generated as a text string, it might be necessary to use either the TEXTTODATE or TEXTTOTIME function in a map rule to convert the text string to the date and time format required by the database. If you are generating new type trees, it is recommended that you disable this check box.

The **Represent date/time columns as text items** check box is modal. After it has been disabled, it will remain disabled for all subsequent type tree generations, regardless of source. Therefore, you must be careful in determining this setting.

Database interface designer settings

When you define an OLE DB database in the Database Interface Designer, in addition to the common settings available for all of the database adapters in the Database Definition dialog, you must enter the information specific to OLE DB.

- [Database Designer dialog](#)
- [Stored procedures native call syntax](#)

Database Designer dialog

The OLE DB Adapter-specific settings are described in the following table:

Setting		Description
Data Link		
	Database Interface Designer	Location of the Universal Data Location (.udl) file on your OLE DB development PC that is used to access the database information for design-time purposes

Setting		Description
	Runtime	Location of the Universal Data Location (.udl) file defined in your OLE DB database host platform to be used to access the database for runtime (map execution) purposes both from the Map Designer and either a Command Server or a Launcher.
Security		
	User ID	The user ID to connect to the database.
	Password	The authorization password to connect to the database.

Stored procedures native call syntax

A stored procedure can be accessed from within a map by specifying the native call syntax in the following:

- a query that is specified in an input card
- the first argument in a `DBQUERY` or `DBLOOKUP` function
This argument does not need to be a literal. The arguments to the stored procedure might be determined at map execution time.
- the SQL Statement adapter command (`-STMT`) in `DBQUERY`, `DBLOOKUP`, or `GET` functions

For information about using `DBLOOKUP` or `DBQUERY`, see the *Database Interface Designer* documentation. For information about using the syntax for device-independent calls to access return values and output adapter commands, see the *Database Interface Designer* documentation. Also see that documentation for information about calling stored procedures with object type parameters.

For example, a call to a stored procedure using OLE DB in a `DBLOOKUP` might be:

```
DBLOOKUP      ("{call MyAddNameProc('' + Name:Column + '',-1)}",
    "mydb.mdq",
    "MyDB")
```

String literals must be contained within single quotation marks; adapter arguments must be separated by commas.

OLE DB Adapter commands

This documentation describes the functions and usage of the OLE DB Adapter commands and their options.

- [Adapter command summary](#)
- [Adapter commands for a source](#)
- [Adapter commands for a target](#)

Adapter command summary

The following is a summary of the adapter commands that can be used to specify data sources and targets. The commands you use depends upon if you are specifying a source or target, if a database or query file (.mdq) is used, and the situations in which you are using the command. Adapter commands can be used in `GET`, `Source`, `Command` or `PUT`, `Target`, `Command` settings in the Map Designer and Integration Flow Designer, using `GET`, `PUT`, `DBLOOKUP`, or `DBQUERY` function calls, or overriding a data source or target using execution commands in a `RUN` function or on the command line.

- [Adapter specific commands](#)
- [Database Adapter Type \(-DBTYPE\)](#)
- [Row Count \(-ROWCNT\)](#)
- [Data Source \(-SOURCE\)](#)

Adapter specific commands

These adapter commands are OLE DB specific database parameters. For a complete listing of all database adapter commands, see the *Resource Adapters* documentation.

- [Database Adapter Type \(-DBTYPE\)](#)
- [Row Count \(-ROWCNT\)](#)
- [Data Source \(-SOURCE\)](#)

Database Adapter Type (-DBTYPE)

Use the Database Adapter Type adapter command (`-DBTYPE`) to specify the database adapter type.

-DBTYPE OLEDB

Option

Description

OLEDB

The database adapter type is OLE DB.

This command must be specified if the original card is not a database and no database is specified using the Database/Query adapter command (-MDQ) and the Database Name adapter command (-DBNAME).

Row Count (-ROWCNT)

Use the Row Count adapter command (-ROWCNT) to specify the number of rows to be retrieved per fetch.

-ROWCNT row_count

Option

Description

row_count

This is the number of rows to be retrieved per fetch.

The time to fetch is optimized in accordance with the increase in the number of rows to retrieve. Therefore, a significant performance improvement might be gained by retrieving as many rows as possible in a single fetch. The number of rows retrieved is only limited by the amount of memory installed on the querying PC.

The default for the number of rows to be retrieved per fetch for the OLE DB Adapter is one row.

Data Source (-SOURCE)

Use the Data Source adapter command (-SOURCE) to specify the OLE DB data link.

-SOURCE datalink

Option

Description

datalink

Specify the OLE DB data link.

Adapter commands for a source

This summary shows the syntax of the adapter commands that can be used when defining a data source using an .mdq file or without using an .mdq file, including both the required and optional adapter commands in the following situations:

- using a GET>Source>Command setting in the Map Designer and Integration Flow Designer
- overriding a data source using the Input Source Override - Database execution command (-ID) using a RUN function or on the command line
- using a DBLOOKUP, DBQUERY, or GET function in map or component rules
- [GET> Source> Command setting](#)
- [-ID Execution command](#)
- [DBLOOKUP or DBQUERY functions](#)
- [GET function](#)

GET> Source> Command setting

Use the Map Designer or Integration Flow Designer to specify **Database** as the value for the GET>Source setting and enter the adapter commands for the **Command** setting.

With Database/Query File	Without Database/Query File
[-DBTYPE OLEDB]	-DBTYPE OLEDB
[-SOURCE <i>datalink</i>]	-SOURCE <i>datalink</i>
[-STMT <i>SQL_statement</i>]	-STMT <i>SQL_statement</i>
[-FILE [directory]]	[-FILE [directory]]
[-VAR <i>name=value...</i>]	[-VAR <i>name=value...</i>]
[-USER <i>user_ID</i>]	[-USER <i>user_ID</i>]
[-PASSWORD <i>password</i>]	[-PASSWORD <i>password</i>]
[-CSTMT [<i>number</i>]]	[-CSTMT [<i>number</i>]]
[-ROWCNT <i>row_count</i>]	[-ROWCNT <i>row_count</i>]
[-AUDIT[G][+] [<i>full_path</i>]]	[-AUDIT[G][+] [<i>full_path</i>]]
[[-TRACE]-TRACEERR[+] [<i>full_path</i>]]	[[-TRACE]-TRACEERR[+] [<i>full_path</i>]]

-ID Execution command

Use the Input Source Override - Database execution command (-ID) to designate a database as the source. Or, use it to override one or more of the adapter command settings or database definitions in a RUN function or on the command line.

The following adapter commands are shown using a single quotation mark, which is the Windows syntax. For non-Windows platforms, use two single quotation marks followed by one single quotation mark and end with one single quotation mark followed by two single quotation marks.

- [The source in the compiled map is a database](#)
- [The source in the compiled map is not a database](#)

The source in the compiled map is a database

With Database/Query File

```
'[-MDQ mdq_file -DBNAME database_name]
[-QUERY query_name|-STMT SQL_stmt]
[-FILE [directory]]
[-VAR name=value...]
[-USER username]
[-PASSWORD password]
[-CCARD|-CSTMT [number]]
[-ROWCNT row_count]
[-AUDIT[G][+] [full_path]]
[{-TRACE|-TRACEERR}[+][full_path]]'
```

Without Database/Query File

```
'-DBTYPE OLEDB
-SOURCE datalink
-STMT SQL_statement
[-FILE [directory]]
[-VAR name=value...]
[-USER user_ID]
[-PASSWORD password]
[-CCARD|-CSTMT [number]]
[-ROWCNT row_count]
[-AUDIT[G][+] [full_path]]
[{-TRACE|-TRACEERR}[+][full_path]]'
```

The source in the compiled map is not a database

With Database/Query File

```
'-MDQ mdq_file
-DBNAME database_name
-QUERY query_name|-STMT SQL_stmt
[-FILE [directory]]
[-VAR name=value...]
[-USER username]
[-PASSWORD password]
[-CCARD|-CSTMT [number]]
[-ROWCNT row_count]
[-AUDIT[G][+] [full_path]]
[{-TRACE|-TRACEERR}[+][full_path]]'
```

Without Database/Query File

```
'-DBTYPE OLEDB
-SOURCE datalink
-STMT SQL_statement
[-FILE [directory]]
[-USER user_ID]
[-PASSWORD password]
[-CCARD|-CSTMT [number]]
[-ROWCNT row_count]
[-AUDIT[G][+] [full_path]]
[{-TRACE|-TRACEERR}[+][full_path]]'
```

DBLOOKUP or DBQUERY functions

The DBLOOKUP and DBQUERY functions can be used in component rules in the Type Designer and map rules in the Map Designer when creating a map that can be used with a database.

With Database/Query File	Without Database/Query File
DBLOOKUP ("SQL_statement", "-MDQ mdq_file -DBNAME database_name [-USER username] [-PASSWORD password] [-CCARD -CSTMT [number]] [-ROWCNT row_count] [-AUDIT[G][+] [full_path]] [{-TRACE -TRACEERR}[+][full_path]])"	DBQUERY ("SQL_statement", "-DBTYPE OLEDB -SOURCE datalink [-USER username] [-PASSWORD password] [-CCARD -CSTMT [number]] [-ROWCNT row_count] [-AUDIT[G][+] [full_path]] [{-TRACE -TRACEERR}[+][full_path]])"

GET function

The `GET` function returns the data from the source adapter.

With Database/Query File	Without Database/Query File
<code>GET ("DB", "-MDQ mdq_file -DBNAME database_name -QUERY query_name -STMT SQL_stmt [-FILE [directory]] [-VAR name=value...] [-USER username] [-PASSWORD password] [-CCARD -CSTMN [number]] [-ROWCNT row_count] [-AUDIT[G][+] [full_path]] [{-TRACE}-TRACEERR][+] [full_path]]")</code>	<code>GET "DB", "-DBTYPE OLEDB -SOURCE datalink -STMT SQL_stmt [-FILE [directory]] [-USER username] [-PASSWORD password] [-ROWCNT row_count] [-CCARD -CSTMN [number]] [-AUDIT[G][+] [full_path]] [{-TRACE}-TRACEERR][+] [full_path]]")</code>

Adapter commands for a target

This summary shows the syntax of the adapter commands that can be used when defining a data target either using or not using an `.mdq` file, including both the required and optional adapter commands in the following situations:

- using the `PUT>Target>Command setting` in the Map Designer and Integration Flow Designer
- overriding a data source using the Output Source Override - Database execution command (`-OD`) using a `RUN` function or on the command line
- using the `PUT` function in map or component rules
- [PUT>Target>Command setting](#)
- [-OD Execution command](#)
- [PUT function](#)

PUT> Target> Command setting

Use the Map Designer or the Integration Flow Designer to specify **Database** as the value for the `PUT>Target` setting and enter the adapter commands for the **Command** setting.

With Database/Query File	Without Database/Query File
<code>[-DBTYPE OLEDB] [-SOURCE datalink] [-PROC procedure_name -TABLE table_name] [-USER user_ID] [-PASSWORD password] [-CSTMN [number]] [-DELETE] [-UPDATE [OFF ONLY]] [-BADDATA[+] full_path] [-AUDIT[G][+] [full_path]] [{-TRACE}-TRACEERR][+] [full_path]]</code>	<code>-DBTYPE OLEDB -SOURCE datalink -PROC procedure_name -TABLE table_name [-USER user_ID] [-PASSWORD password] [-CSTMN [number]] [-DELETE] [-BADDATA[+] full_path] [-AUDIT[G][+] [full_path]] [{-TRACE}-TRACEERR][+] [full_path]]</code>

-OD Execution command

Use the Output Source Override - Database execution command (`-OD`) to designate a database as a target. You can also use it to override one or more of the adapter command settings or database definitions in a `RUN` function or on the command line.

The following adapter commands are shown using a single quotation mark, which is the Windows syntax. For non-Windows platforms, use two single quotation marks followed by one single quotation mark and end with one single quotation mark followed by two single quotation marks.

- [The target in the compiled map is a database](#)
- [The target in the compiled map is not a database](#)

The target in the compiled map is a database

With Database/Query File

```
' [-MDQ mdq_file -DBNAME database_name]  
[-PROC procedure_name|-TABLE table_name]  
[-USER username]  
[-PASSWORD password]
```

```

[-CCARD|-CSTMT [number]]
[-DELETE]
[-UPDATE [OFF|ONLY]]
[-BADDATA[+] full_path]
[-AUDIT[G][+] [full_path]]
[{-TRACE|-TRACEERR}[+] [full_path]]'

```

Without Database/Query File

```

'-DBTYPE OLEDB
-SOURCE datalink
-PROC procedure_name|-TABLE table_name
[-USER username]
[-PASSWORD password]
[-CCARD|-CSTMT [number]]
[-DELETE]
[-UPDATE [OFF|ONLY]]
[-BADDATA[+] full_path]
[-AUDIT[G][+] [full_path]]
[{-TRACE|-TRACEERR}[+] [full_path]]'

```

The target in the compiled map is not a database

With Database/Query File

```

'-MDQ mdq_file
-DBNAME database_name
-PROC procedure_name|-TABLE table_name
[-USER username]
[-PASSWORD password]
[-CCARD|-CSTMT [number]]
[-DELETE]
[-UPDATE [OFF|ONLY]]
[-BADDATA[+] full_path]
[-AUDIT[G][+] [full_path]]
[{-TRACE|-TRACEERR}[+] [full_path]]'

```

Without Database/Query File

```

'-DBTYPE OLEDB
-SOURCE datalink
-PROC procedure_name|-TABLE table_name
[-USER username]
[-PASSWORD password]
[-CCARD|-CSTMT [number]]
[-DELETE]
[-UPDATE [OFF|ONLY]]
[-BADDATA[+] full_path]
[-AUDIT[G][+] [full_path]]
[{-TRACE|-TRACEERR}[+] [full_path]]'

```

PUT function

Use the **PUT** function to pass data to the target adapter.

With Database/Query File	Without Database/Query File
<pre> PUT ("DB", "-MDQ mdq_file -DBNAME database_name -PROC procedure_name -TABLE table_name [-USER username] [-PASSWORD password] [-CCARD -CSTMT [number]] [-DELETE] [-UPDATE [OFF ONLY]] [-BADDATA[+] full_path] [-AUDIT[G][+] [full_path]] [{-TRACE -TRACEERR}[+] [full_path]]") </pre>	<pre> PUT ("DB", "-DBTYPE OLEDB -SOURCE datalink -PROC procedure_name -TABLE table_name [-USER username] [-PASSWORD password] [-CCARD -CSTMT [number]] [-DELETE] [-BADDATA[+] full_path] [-AUDIT[G][+] [full_path]] [{-TRACE -TRACEERR}[+] [full_path]]") </pre>

Restrictions and limitations

The Database Interface Designer and adapters offer tremendous options and functions for accessing and manipulating data contained within a database. However, there are some restrictions and limitations of certain functions when using the adapters for OLE DB. These functions are:

- **Database Triggers**
Using a data source as an input event trigger for the Launcher is not supported.
- **Bind Facility**
Using bind values in database functions is not supported.

Return codes and error messages

Return codes and messages are returned when the particular activity completes. Return codes and messages might also be recorded as specified in the audit logs, trace files, execution summary files, and so on.

For information about error codes and messages returned by database-specific adapters, see [Return Codes and Error Messages > Database-specific Adapter Messages](#) in the *Resource Adapters* documentation.

Troubleshooting tools are available in case you encounter problems while using the database adapters. For example, if you attempt to run a map that uses the adapter and encounter problems or do not receive the expected results, use the following adapter troubleshooting tools:

- adapter audit log (**.log**)
- adapter trace file (**.mtr**)

OpenPGP Adapter overview

Use the OpenPGP Adapter to exchange information with the GnuPG application. The OpenPGP Adapter creates so called *pipes* to perform the Inter Process Communication (IPC) between the process that uses the adapter, for example, Command Server and the GnuPG process.

The OpenPGP Adapter is based on the Gnu implementation of the OpenPGP Adapter standard, called GnuPG. The adapter uses native operating system calls to start the GnuPG application, and then it uses pipes to exchange information with it.

- [Pipes](#)
- [System requirements](#)
- [Command alias](#)
- [OpenPGP Adapter commands](#)
- [Syntax summary](#)

Pipes

The OpenPGP Adapter uses the following pipes:

- **Status pipe** to obtain status information from GnuPG about the operation that is being performed. For example, status messages that describe steps in the encryption procedure.
- **Command pipe** to specify information during the GnuPG execution that is normally provided through the standard input, either a shell or prompt. This pipe is used to provide the user password when GnuPG needs it, for example, when signing a message. See [User Password \(-PW\)](#) to see how a password is provided to GnuPG from a map.
- **Standard input pipe** to provide GnuPG with the data on which the operation is performed, for example, plain text for encryption.
- **Standard output pipe** to read processed data, for example, cipher text that represents the encrypted data.

System requirements

The minimum requirements and installation instructions for the OpenPGP Adapter are detailed in the release notes. It is assumed that a Command Server has already been installed on the computer where the adapter is to be installed for runtime purposes.

Command alias

Adapter commands can be specified by using a command string on the command line or creating a command file that contains adapter commands. The execution command syntax is:

```
-IM[alias] card_num  
-OM[alias] card_num
```

where **-IM** is the Input Source Override execution command and **-OM** is the Output Target Override execution command, *alias* is the adapter alias, and *card_num* is the number of the input or output card. The following table shows the adapter alias and its execution command.

Adapter	Alias	As Input	As Output
OpenPGP Adapter	OPGP	-IMOPGP <i>card_num</i>	-OMOPGP <i>card_num</i>

OpenPGP Adapter commands

This documentation describes the functions and use of the OpenPGP Adapter commands and their options.

- [List of commands](#)

List of commands

The following table lists valid commands for the OpenPGP Adapter, the command syntax, and if the command is supported (✓) for use with data sources, targets, or both.

Command	Syntax	Source	Target
Action (-ACTION)	-ACTION <i>action</i>	✓	✓
Additional Command (-ADDCMD)	-ADDCMD <i>additional_command(s)</i>	✓	✓
Comment (-COMMENT)	-COMMENT <i>comment</i>	✓	✓
Compression Algorithm (-COMALG)	-COMALG <i>algorithm_id</i>	✓	✓
Decode (-DECODE)	-DECODE	✓	✓
Encode (-ENCODE)	-ENCODE	✓	✓
Encryption Algorithm (-ENCALG)	-ENCALG <i>algorithm_id</i>	✓	✓
Encryption Type (-ENCTYPE)	-ENCTYPE <i>type</i>	✓	✓
Message Digest (-DIGEST)	-DIGEST	✓	✓
Message File Name (-FILE)	-FILE <i>filename</i>	✓	✓
Message Recipient (-R)	-R <i>recipient</i>	✓	✓
Signature Type (-SIGTYPE)	-SIGTYPE <i>type</i>	✓	✓
Signing Algorithm (-SIGALG)	-SIGALG <i>algorithm_id</i>	✓	✓
Trace (-T)	-T[+][<i>file_name</i>]	✓	✓
User Login (-U)	-U <i>user</i>	✓	✓
User Password (-PW)	-PW <i>password</i>	✓	✓

- [Action \(-ACTION\)](#)
- [Additional Command \(-ADDCMD\)](#)
- [Comment \(-COMMENT\)](#)
- [Compression Algorithm \(-COMALG\)](#)
- [Decode \(-DECODE\)](#)
- [Encode \(-ENCODE\)](#)
- [Encryption Algorithm \(-ENCALG\)](#)
- [Encryption Type \(-ENCTYPE\)](#)
- [Message Digest \(-DIGEST\)](#)
- [Message File Name \(-FILE\)](#)
- [Message Recipient \(-R\)](#)
- [Signature Type \(-SIGTYPE\)](#)
- [Signing Algorithm \(-SIGALG\)](#)
- [Trace \(-T\)](#)
- [User Login \(-U\)](#)
- [User Password \(-PW\)](#)

Action (-ACTION)

Use the Action adapter command (-ACTION) to specify what action needs to be performed by the adapter.

-ACTION *action*

The following table lists the *action* values and their meaning:

Action	Meaning
SIGN	Only sign the message and return the signed data. The -PW adapter command must be specified along with this action to specify the password used to access the signer's private key. The -USER adapter command might be used to specify the signer other than the default user.
ENCRYPT	Only encrypt the message and return the encrypted data. The -R adapter command must be used along with this action to specify the user for which the message is being encrypted.
SIGNANDENCRYPT	Sign and encrypt the message and return the signed and encrypted data.
VERIFY	Only verify the signed message. No data is returned.
DECRYPT	Only decrypt the encrypted message and return the original data. If the message is signed, the signature is verified as well. This is the default action.
VERIFYANDDECRYPT	Same as DECRYPT. The signature is automatically verified if it is present in the encrypted message.

Additional Command (-ADDCMD)

Use the Additional Command adapter command (-ADDCMD) to specify additional options in the command line that is passed to the **GnuPG** executable.

These options are added to the options specified by the OpenPGP Adapter in the adapter command. See the GnuPG documentation to learn about the various command line options that can be used with the GnuPG.

If the Trace adapter command (-T) is specified, the adapter will trace the complete command line that was passed to GnuPG.

```
-ADDCMD " additional_command(s) "
```

For example:

```
-ADDCMD " --armor --version -verbose "
```

- --armor creates ASCII armored output
 - --version displays the GnuPG version in the output message. By default, the version is not displayed.
 - --verbose generates some extra information about the progress of the performed action, which can later be looked at in the trace file.
- Use extreme caution with the -ADDCMD adapter command. Make sure that the GnuPG command options specified through -ADDCMD are not in conflict with the commands that the adapter uses. Do not use the -ADDCMD adapter command to specify options that can be specified through other adapter commands. For example, the -ACTION SIGN adapter command should be used for signing documents, and not -ADDCMD "--sign". The former format instructs the adapter to communicate with GnuPG in the signing mode, while the later one only tells the adapter to append additional commands to the initial GnuPG command line.

Also, note that the --comment GnuPG command option should not be specified through the -ADDCMD adapter command. The OpenPGP Adapter provides the -COMMENT adapter command that can be used for this purpose.

Comment (-COMMENT)

Use the Comment adapter command (-COMMENT) to specify the comment that appears in the encrypted or signed message's header. This adapter command corresponds to the --comment GnuPG command option.

```
-COMMENT " comment "
```

For example:

```
-COMMENT " Message signed with GnuPG "
```

When the -COMMENT adapter command is not present; the comment is omitted from the generated message.

When the comment value contains no white spaces, quotes can be omitted. However, note that on UNIX systems, the adapter does not support comments with white spaces. Replace the white spaces with other characters, for example, the underscore character '_', before passing it to the adapter.

Compression Algorithm (-COMALG)

Use the Compression Algorithm adapter command (-COMALG) to specify the compression algorithm used to compress the message.

It corresponds to the --compress-algo NAME GnuPG command option.

```
-COMALG algorithm_id
```

The following table lists the allowed values for the *algorithm_id* and the corresponding algorithms:

Algorithm_id	Algorithm
1	ZLIB compression (RFC1951)
2	RFC1950 compression (default)

Decode (-DECODE)

The -DECODE adapter command is a special command that is used in the adapter chaining process.

The -DECODE special command should be used *only* when chaining the adapter with another adapter, for example, the E-mail adapter. The -DECODE adapter command has no arguments.

```
-DECODE
```

Encode (-ENCODE)

The -ENCODE adapter command is a special command that is used in the adapter chaining process.

The -ENCODE special command should be used *only* when chaining the adapter with another adapter, for example, the E-mail adapter. The -ENCODE adapter command has no arguments.

```
-ENCODE
```

Encryption Algorithm (-ENCALG)

Use the Encryption Algorithm adapter command (-ENCALG) to specify the cipher algorithm used in the message encryption process.

-ENCALG *algorithm_id*

The following table lists the allowed values for the *algorithm_id* and the corresponding algorithms:

Algorithm_id	Algorithm
3DES	Triple DES Block Cipher
CAST5	CAST5 128-bit block cipher
BLOWFISH	Blowfish block cipher
RIJNDAEL	AES (Rijndael) block cipher 128 bit key
RIJNDAEL192	AES (Rijndael) block cipher 192 bit key
RIJNDAEL256	AES (Rijndael) block cipher 256 bit key
TWOFISH	Twofish 128 bit block cipher

The -ENCALG adapter command is meaningful only if the following conditions are met:

- -ENCODE adapter command is used
- -ACTION ENCRYPT or -ACTION SIGNANDENCRYPT adapter command is used

Encryption Type (-ENCTYPE)

Use the Encryption Type adapter command (-ENCTYPE) to specify the type of the encryption.

-ENCTYPE *type*

The allowed values for *type* are listed in the following table:

Type	Meaning
PUBLICKEY	Use public-key cipher for the message encryption. This is the default.
SYMMETRIC	Use only symmetric cipher for the message encryption.

Message Digest (-DIGEST)

Use the Message Digest adapter command (-DIGEST) to calculate the message digest for the specified data. The MD5 and SHA1 message digests can be calculated. The syntax is:

-DIGEST

This is an optional command.

The following table illustrates the usage of the -DIGEST adapter command depending on if the -ACTION and -SIGALG commands are specified:

	-SIGALG <i>algorithm_id</i>	-SIGALG not specified
-ACTION VERIFY_AND_DECRYPT or DECRYPT	The message is first verified (and/or decrypted), and the message digest is calculated for the original data using the algorithm specified with <i>algorithm_id</i> .	The message is first verified (and/or decrypted) and the message digest is calculated for the original data using SHA1 algorithm.
-ACTION not specified	The message digest is calculated directly for the input data using the <i>algorithm_id</i> algorithm.	The message digest is calculated directly for the input data using SHA1 algorithm.
-ACTION VERIFY or SIGN or ENCRYPT or SIGN_AND_ENCRYPT	invalid usage	invalid usage

The -ACTION VERIFY command is not supported because it does not return any data. If no data is returned, there is no information on which to calculate the message digest.

To verify the signature in a signed message and calculate the message digest for the originally signed data, -ACTION VERIFY_AND_DECRYPT or -ACTION DECRYPT should be used in combination with the -DIGEST command.

Message File Name (-FILE)

Use the Message File Name adapter command (**-FILE**) to specify the file used by the adapter on input or output depending on the map design scenario.

-FILE filename

Where *filename* is the full path to the file that contains the message data.

The message data will be plain or cipher text, depending on the action that is to be performed.

The actual interpretation of the value depends on the scenario in which the adapter is being used.

Depending on the context in which the adapter is being used, the **-FILE** adapter command will be treated differently. The possible scenarios are:

- [Input Card](#) (source adapter type is OpenPGP Adapter)
- [Output Card](#) (target adapter type is OpenPGP Adapter)
- [GET Map Function](#)
- [PUT Map Function](#)
- [Input card](#)
- [Output card](#)
- [GET Map function](#)
- [PUT map function](#)

Input card

When the source adapter type is OpenPGP Adapter the **-FILE** adapter command is required. The adapter performs the action specified in the **-ACTION** adapter command on the data from the specified file. The processed data is returned to the execution engine.

Output card

When the target adapter type is OpenPGP Adapter the **-FILE** adapter command is required. The adapter performs the action specified in the **-ACTION** adapter command on the data that it receives from the execution engine. The processed data is stored to the specified file. The **-ENCODE** and **-DECODE** adapter commands should not be specified in these cases.

GET Map function

If the **GET** function has three parameters, the adapter performs the action specified in the **-ACTION** adapter command on the data in the third parameter of the **GET** function. In this case, the **-FILE** adapter command might be omitted. If specified, it will be ignored by the adapter.

If the input data is not provided to the adapter from the execution engine, then the **-FILE** command must be specified. In that case, the adapter performs the action specified in the **-ACTION** adapter command on the data from the specified file.

In both cases, the processed data is returned to the execution engine.

PUT map function

The adapter performs the action specified in the **-ACTION** adapter command on the data in the third parameter of the **PUT** function.

If the adapter is not participating in adapter chaining, when its output is used as input to another adapter, the **-FILE** adapter command is required.

If adapter chaining is taking place, or **-ENCODE** or **-DECODE** is explicitly specified in the adapter command, the adapter ignores the **-FILE** adapter command and returns the processed data back to the execution engine to be passed as input to another adapter in the chain.

Message Recipient (-R)

Use the Message Recipient adapter command (**-R**) to specify the recipient of the message. The message is then encrypted for this particular recipient.

-R recipient

Where *recipient* corresponds to the local GnuPG user.

This command is required when the input message needs to be encrypted. The adapter encrypts the input message for the specified recipient.

Signature Type (-SIGTYPE)

Use the Signature Type adapter command (**-SIGTYPE**) to specify the type of the signature.

-SIGTYPE type

The allowed values for the *type* are listed in the following table:

Type	Meaning
NORMAL	Sign the data with the regular OpenPGP Adapter signature.
CLEAR	Sign the data with the clear text signature.
DETACHED	Creates a detached signature.

Signing Algorithm (-SIGALG)

Use the Signing Algorithm adapter command (-SIGALG) to specify the digest algorithm used for hashing when signing a message. It corresponds to the **--digest-algo NAME** option for the GnuPG executable.

-SIGALG algorithm_id

The following table lists the allowed values for the *algorithm_id* and the corresponding algorithms:

Algorithm_id	Algorithm
SHA1	Secure Hash Algorithm version 1.
MD5	Rivest's Message Digest.
RIPEMD160	160 bit output Message Digest.

The -SIGALG adapter command can always be specified, but its value will only be used when a message needs to be signed. This happens when the following condition is met:

- -ACTION SIGN or -ACTION SIGNANDENCRYPT adapter command is used

Trace (-T)

Use the Trace adapter command (-T) to produce a diagnostics file that contains detailed information about OpenPGP Adapter activity.

The default filename is **m4opgp.mtr** and it is created in the map directory unless otherwise specified.

-T[E] [+] [file_name]

Option

Description
E Produce a trace file containing only the adapter errors that occurred during map execution.
+ Append trace information to the existing trace file.
file_name Creates a trace file with the specified name in the specified directory.

You can override the adapter command line trace options dynamically using the Management Console. See "Dynamic Adapter Tracing" in the Launcher documentation for more information.

User Login (-U)

Use the User Login adapter command (-U) to specify the local user which needs to perform the action specified in -ACTION adapter command.

This adapter command corresponds to the **--local-user gpg** option.

-U user

Where *user* is the local GnuPG user.

User Password (-PW)

Use the User Password adapter command (-PW) to specify the password, also called passphrase, of the user whose private key needs to be accessed.

A password is required when signing or decrypting a message. It is also required when encrypting a message with a symmetric cipher (-ENCTYPE SYMMETRIC).

In these scenarios, the `-PW` adapter command is required. The `-PW` adapter command value is passed to the GnuPG executable through the *command pipe*. See ["Pipes"](#) for more information.

`-PW password`

Where *password* is the password of the local user specified in the `-U` adapter command, or the default user, if `-U` is not specified.

Syntax summary

This documentation discusses the OpenPGP Adapter syntax summary and how it is used.

- [Data sources and targets](#)
 - [Example 1](#)
 - [Example 2](#)
-

Data sources and targets

The following is the command syntax of the OpenPGP Adapter commands used for data sources and targets:

```
[-ACTION action] [-ADDCMD " additional_command(s) "]  
[-COMMENT " comment "] [-COMALG algorithm_id]  
[-DECODE]  
[-DIGEST] [-ENCODE]  
[-ENCALG algorithm_id]  
[-ENCTYPE type]  
[-FILE filename] (*)  
[-R recipient] (*)  
[-SIGTYPE type]  
[-SIGALG algorithm_id]  
[-T[E] [+]] [filename]  
[-U user]  
[-PW password] (*)
```

(*) This command is not optional in certain scenarios. See the description of the specific command to see when it is required.

Example 1

For the GET Source setting in an input card, select OpenPGP Adapter. In the **GET > Source > Command** field, enter:

```
-U foo -ACTION ENCRYPT -R foo -ADDCMD "-armor" -FILE input.txt -T
```

The adapter will:

- read the file **input.txt**
- encrypt it for the **foo** GnuPG user
- return the ASCII armored encrypted message back
- trace the activity in the default trace file **m4opgp.mtr** in the map directory

The encrypted message can then be passed to some target, for example, E-mail, in an output card.

Example 2

In a **PUT** map function call enter:

```
=PUT("OPGP", " -U foo -ACTION ENCRYPT -FILE out. t -R fee -  
ADDCMD "--armor\"", "Message to encrypt")
```

This map function call will:

- encrypt the *Message to encrypt* string for the *fee* GnuPG user, using *fee*'s public key. The message will be encrypted on behalf of the user *foo*.
- generate encrypted data that will be ASCII armored and stored to file **out.txt**.

Oracle Adapter overview

Use the Oracle Adapter to access and manipulate data contained in Oracle databases. The Database Interface Designer along with the database adapters for Windows are part of the Design Studio installation. You can also install database adapters on additional systems for remote database connectivity.

There are requirements for running the Oracle Adapter and other database-specific information required for using specific adapters in your database host environment.

You can use the adapter with a Command Server, a Launcher, the Software Development Kit, or with a map in a map rule.

- [System requirements](#)
- [Database columns and types](#)
- [Database Interface Designer settings](#)
- [Oracle client result cache support](#)

The Oracle adapter supports the Oracle client result cache. The Oracle client result cache stores the results of an Oracle database SQL **SELECT** query in memory on the Oracle client. Subsequent identical SQL queries can retrieve data from the cached query results in memory rather than through I/O to the Oracle database, reducing the number of Oracle server round trips and query executions.

- [Oracle Adapter commands](#)
- [Bulk copy example files](#)
- [Restrictions and limitations](#)
- [Return codes and error messages](#)

System requirements

See the [system requirements](#) for details. In addition to these requirements, to install and run the Oracle adapter:

- You must be currently running the appropriate version of Oracle on the machine upon which you are going to install this adapter. See also the [release notes](#).
- Verify that the Oracle environment variables, for example, ORACLE_HOME, are defined. For information about how to do the verification, see your Oracle documentation.
- When running on UNIX, the Oracle client creates a process whenever a connection is made to Oracle. If any of these processes terminates, defunct processes remain in the process list. Although not required, because these do not cause problems nor do they consume resources, you can prevent this occurrence by adding the following line to the sqlnet.ora file in the \$ORACLE_HOME/network/admin path:

```
bequeath_detach=YES
```

The effect of this line is to create these same connection processes as detached orphans, thus eliminating the possibility of them becoming defunct.

- Set the shared library path environment variable for UNIX.

After installing the adapters on a UNIX system and running the environment variable setup program (**setup**), add the *install_dir* directory to your shared library path environment variable if it is not already there. The following instructions assume that your *install_dir* installation directory is defined in the **DTX_HOME_DIR** environment variable.

If you change your path or library path after you run the **setup** program, you must run the **setup** program again.

Note: These instructions assume that your UNIX environment is the Korn (ksh) shell.

- For HP-UX:

```
SHLIB_PATH=$DTX_HOME_DIR/lib:$ORACLE_HOME/lib:$SHLIB_PATH
export SHLIB_PATH
```

- For Solaris:

```
LD_LIBRARY_PATH=$DTX_HOME_DIR/lib:$ORACLE_HOME/
lib:$LD_LIBRARY_PATH export LD_LIBRARY_PATH
```

- For IBM® RS/6000® AIX®:

```
LIBPATH=$DTX_HOME_DIR/lib:$ORACLE_HOME/lib:$LIBPATH
export LIBPATH
```

Database columns and types

The Database Interface Designer and **mtsmaker** with the Type Tree Maker generate type trees for queries, tables, views, and stored procedures in an Oracle-compliant DBMS (Database Management System). Item types will be created in a type tree that represent the data types of the columns of a query, table, view, or stored procedure.

The Database Interface Designer and **mtsmaker** get information about columns by calling Oracle to describe the columns associated with a query, table, view, or stored procedure. Oracle returns the data type, length, and other information to the Database Interface Designer and **mtsmaker**. The Database Interface Designer and **mtsmaker** then map the Oracle data types to item types in a type tree.

Note: Type trees for messages can be generated only using the Database Interface Designer. For more information, see the Database Interface Designer documentation.

- [Item type properties](#)
- [Date and time format](#)
- [XML type](#)
- [URIType](#)

Item type properties

The following table lists the Oracle data types and the values of the item type properties to which they correspond in a generated type tree. The DBMS dictates the length of each type except for the DATE type. The DATE type length is 19.

Oracle Data Type	Interpret as	Item Subclass, Presentation
VARCHAR2	Character	Text
NVARCHAR2	Character	Text
INTEGER	Character	Number, Integer

Oracle Data Type	Interpret as	Item Subclass, Presentation
NUMBER Precision = 0 Scale = 0	Character	Number, Decimal
NUMBER Precision > 0 Scale = 0	Character	Number, Integer
NUMBER Precision > 0 Scale > 0	Character	Number, Decimal
FLOAT	Character	Number, Decimal
LONG	Character	Text
DATE	Character	Date & Time
RAW	Binary	Text
LONG RAW	Binary	Text
CHAR	Character	Text
NCHAR	Character	Text
BLOB	Binary	Text
CLOB	Character	Text
BFILE	Binary	Text
NCLOB	Character	Text
TIMESTAMP(f)	Character	
TIMESTAMP(f) WITH TIME ZONE	Character	Text
TIMESTAMP(f) WITH LOCAL TIME ZONE	Character	
INTERVAL DAY(d) TO SECOND(f)	Character	Text
INTERVAL YEAR(y) TO MONTH	Character	Text
XMLType	Character	Text
URIType	Character	Text

*

The BLOB and NCLOB data types are not supported as parameters for stored procedures. The Oracle Adapter supports object types. These are alternatively referred to as abstract data types (ADT). The object type is a composite type that includes data types such as number, date, varchar, and so on. Object types can include nested object types.

When generating Oracle type trees that contain ADTs, you must manually edit the tree to make the ADT group required. This will prevent atomic null violations.

Date and time format

The Oracle date and time format for both input and output is:

`ccyy-mm-dd hh24:mm:ss [.6-6]`

If the group format is fixed, the field is padded to 19 with trailing spaces.

Formats

Format	Description
cc	two-digit century
yy	two-digit year
mm	two-digit month
dd	two-digit day
hh	two-digit hour
mi	two-digit minute
ss	two-digit second
.fff...	optional fractional seconds
ffffff	six-digit fraction of a second
hh24	two-digit hour using a 24-hour day

An example is 2001-08-27 23:00:00, which specifies August 27, 2001 at 11:00 PM.

For the **DATE** column type, there are never fractional seconds; hence, the width of the field is always 19. For the **TIMESTAMP** and **TIMESTAMP WITH LOCAL TIME** columns, the precision of the fractional seconds is determined by the column definition.

In the Generate Type Tree from dialog, use the Represent date/time columns as text items check box to define whether to automatically format this information as Date & Time (which is with this check box disabled, the result of which is shown in the Item Subclass, Presentation column) as a text string. When generated as a text string, it might be necessary to use either the **TEXTTODATE** or **TEXTTOTIME** function in a map rule to convert the text string to the date and time format required by the database. If you are generating new type trees, it is recommended that you disable this check box.

This Represent date/time columns as text items check box is modal. After it has been disabled, it will remain disabled for all subsequent type tree generations, regardless of source. Therefore, you must be careful in determining this setting.

XML type

XMLEType is supported and can still be used to a limited degree. It is possible to query **XMLEType** columns as well as inserting data into them.

To query from an **XMLEType** column, the `getClobVal` method of the **XMLEType** object must be used.

Consider the table created by this statement:

```
CREATE TABLE MY_XML (myCol sys.xmletype)
```

The following query can be executed to get the XML data:

```
SELECT t.myCol.getClobVal() FROM MY_XML t
```

To insert into the table, you must use the `createXML` method. It is not possible to do this by creating a type tree for the table and using this tree in an output card. Instead, the `INSERT` statement should be executed through a `DBLOOKUP`.

The following is an example of the syntax of the SQL:

```
INSERT INTO MY_XML VALUES (sys.xmletype.createXML
(`<XMLDocument>cat</XMLDocument>'))
```

Oracle parses the XML document when it is inserted. Therefore, the above statement will fail if the XML is invalid.

URIType

URIType provides a means to store a URI in the database, rather than the document referenced by the URI. In the same way that the methods of the **XMLEType** object can be used to insert or create XML document objects, methods of **URIType** objects can be used to insert or query URIs.

Consider the table created by this statement:

```
CREATE TABLE uri_tab (url sys.uritype)
```

The following query can be executed to get the document referenced by the URI:

```
SELECT e.url.getLOB() FROM uri_tab e
```

To insert into the table, you must use the `createHTTPURI` or `createDBURI` method. It is not possible to do this by creating a type tree for the table and using this tree in an output card. Instead, the `INSERT` statement must be executed through a `DBLOOKUP`.

The following is an example of the syntax:

```
INSERT INTO uri_tab VALUES (sys.httpuritype.createHttpuri
('http://www.oracle.com'))
```

or

```
INSERT INTO uri_tab VALUES (sys.dburitype.createDBuri
('/SCOTT/ EMPLOYEE/ROW[ENAME="Jack"]'))
```

Database Interface Designer settings

When you define an Oracle database in the Database Interface Designer, in addition to the common settings available for all the database adapters in the Database Definition dialog, you need to enter the information specific for Oracle.

- [Database definitions](#)
- [Specifying connect strings](#)
- [Stored procedures native call syntax](#)

Database definitions

The Oracle Adapter-specific settings in the Database Interface Designer are described in the following table.

Setting		Description
Connect String		

Setting		Description
	Database Interface Designer	The connect string used to connect to the Oracle database that is accessed by the Database Interface Designer for design-time purposes
	Runtime	The connect string used to connect to the Oracle database to be accessed for runtime (map execution) purposes both from the Map Designer and either a Command Server or a Launcher
Security		
	User ID	The user ID used to connect to the database
	Password	The authorization password to connect to the database

Specifying connect strings

If an SQL*Net host connection string is specified, the connection is established through SQL*Net. If no connection string is specified, a direct connection is established to the database identified by the current ORACLE_SID environment setting.

For example, specifying

```
UserID
  Scott
Connect String
    remote_db
```

results in connecting with the connection string `Scott@remote_db`.

Note: A connection string is always required when connecting to a remote database. For more information about the syntax of the connection string, see your Oracle documentation.

Stored procedures native call syntax

A stored procedure can be accessed from within a map by specifying the native call syntax in:

- A query that is specified in an input card.
- The first argument in a DBQUERY or DBLOOKUP function.
This argument does not need to be a literal. The arguments to the stored procedure might be determined at map execution time.
- The SQL Statement adapter command (-STMT) in DBQUERY, DBLOOKUP, or GET functions.

For information about using DBLOOKUP or DBQUERY, or for information about using the syntax for device-independent calls to access return values and output adapter commands, see the Database Interface Designer documentation. In addition, you will find information about calling stored procedures with object type parameters.

For example, a call to a stored procedure using Oracle in a DBLOOKUP might be:

```
DBLOOKUP ("begin MyAddNameProc('"+ Name:Column +" ',-1);end;"  
"mydb.mdq",  
"MyDB")
```

Note: String literals must be contained within single quotation marks; adapter arguments must be separated by commas.

Oracle client result cache support

The Oracle adapter supports the Oracle client result cache. The Oracle client result cache stores the results of an Oracle database SQL **SELECT** query in memory on the Oracle client. Subsequent identical SQL queries can retrieve data from the cached query results in memory rather than through I/O to the Oracle database, reducing the number of Oracle server round trips and query executions.

Maps that make identical SQL **SELECT** queries to an Oracle database can potentially improve performance by using the adapter cache (**-CACHE**) option. The queries must be for simple data types in a read-only or read-mostly database.

- [SQL SELECT query requirements](#)
SELECT queries must be syntactically identical to retrieve results from the Oracle client result cache and to reuse database connections. Use bind variables to create syntactically identical **SELECT** queries.
- [Tuning Oracle adapter database connections](#)
The Oracle adapter **-CACHE** option defines a unique database connection for each **SELECT** statement. This connection is reserved solely for cache-based queries and is re-used only if another **SELECT** query matches identically. Therefore, the total number of Oracle connections that an existing map requires to run can increase when you add the **-CACHE** option to the adapter command.
- [Enabling the Oracle adapter to use the Oracle client result cache](#)
Enable the client result cache on the Oracle server, and configure the Oracle adapter command and the SQL **SELECT** query to use the cache.

SQL SELECT query requirements

SELECT queries must be syntactically identical to retrieve results from the Oracle client result cache and to reuse database connections. Use bind variables to create syntactically identical **SELECT** queries.

Related information

- [Using bind values in database functions](#)
-

Tuning Oracle adapter database connections

The Oracle adapter **-CACHE** option defines a unique database connection for each **SELECT** statement. This connection is reserved solely for cache-based queries and is re-used only if another **SELECT** query matches identically. Therefore, the total number of Oracle connections that an existing map requires to run can increase when you add the **-CACHE** option to the adapter command.

For example, a map that doesn't use the **-CACHE** option uses one database connection for three unique **SELECT** queries. When you add the **-CACHE** option to each **SELECT** statement, the map requires a database connection for each unique **SELECT** statement (a total of three connections). Two instances of this map that run in parallel require six database connections (instead of two connections without the **-CACHE** option).

Opening a database connection adds to the total time to complete the query, but keeping a connection open consumes memory on the server. You need to tune the number of open database connections to maximize connection reuse and minimize the number of open but unused connections.

You can use the `/runtime/Connection Manager/DB` path of the config.yaml configuration file to limit the number of idle database connections. The `/runtime/Connection Manager/DB/sLim` path limits the number of idle database connections, but that limit can be exceeded during peak load. The optimal number of database connections is affected by the number of connections that each map requires and the number of running instances of each map.

To count the number of connections that a map requires, enable the **-AUDIT** adapter command with the global option (**-AG**) on the first database operation in the map. The **-AG** command audits all subsequent database operations in the map.

The audit log includes the following information about a map database query:

Attribute	Description
Connections	A list of unique values. Each unique value represents a connection.
Time	The total elapsed time for the database adapter to process and execute the query.
Database	The portion of the total time for the Oracle driver to return a query result set.

After a query runs and the result is cached, the total elapsed time of subsequent identical queries is reduced, whether the queries are in the same map or in other maps.

Related information

- [-AUDIT adapter command](#)
-

Enabling the Oracle adapter to use the Oracle client result cache

Enable the client result cache on the Oracle server, and configure the Oracle adapter command and the SQL **SELECT** query to use the cache.

Procedure

- On the Oracle server, set the `CLIENT_RESULT_CACHE_SIZE` initialization parameter and restart the database. For example:

```
alter system set client_result_cache_size=4M scope=spfile
```

See your Oracle documentation for details and for other Oracle server initialization parameters.

- Add the **-CACHE** option to the Oracle adapter command line on the input card or on the **GET**, **DBLookup**, or **DBQuery** functions.
- Add the SQL `RESULT_CACHE` hint to the **SELECT** query. Ensure that there are no spaces between the asterisk (*) character and the and plus (+) character in the hint. For example:

```
=GET("DB","-CACHE -STMT ""SELECT /*+ RESULT_CACHE */ name FROM employee WHERE id=:bind('12')"" -DBTYPE ORACLE -CONNECT ORA -U u1 -P p1")
```

Oracle Adapter commands

This topic describes the functions and usage of the Oracle Adapter commands and their options.

- [Adapter properties and commands](#)

This topic describes the functions and usage of the Oracle Adapter commands and their options.

- [Adapter commands for a source](#)

- [Adapter commands for a target](#)

- [Database triggers using Oracle](#)

Adapter properties and commands

This topic describes the functions and usage of the Oracle Adapter commands and their options.

Adapter command summary

Use the adapter commands when specifying data sources and targets. The applicability of many of the commands depends upon whether you are specifying a source or target, whether a database/query file (.mdq) is used, and the situations in which the command applies.

Adapter commands can be used in GET->Source->Command or PUT->Target->Command settings in the Map Designer and Integration Flow Designer, using GET, PUT, DBLOOKUP, or DBQUERY function calls, or overriding a data source or target using execution commands in a RUN function or on the command line.

The following adapter commands are Oracle-specific database parameters.

Connect string

Specifies the host connect string. The corresponding adapter command is **-CONNECT**.

-CONNECT *connect_string*

connect_string: Specify the host connect string. If an SQL*Net host connection string is specified, the connection is established through SQL*Net. If none is specified, a direct connection is established to the database identified by the ORACLE_SID environment setting.

Database adapter type

This property specifies the database adapter type. The corresponding adapter command is **-DBTYPE**.

This command must be specified if the original card is not a database and no database is specified using the Database/Query adapter command (-MDQ) and the Database Name adapter command (-DBNAME).

-DBTYPE ORACLE

ORACLE: The database adapter type is Oracle.

Client result cache

Enables the Oracle adapter to retrieve the results of a **SELECT** query from the Oracle client result cache. The corresponding adapter command is **-CACHE**.

Trigger

Specifies the trigger string containing the options defined for a trigger specification. The corresponding adapter command is **-TR** (or **-TRIG**). This command is used to specify the trigger string for the trigger specification.

-TRIG *trigger_string*

trigger_string: Specify the trigger string containing the options defined for a trigger specification.

Row count

Specifies the number of rows to be retrieved per fetch. The corresponding adapter command is **-ROWCNT**.

-ROWCNT *row_count*

row_count: Specify the number of rows to be retrieved per fetch.

The time to fetch is optimized in accordance with the increase in the number of rows to retrieve. Therefore, a significant performance improvement may be gained by retrieving as many rows as possible in a single fetch.

The number of rows retrieved is limited only by the amount of memory installed on the querying computer.

The Oracle Adapter default for the number of rows to be retrieved per fetch is based upon an allocation of 64K buffer. A calculation is then performed to indicate the number of rows that can be held in the buffer.

-INSERTALL

Enables multiple rows of data to be inserted into an Oracle table at once. The corresponding adapter command is **-IA** (or **-INSERTALL**).

By batching a user-specified number of rows, execution times can be improved, when network latency exists between the Oracle adapter and the Oracle server. This option uses the Oracle **INSERT ALL** database statement, which allows multiple inserts with one database command, reducing network roundtrips during the output card processing of multiple input records.

The **-INSERTALL** command can be specified on output cards and works only on tables containing supported Oracle built-in data types. Custom types like Abstract Data Types (ADTs) will force the adapter to use one database insert operation at a time. This limitation applies when using the Default On NULL Oracle capability, as activated by the **-DON** command.

The **-INSERTALL** database statement constructed by the Oracle adapter can be viewed in the log file.

By default, a batch of ten inserts is used. However, this number can be overridden on the command line, depending on the network configuration and deployment requirements. The ideal number for performance will vary based on the network setup. The performance of various values can be compared using the audit option on the command line

Adapter commands for a source

This summary shows the syntax of the adapter commands that can be used when defining a data source using an .mdq file or without using one, including both the required and optional adapter commands in the following situations:

- Using the GET->Source->Command setting in the Map Designer and Integration Flow Designer.
 - Overriding a data source using the Input Source Override - Database execution command (-ID) using a RUN function or on the command line.
 - Using a DBLOOKUP, DBQUERY, or GET function in map or component rules.
 - [GET > Source > Command setting](#)
 - [Database execution command: input source override \(-ID\)](#)
 - [DBLOOKUP and DBQUERY functions](#)
 - [GET function](#)
-

GET > Source > Command setting

In an input card, when you specify Database as the value for the GET->Source setting, the Command field is displayed where you can enter adapter commands.

Database command options when there is a database/query file:

```
[-DBTYPE ORACLE]
[-CACHE]
[-CONNECT connect_string]
[-STMT SQL_statement]
[-FILE [directory]]
[-VAR name=value...]
[-TRIG trigger_string]
[-USER user_ID]
[-PASSWORD password]
[-CSTMT [number]]
[-ROWCNT row_count]
[-CACHE]
[-GTX]
[-AUDIT[G] [+]
 [full_path]]
[{-TRACE|-TRACEERR} [+]
 [full_path]]
```

Database command options when there is not a database/query file:

```
-DBTYPE ORACLE
[-CACHE]
-STMT SQL_statement
[-CONNECT connect_string]
[-FILE [directory]]
[-VAR name=value...]
[-TRIG trigger_string]
[-USER user_ID]
[-PASSWORD password]
[-CSTMT [number]]
[-ROWCNT row_count]
[-CACHE]
[-GTX]
[-AUDIT[G] [+]
 [full_path]]
[{-TRACE|-TRACEERR} [+]
 [full_path]]
```

Database execution command: input source override (-ID)

Use the Input Source Override - Database execution command (-ID) to designate a database as the source or you can override one or more of the adapter command settings or database definitions in a RUN function or on the command line.

The adapter commands are shown using a single quotation mark, which is the Windows syntax. For non-Windows platforms, use two single quotation marks followed by one single quotation mark and end with one single quotation mark followed by two single quotation marks.

Scenario: compiled map source is a database

Database/query file (.mdq):

```
'[-MDQ mdq_file
-DBNAME database_name]
[-CACHE]
[-QUERY query_name]-STMT SQL_stmt]
[-FILE [directory]]
[-VAR name=value...]
[-TRIG trigger_string]
[-USER username]
[-PASSWORD password]
[-CCARD|-CSTMT [number]]
[-ROWCNT row_count]
[-GTX]
[-AUDIT[G] [+]
 [full_path]]
[{-TRACE|-TRACEERR} [+]
 [full_path]]'
```

No database/query file (.mdq):

```
'-DBTYPE ORACLE
[-CACHE]
```

```

-STMT SQL_statement
[-CONNECT connect_string]
[-FILE [directory]]
[-VAR name=value..]
[-TRIG trigger_string]
[-USER user_ID]
[-PASSWORD password]
[-CCARD|-CSTM [number]]
[-ROWCNT row_count]
[-GTX]
[-AUDIT[G] [+|] [full_path]]
[{-TRACE|-TRACEERR} [+|] [full_path]]'

```

Scenario: compiled map source is not a database

Database/query file (.mdq):

```

'[-MDQ mdq_file
-DBNAME database_name
[-CACHE]
[-QUERY query_name|-STMT SQL_stmt]
[-FILE [directory]]
[-VAR name=value..]
[-TRIG trigger_string]
[-USER username]
[-PASSWORD password]
[-CCARD|-CSTM [number]]
[-ROWCNT row_count]
[-GTX]
[-AUDIT[G] [+|] [full_path]]
[{-TRACE|-TRACEERR} [+|] [full_path]]'

```

No database/query file (.mdq):

```

'-DBTYPE ORACLE
[-CACHE]
-STMT SQL_statement
[-CONNECT connect_string]
[-FILE [directory]]
[-TRIG trigger_string]
[-USER user_ID]
[-PASSWORD password]
[-CCARD|-CSTM [number]]
[-ROWCNT row_count]
[-GTX]
[-AUDIT[G] [+|] [full_path]]
[{-TRACE|-TRACEERR} [+|] [full_path]]'

```

DBLOOKUP and DBQUERY functions

The DBLOOKUP and DBQUERY functions can be used in component rules in the Type Designer and map rules in the Map Designer when creating a map that can be used with a database.

Using DBLOOKUP with a database/query file

```

DBLOOKUP ("SQL_statement",
"-MDQ mdq_file
-DBNAME database_name
[-CACHE]
[-USER username]
[-PASSWORD password]
[-CCARD|-CSTM [number]]
[-ROWCNT row_count]
[-GTX]
[-AUDIT[G] [+|] [full_path]]
[{-TRACE|-TRACEERR} [+|] [full_path]]")

```

Using DBQUERY without a database/query file

```

DBQUERY ("SQL_statement",
"-DBTYPE ORACLE
[-CACHE]
[-CONNECT connect_string]
[-USER username]
[-PASSWORD password]
[-CCARD|-CSTM [number]]
[-ROWCNT row_count]
[-GTX]
[-AUDIT[G] [+|] [full_path]]
[{-TRACE|-TRACEERR} [+|] [full_path]]")

```

GET function

The GET function returns the data from the source adapter.

Using GET with a database/query file

```
GET ("DB",
"-MDQ mdq_file
-DBNAME database_name
[-CACHE]
{QUERY query_name|-STMT SQL_stmt}
[-FILE [directory]]
[-VAR name=value...]
[-USER username]
[-PASSWORD password]
[-CCARD|-CSTM [number]]
[-ROWCNT row_count]
[-GTX]
[-AUDIT[G] [+| [full_path]]
[{-TRACE|-TRACEERR} [+| [full_path]]])
```

Using GET without a database/query file

```
GET "DB",
"-DBTYPE ORACLE
[-CACHE]
-STM SQL_stmt
[-CONNECT connect_string]
[-FILE [directory]]
[-USER username]
[-PASSWORD password]
[-ROWCNT row_count]
[-CCARD|-CSTM [number]]
[-GTX]
[-AUDIT[G] [+| [full_path]]
[{-TRACE|-TRACEERR} [+| [full_path]]])
```

Adapter commands for a target

This summary shows the syntax of the adapter commands that can be used when defining a data target using an .mdq file or without using one, including both the required and optional adapter commands in the following situations:

- Using the PUT > Target > Command setting in the Map Designer and Integration Flow Designer.
 - Overriding a data source using the Output Source Override - Database execution command (-OD) using a RUN function or on the command line
 - Using the PUT function in map or component rules.
- [PUT > Target > Command setting](#)
 - [PUT function](#)
 - [Database execution command: output source override \(-OD\)](#)

PUT > Target > Command setting

In an output card, when you specify Database as the value for the PUT > Target setting, the Command field is displayed so that you can enter adapter commands.

Database command options when there is a database/query file

```
[-DBTYPE ORACLE]
[-CONNECT connect_string]
[-PROC procedure_name|-TABLE table_name]
[-USER user_ID]
[-PASSWORD password]
[-CSTM [number]]
[-DELETE]
[-UPDATE [OFF|ONLY]]
[-GTX]
[-BADDATA[+| full_path]
[-AUDIT[G] [+| [full_path]]
[{-TRACE|-TRACEERR} [+| [full_path]]])
```

Database command options when there is no database/query file

```
-DBTYPE ORACLE
-PROC procedure_name|-TABLE table_name
[-CONNECT connect_string]
[-USER user_ID]
[-PASSWORD password]
[-CSTM [number]]
[-DELETE]
[-GTX]
[-BADDATA[+| full_path]
[-AUDIT[G] [+| [full_path]]
[{-TRACE|-TRACEERR} [+| [full_path]]])
```

PUT function

Use the PUT function to pass data to the target adapter.

Using PUT with a database/query file

```
PUT ("DB",
"-MDQ mdq_file
-DBNAME database_name
-PROC procedure_name| -TABLE table_name
[-USER username]
[-PASSWORD password]
[-CCARD|-CSTMT [number]]
[-DELETE]
[-UPDATE [OFF|ONLY]]
[-GTX]
[-BADDATA[+] full_path]
[-AUDIT[G] [+][ full_path]]
[{-TRACE|-TRACEERR} [+][ full_path]]")
```

Using PUT without a database/query file

```
PUT "DB",
"-DBTYPE ORACLE
-PROC procedure_name| -TABLE table_name
[-CONNECT connect_string]
[-USER username]
[-PASSWORD password]
[-CCARD|-CSTMT [number]]
[-DELETE]
[-GTX]
[-BADDATA[+] full_path]
[-AUDIT[G] [+][ full_path]]
[{-TRACE|-TRACEERR} [+][ full_path]]")
```

Database execution command: output source override (-OD)

Use the Output Source Override - Database execution command (-OD) to designate a database as a target or you can override one or more of the adapter command settings or database definitions in a RUN function or on the command line.

The adapter commands are shown using a single quotation mark, which is the Windows syntax. For non-Windows platforms, use two single quotation marks followed by one single quotation mark and end with one single quotation mark followed by two single quotation marks.

Scenario: compiled map target is a database

Database/query file (.mdq):

```
' [-MDQ mdq_file -DBNAME database_name]
[-PROC procedure_name| -TABLE table_name]
[-USER username]
[-PASSWORD password]
[-CCARD|-CSTMT [number]]
[-DELETE]
[-UPDATE [OFF|ONLY]]
[-GTX]
[-BADDATA[+] full_path]
[-AUDIT[G] [+][ full_path]]
[{-TRACE|-TRACEERR} [+][ full_path]]'
```

No database/query file (.mdq):

```
'-DBTYPE ORACLE
-PROC procedure_name| -TABLE table_name
[-CONNECT connect_string]
[-USER user_ID]
[-PASSWORD password]
[-CCARD|-CSTMT [number]]
[-DELETE]
[-UPDATE [OFF|ONLY]]
[-GTX]
[-BADDATA[+] full_path]
[-AUDIT[G] [+][ full_path]]
[{-TRACE|-TRACEERR} [+][ full_path]]'
```

Scenario: compiled map target is not a database

Database/query file (.mdq):

```
'-MDQ mdq_file
-DBNAME database_name
-PROC procedure_name| -TABLE table_name
[-USER username]
```

```
[-PASSWORD password]  
[-CCARD|-CSTMT [number]]  
[-DELETE]  
[-UPDATE [OFF|ONLY]]  
[-GTX]  
[-BADDATA[+] full_path]  
[-AUDIT[G] [+]] [full_path]]  
[{-TRACE|-TRACEERR} [+]] [full_path]]'
```

No database/query file (.mdq):

```
'-DBTYPE ORACLE  
-PROC procedure_name| -TABLE table_name  
[-CONNECT connect_string]  
[-USER user_ID]  
[-PASSWORD password]  
[-CCARD|-CSTMT [number]]  
[-DELETE]  
[-UPDATE [OFF|ONLY]]  
[-GTX]  
[-BADDATA[+] full_path]  
[-AUDIT[G] [+]] [full_path]]  
[{-TRACE|-TRACEERR} [+]] [full_path]]'
```

Database triggers using Oracle

When using Oracle, the triggering installation scripts must be installed and run by your system administrator. For more information about these scripts, see the topic about database triggers in the Database Interface Designer documentation.

Bulk copy example files

The bulk copy function transfers data between a table and an operation system file at high speeds. It is the most common way to load large amounts of data into a database table.

It is frequently used to transfer data between database vendors, spreadsheet applications and servers.

The example for this adapter is located at *install_dir\examples\adapters\oracle*.

Restrictions and limitations

The Database Interface Designer and database adapters offer options and functions for accessing and manipulating data contained within a database.

There are no restrictions or limitations when using the Oracle Adapter. However, the BLOB and NCLOB data types are not supported as parameters for stored procedures.

Return codes and error messages

Return codes and messages are returned when the particular activity completes. Return codes and messages can also be recorded as specified in the audit logs, trace files, execution summary files, and so forth.

For information about error codes and messages returned by database-specific adapters, see the Resource Adapters documentation.

Various troubleshooting tools are available in case you encounter problems while using the database adapters. For example, if you attempt to run a map that uses the adapter and encounter problems or do not receive the expected results, use the following adapter troubleshooting tools:

- adapter audit log (.log)
- adapter trace file (.mtr)

Oracle AQ Adapter overview

The Oracle Advanced Queuing (AQ) Adapter handles the enqueueing (posting of messages) and dequeuing (removal of messages) to or from a queue based upon what has been specified using the adapter commands such as the receipt of a message or messages with specific characteristics defined

- [Oracle AQ message content](#)
- [System requirements](#)
- [Command alias](#)
- [Oracle AQ Adapter commands](#)
- [Syntax summary](#)
- [Data format](#)
- [Trace log](#)
- [Return codes and error messages](#)

Oracle AQ message content

The Oracle AQ is a first-in-first-out (FIFO) queue. Oracle AQ has the ability to support messages of two data types:

- Raw
- Object types

These are alternatively referred to as abstract data types (ADT). The object type is a composite type that includes Oracle data types such as number, date, varchar, and so on. Object types can include nested object types.

Types allowed in a message on a queue are controlled by the queue table to which the queue belongs. A queue table also controls the consumer type (single or multiple), default exception queue, message grouping, and default sorting option on dequeuing for all queues created within the queue table.

Individual queues control default delay for creation and message expiration date for all messages within the queue. A subscriber list can be configured for multi-consumer queues at the queue level. A subscriber can be another queue or an application identified by its agent name.

Queues can be propagated through an administrative interface. Subscription lists can be dynamically set and altered through the administrative interface as well.

When an individual message is enqueued, a specific recipient list that overrides subscription list setup for the queue can be specified, along with message grouping, correlation ID, and delay or expiration time for the message. Enqueuing also controls the priority and the relative position of the enqueued message. When a message is enqueued, Oracle assigns it a system-wide global ID (the Message ID) that is returned to the application.

Dequeuing specifications can include the relative position, correlation ID, and wait period of the message being dequeued. For multi-consumer queues, only a consumer whose name is on the recipient list can dequeue a message. Messages that are not dequeued before their expiration time are automatically moved into the exception queue. The adapters do not support the positioning of messages relative to a message ID.

System requirements

The minimum system requirements and operating system requirements for the Oracle AQ Adapter are detailed in the release notes. In addition to these requirements, the following are additional requirements to install and run the Oracle AQ Adapter:

- To install and run the Oracle AQ Adapter, the Oracle Enterprise Edition version and the Oracle AQ Adapter must be installed on the machine hosting the Command Server.
- When running on a UNIX system, the Oracle client creates a process whenever a connection is made to Oracle. If any of these processes terminates, defunct processes will remain in the process list. Although not required (because these do not cause problems nor do they consume resources), you can prevent this occurrence by adding the following line to the **sqlnet.ora** file in the **\$ORACLE_HOME/network/admin** path:

```
bequeath_detach=YES
```

The effect of this line will be to create these same connection processes as detached orphans, thus eliminating the possibility of them becoming defunct.

Command alias

Adapter commands can be specified by using an execution command string on the command line or by creating a command file that contains adapter commands dictating the desired execution settings. For information about all of the options you can use with the execution commands for adapters or how to create a command file, see the *Execution Commands* documentation.

Use the Input Source Override - Message execution command (**-IM**) and the Output Target Override - Message execution command (**-OM**) with the appropriate Oracle AQ Adapter-specific alias. The execution command syntax is:

```
-IM[ alias ] card_num  
-OM[ alias ] card_num
```

where **-IM** is the Input Source - Override execution command and **-OM** is the Output Target - Override execution command, *alias* is the adapter alias, and *card_num* is the number of the input or output card, respectively. The following table shows the adapter alias and execution commands.

Adapter	Alias	As Input	As Output
Oracle AQ	AQ	-IMAQcard_num	-OMAQcard_num

When using an adapter alias with the execution command, the Oracle AQ Adapter commands can be issued on the command line or in a command file. Use the adapter commands to specify adapter functions such as:

- Setting a priority level of a message.
- Identifying subscribers and recipients when using multi-consumer queues.
- Retrieving a logical message from a source queue with a correlation identifier.

For example, to override the adapter commands defined in output card 1, the command string for the adapter in an Oracle AQ environment might be:

```
-OMAQ1 '-Q DEV.ITQUE -C 4aq -US wb -PW y4i6 -EQ ERR.ITQUE -T'
```

This example execution command string defines the Oracle AQ Adapter as the data target. The required Queue Name adapter command (**-Q DEV.ITQUE**) specifies a queue named **DEV.ITQUE**. The required connect string **4aq** is specified using the Connect String adapter command (**-C 4aq**). The user ID **wb** is specified using the required User ID adapter command (**-US wb**). The password **y4i6** is specified using the required Password adapter command (**-PW y4i6**). The Exception Queue adapter command (**-EQ ERR.ITQUE**) specifies an exception queue **ERR.ITQUE**, in which the message is moved if it is not successfully processed. The Trace adapter command (**-T**) indicates that a trace file will be created to report adapter activity information, recording the events that occur while the adapter is retrieving this data.

Note: If the parameter of an adapter command used on the command line is a negative number, the value must be enclosed in double quotation marks so that it cannot be mistaken for another adapter command. For example, enter **-P "-1"** for a priority level of negative one.

- [GET > Source settings](#)
- [PUT > Target settings](#)

GET > Source settings

Use the Map Designer **GET** settings to configure options for input map cards that use the Oracle AQ Adapter. For the **GET > Source setting**, select OracleAQ to use the Oracle AQ Adapter.

- [GET > Source > Command setting](#)
- [Source > Transaction > OnSuccess setting](#)
- [Source > Transaction > OnFailure setting](#)
- [Source > Transaction > Scope setting](#)
- [Source > Transaction > Warnings setting](#)

GET > Source > Command setting

Use the **Command** setting to enter adapter commands. For the syntax of the commands, see ["Commands"](#).

Source > Transaction > OnSuccess setting

Use the **OnSuccess** setting with adapter sources to prevent deletion of the source message(s) from the server from which the messages originate. The default value is **Keep**.

Value	Description
Keep	Keep source messages on the server.
Delete	Delete source messages following a successful map completion.

Source > Transaction > OnFailure setting

Use the **OnFailure** setting with adapter sources to select the rollback (restore a previous version of the message) or commit behavior if the map does not complete successfully. The default value is **Rollback**.

Value	Description
Rollback	If the map does not successfully complete, sources retrieved through the adapter are not deleted nor are the targets stored.
Commit	The adapter behaves as if the map was successful, and the action (Keep or Delete) specified by the OnSuccess setting is taken.

Source > Transaction > Scope setting

Use the **Scope** setting to specify when to check for success (**OnSuccess**) or failure (**OnFailure**) of a map, card, or burst. This is so that the rollback and retry options can be performed as specified. The default value is **Map**.

Value	Description
Map	Check for success or failure at the completion of each map. If the map completes successfully, use the OnSuccess setting. If the map fails, use the OnFailure setting.
Burst	Check for success or failure at the completion of each burst. If the burst is successful, use the OnSuccess setting. If the burst fails, use the OnFailure setting.
Card	Check for success or failure at the completion of each card. If the card is processed successfully, use the OnSuccess setting. If the card fails, use the OnFailure setting.

Source > Transaction > Warnings setting

Use the **Warnings** setting to determine whether to fail the map or ignore warning conditions for the specific input map card. The default value is **Ignore**.

Value	Description
Ignore	

Ignore Ignore warnings for this map card.

Fail

Fail the map upon receiving a warning for this map card.

PUT > Target settings

Use the Map Designer **PUT** settings to configure options for output map cards that use the Oracle AQ Adapter. Use the **PUT > Target** setting to select the data target. Select OracleAQ to use the Oracle AQ Adapter.

- [PUT > Target > Command setting](#)
 - [Target > Transaction > OnSuccess setting](#)
 - [Target > Transaction > OnFailure setting](#)
 - [Target > Transaction > Scope setting](#)
 - [Target > Transaction > Warnings setting](#)
-

PUT > Target > Command setting

Use the **Command** setting to enter adapter commands. For the syntax of the commands, see "[Commands](#)".

Target > Transaction > OnSuccess setting

Use the **OnSuccess** setting with adapter targets to prevent a message from being unnecessarily created during any processes. The default value is Create.

Value

Description

Create

Upon successful completion of the burst, map, card, or rule (determined by the **Scope** settings), create the message.

CreateOnContent

Upon successful completion of the burst, map, card, or rule (determined by the **Scope** settings), create the message only if it is not an empty message.

Target > Transaction > OnFailure setting

Use the **OnFailure** setting with adapter targets to select the rollback or commit behavior if the map does not complete successfully. The default value is Rollback.

Value

Description

Rollback

If the map does not complete successfully, the target is not stored.

Commit

The adapter behaves as though the map was successful and the action (**Create** or **CreateOnContent**) specified by the **OnSuccess** setting is taken.

Target > Transaction > Scope setting

Use the **Scope** setting to specify when to check for success (**OnSuccess**) or failure (**OnFailure**) of a map, burst, or card so that the rollback and retry options can be performed as specified. The default value is Map.

Value

Description

Map

Check for success or failure at the completion of each map. If the map completes successfully, use the **OnSuccess** setting. If the map fails, use the **OnFailure** setting.

Burst

Check for success or failure at the completion of each burst. If the burst is successful, use the **OnSuccess** setting. If the burst fails, use the **OnFailure** setting.

Card

Check for success or failure at the completion of each card. If the card is processed successfully, use the **OnSuccess** setting. If the card fails, use the **OnFailure** setting.

Target > Transaction > Warnings setting

Use the **Warnings** setting to determine whether to fail the map or ignore warning conditions for the specific output map card. The default value is Ignore.

Value

Description

Ignore

Ignore warnings for this map card.

Fail

Fail the map upon receiving a warning for this map card.

Oracle AQ Adapter commands

When used with the Launcher, an Oracle AQ Adapter for a source or a target is configured using the Integration Flow Designer.

The adapter commands allow for connection to Oracle and control of the following Oracle AQ attributes:

- Consumer/producer agent names
- Queue names
- Message correlation IDs
- Message expiration and delay times
- Exception queues
- Wait periods on dequeuing
- Message's recipient list
- Message priority
- [List of commands](#)

List of commands

The following table lists valid commands for the MSMQ adapter, the adapter command syntax, and whether the adapter command is supported (✓) for use with data sources, data targets, or both.

Command	Syntax	Source	Target
Connect String (-C)	-C connect_string	✓	✓
Correlation ID (-CID)	-CID value	✓	✓
Database Name (-DBNAME)	-DBNAME database_name	✓	✓
Database/Query File (-MDQ)	-MDQ mdq_file	✓	✓
Delay (-D)	-D wait_time_in_sec		✓
Exception Queue (-EQ)	-EQ except_queue_name		✓
Expiration (-E)	-E exp_time_in_sec		✓
Global Transaction Management (-GTX) (Oracle, Oracle AQ, Microsoft SQL Server only)	-GTX	✓	✓
Header (-HDR)	-HDR	✓	
Immediate (-I)	-I	✓	✓
IP address (-IP)	-IP	✓	
Listen (-LSN)	-LSN {0 S wait_time_in_sec}	✓	
No Data (-NODATA)	-NODATA	✓	
Password (-PW)	-PW {password@[filename]}	✓	✓
PORT number (-PORT)	-PORT	✓	
Priority (-P)	-P level_number		✓
Quantity (-QTY)	-QTY {value S}	✓	
Queue Name (-Q)	-Q queue_name	✓	✓
Recipient (-R)	-R name name,name... @full_path or -R name	✓	✓
Sender (-S)	-S sender_name		✓
Trace (-T)	-T[E][+]{[path level]} [level path]}	✓	✓
User ID (-US)	-US user_ID	✓	✓

- [Connect string \(-C\)](#)
- [Correlation ID \(-CID\)](#)
- [Database name \(-DBNAME\)](#)
- [Database/query file \(-MDQ\)](#)
- [Delay \(-D\)](#)
- [Exception queue \(-EQ\)](#)
- [Expiration \(-E\)](#)
- [Global transaction management \(-GTX\)](#)
- [Header \(-HDR\)](#)
- [Immediate \(-I\)](#)
- [IP address \(-IP\)](#)
- [Listen \(-LSN\)](#)
- [No data \(-NODATA\)](#)
- [Password \(-PW\)](#)
- [PORT number \(-PORT\)](#)
- [Priority \(-P\)](#)
- [Quantity \(-QTY\)](#)
- [Queue name \(-Q\)](#)
- [Recipient \(-R\)](#)
- [Sender \(-S\)](#)

- [Trace \(-T\)](#)
- [User ID \(-US\)](#)

Connect string (-C)

Use the Connect String adapter command (-C) to specify the connection string service name or Oracle service.

-C connect_string

Option

Description

connect_string

This is the connection string service name or Oracle service.

The Connect String adapter command (-C) is required for data sources and data targets.

Correlation ID (-CID)

Use the Correlation ID adapter command (-CID) to specify a particular correlation identifier (CID) for a source or target. For a data source, use this command to retrieve messages from a queue with a specific correlation ID. For a data target, use this command to assign a correlation ID to a message when placing it on a queue.

-CID value

Option

Description

value

This is the correlation identifier used for a source or target.

A correlation identifier is an alphanumeric string up to 128 characters in length.

The * wildcard character can be specified as a data source and is mapped to the Oracle wildcard character %. For more information about wildcard notation, see the *Launcher* documentation.

Database name (-DBNAME)

Use the Database Name adapter command (-DBNAME) to specify the name of a database defined in the database/query file (.mdq) that is specified using the Database/Query File adapter command (-MDQ). The Database Name adapter command can be used for a source or target or in a GET or PUT function.

-DBNAME database_name

The *database_name* you supply is case-sensitive.

Database/query file (-MDQ)

Use the Database/Query File adapter command (-MDQ) to specify the name of the database/query file (.mdq) that contains the database definition for the database to be used (as specified using the -DBNAME adapter command). This Database/Query File adapter command can be used for a source or target or in a GET or PUT function.

-MDQ mdq_file

When this adapter command is used, the .mdq file must be available at map run time. Also, if the database/query file is not in the same directory as the compiled map file, the full path name is required.

Note: Using the combination of the Database Name adapter command (-DBNAME) and the Database/Query File adapter command (-MDQ) provides the same information as the combination of the Connect String adapter command (-C), the User Name adapter command (-US), and the Password adapter command (-PW). If both sets of commands exist in the command line, the -C, -US, and -PW combination takes precedence.

Delay (-D)

Note: The Delay adapter command (-D) requires that the queue manager be started.

Use the Delay adapter command (-D) to specify the delay of the enqueued message for data targets.

By default, when a message is enqueued, it is immediately available for dequeuing. Use the Delay adapter command (-D) for data targets to specify a length of time to wait (in seconds) after which a message is available for dequeuing.

-D wait_time_in_sec

Option

Description

wait_time_in_sec

This is the length of time to wait (in seconds), after which a message is available for dequeuing.

Exception queue (-EQ)

Use the Exception Queue adapter command (-EQ) for data targets to specify an exception queue to which a message is moved if it is not processed successfully.

-EQ except_queue_name

Option

Description

except_queue_name

This is the exception queue to which a message is moved if it is not processed successfully.

Messages are moved to the exception queue when either of the following occurs:

- The number of unsuccessful consumer dequeue attempts exceeds the permitted limit.
- The message has reached the expiration time set by the Expiration adapter command (-E).

Expiration (-E)

Note: The Expiration adapter command (-E) requires that the queue manager be started.

Use the Expiration adapter command (-E) for data targets to specify the length of time (in seconds) that a message is available on the queue before it expires. If the message is not dequeued within the specified time, the message expires and is moved to the exception queue.

-E exp_time_in_sec

Option

Description

exp_time_in_sec

This is the length of time (in seconds) that a message is available on the queue before it expires.

If the Delay adapter command (-D) is used to specify a delay time, the expiration time begins after the delay time has lapsed. For example, if a delay time of two seconds is specified with a five-second expiration time, the message expires in seven seconds and is moved to the exception queue.

Global transaction management (-GTX)

Use the Global Transaction adapter command (-GTX) to indicate that the operations on a card (whether input or output) should be processed within a global transaction.

-GTX

For more information about global transaction management, see the Global Transaction Management documentation.

Header (-HDR)

Use the Header adapter command (-HDR) for data sources to specify inclusion of header information with the actual message.

-HDR

For data targets, the Header adapter command (-HDR) is not necessary. The adapter detects the presence of a header in the target data. The header data overrides any values specified for the corresponding elements with adapter commands.

The following header elements are used to return message attributes for data sources or to specify message attributes for data targets.

Attribute	For a Data Source	For a Data Target
Priority	This returns the priority with which the message was enqueued.	This specifies the message priority and overrides the Priority adapter command (-P).
Delay	This returns the delay with which the message was enqueued.	This specifies the message delay and overrides the Delay adapter command (-D).
Expiration	This returns the expiration with which the message was enqueued.	This specifies the message expiration and overrides the Expiration adapter command (-E).
Correlation	This returns the correlation ID of the received message.	This specifies the message correlation ID and overrides the Correlation ID adapter command (-CID).
Exception Queue	This value is always NONE.	This specifies the message exception queue and overrides the Exception Queue adapter command (-EQ).
Sender	This value is always NONE.	This specifies the sender of the message and overrides the Sender adapter command (-S).
Enqueue Time	This returns the time at which the message was enqueued. The format is: <i>YYYY-MM-DD HH24:MI:SS</i>	On output, this value is ignored.
Recipients	This value is always NONE.	This specifies the comma-separated list of recipients of the message and overrides the Recipient adapter command (-R).

The format of header information is contained within an XML tag <Header> as follows:

```
<Header>
<Priority>... </Priority>
```

```
<Delay>... </Delay>
<Expiration>... </Expiration>
<Correlation>... </Correlation>
<ExceptionQueue>... </ExceptionQueue>
<Sender>... </Sender>
<EnqueueTime> ... </EnqueueTime>
<Recipients>... </Recipients>
</Header>
```

Immediate (-I)

Use the Immediate adapter command (`-I`) for data sources and data targets to dictate immediate enqueueing or dequeuing of the message.

`-I`

When the Immediate adapter command (`-I`) is specified, a message is acted upon immediately (enqueued or dequeued). The default behavior is to wait for successful map completion and current transaction completion.

When the Immediate adapter command (`-I`) is specified, it is not possible to roll back based upon the success status of the map.

IP address (-IP)

Use the IP address adapter command (`-IP`) to specify the IP address of the network interface to listen for incoming notifications from subscriptions to which the adapter is subscribed.

`-IP`

This adapter command is applicable only when using the adapter under the Launcher and when the adapter is configured as a watch for the events. By default, the adapter listens on all the available network interfaces. In addition to the `-IP` adapter command, another way to explicitly specify the IP address is by using the `WTX_AQ_SUBSCRIBER_IP` environment variable for the Launcher process.

When a value is specified through both the adapter command and the environment variable, the adapter command value takes precedence.

Listen (-LSN)

Use the Listen adapter command (`-LSN`) for data sources to indicate the length of time (in seconds) that the Oracle AQ Adapters wait for a message to be received.

`-LSN {0|S|wait_time_in_sec}`

Option

Description

0

Do not wait at all. If there is no input available, the adapter does not wait for more messages.

S

This is an infinite wait time.

`wait_time_in_sec`

This is the number of seconds that the adapter waits for a message.

For example, to specify the wait time of 45 seconds, the syntax would be:

`-LSN 45`

Map execution is suspended until the message(s) is/are received or until the specified time lapses, at which time a warning message is returned. The warning `No messages were found` is returned if there are no messages. The warning `Fetch unit not reached` is displayed if fewer messages were found than specified with the `FetchUnit` setting in the Map Designer.

No data (-NODATA)

Use the No Data adapter command (`-NODATA`) to return only the message header from the queue but not return its data. This is an alternative to the `-HDR` option.

`-NODATA`

Note that the `OnSuccess` setting is ignored if this option is specified, since it is not possible to delete a message if this option is specified.

Password (-PW)

Use the required Password adapter command (`-PW`) to specify the user authorization for connection to Oracle.

`-PW {password} @filename`

Option

Description

password

This is the string that specifies the user authorization.

@filename

This is the name of the file containing the password. No spaces are allowed in the filename.

The `password` option corresponds to the user specified by the User ID adapter command (`-US`). The password can be specified directly or from a file. Using the `filename` method provides greater password security.

For example, to specify a password of `007`, the syntax would be:

```
-pw 007
```

To specify the password contained in the `jbond` file, the syntax would be:

```
-pw @jbond
```

With regard to the format of the password in the password file, the password must be specified in the first line of this file and can contain both leading and trailing white space characters that are removed during processing. Each line is truncated to 128 characters.

The Password adapter command (`-PW`) is required for both data sources and data targets.

PORT number (-PORT)

Use `-PORT` number adapter command to specify the port number of the network interface to listen for incoming notifications from subscriptions to which the adapter is subscribed.

`-PORT`

This adapter command is applicable only when using the adapter under the Launcher and when the adapter is configured as a watch for the events. By default, the adapter listens on a random port dynamically assigned by Oracle AQ. In addition to the `-PORT` adapter command, another way to explicitly specify the port number is by using the `WTX_AQ_SUBSCRIBER_PORT` environment variable for the Launcher process.

When a value is specified through both the adapter command and the environment variable, the adapter command value takes precedence.

Priority (-P)

Use the Priority adapter command (`-P`) to set the priority level of a message for data targets.

```
-P level_number
```

Option

Description

level_number

The priority can be any number including negative numbers. The lower the number is, the higher is its priority.

When messages with the same priority are placed on a queue, the message enqueued earlier is dequeued first. However, when messages have different priorities, the message assigned the highest priority (lowest number) is dequeued first, regardless of when it was enqueued.

Quantity (-QTY)

Use the Quantity adapter command (`-QTY`) to specify the number of messages to retrieve from the source queue. If the Quantity adapter command (`-QTY`) is not specified, the default value is `1`.

```
-QTY {value|S}
```

Option

Description

S

This returns all messages on the queue.

value

This is a positive integer representing the total number of messages to be retrieved.

For example, to specify a quantity of ten messages, the syntax would be:

```
-QTY 10
```

The Quantity adapter command (`-QTY`) defines the maximum number of messages across all bursts for a map.

When the Quantity adapter command (`-QTY`) is used, the message cursor is not returned to the head of the queue for other source cards in the same map that use the same queue. The number of bursts in a single map invocation can be controlled using the combination of values set in the Map Designer SourceRule `> FetchAs > FetchUnit` setting and the Quantity adapter command (`-QTY`).

When using the Quantity adapter command (`-QTY`) with the `S` option through either the Launcher or the Command Server, the Listen adapter command (`-LSN`) is also required with a value specified other than `S`.

Queue name (-Q)

Use the Queue Name adapter command (-Q) to specify the name of the queue for data sources and data targets.

-Q queue_name

Option

Description

queue_name

This is the name of the queue for data sources and data targets.

The Queue Name adapter command (-Q) is required for data sources and data targets.

The queue specified with the Queue Name adapter command (-Q) cannot be an exception queue.

Recipient (-R)

Use the Recipient adapter command (-R) for data sources and data targets to identify the subscribers and recipients when multi-consumer queues are used.

-R {name|name,name,...|@full_path}

or

-R name

Option

Description

name

For a data source, specify the name of one recipient. Only messages sent to this specified recipient are received.

name, name, ...

For a data target only, specify a list of subscribers delimited with commas (no spaces).

@full_path

For a data target only, specify the full path and file name of a file that contains a list of recipients with each recipient's name appearing on a separate line.

For a data target, specify the name of a single recipient or multiple recipients. The file name must be specified using a full path or a relative path to the directory that contains the map file and must be preceded by the "at" sign (@). For example:

-R @c:\inventory\bikes\tricycles.txt

or

-R @/customer/western/stetson.lst

Regarding the format of the map file, each recipient name should be on a separate line in the file. There can be leading and trailing white space characters that are removed during processing. Each line is truncated to 128 characters.

Sender (-S)

Use the Sender adapter command (-S) for data targets only. This identifies the original sender of a message.

-S sender_name

Trace (-T)

Use the Trace adapter command (-T) to produce a log file with the default name **m4aq.log** in the map execution directory. The Trace adapter command (-T) is used with data sources or data targets to enable adapter tracing.

-T[E] [+| [path] [level]

Option	Description						
E	Produce a trace file containing only the adapter errors that occurred during map execution.						
+	This appends the current trace file to an existing trace file. Without the +, the existing trace is deleted and the new trace is written to an empty file.						
path	This is the relative or full path name of the trace file location. If no name is specified, the default file name is m4aq.log .						
level	This indicates the level of information provided in the trace file. The lower the number, the more basic the information. As level numbers increase, additional information is provided in the trace file. (See the available values below.) If no value is specified, the default is 0. <table border="1"><thead><tr><th>Level</th><th>Description</th></tr></thead><tbody><tr><td>0 or 1</td><td>Provides recorded detailed information about the adapter activity such as messages retrieved, SQL statements executed, errors returned (if any), and so on.</td></tr><tr><td>2 or higher</td><td>In addition to the information provided for a 0 level trace, this level provides information about processing individual object fields.</td></tr></tbody></table>	Level	Description	0 or 1	Provides recorded detailed information about the adapter activity such as messages retrieved, SQL statements executed, errors returned (if any), and so on.	2 or higher	In addition to the information provided for a 0 level trace, this level provides information about processing individual object fields.
Level	Description						
0 or 1	Provides recorded detailed information about the adapter activity such as messages retrieved, SQL statements executed, errors returned (if any), and so on.						
2 or higher	In addition to the information provided for a 0 level trace, this level provides information about processing individual object fields.						

The trace file contains information about internal adapter states.

User ID (-US)

Use the User ID adapter command (`-US`) to specify the user ID for connection to Oracle.

`-US user_ID`

Option

Description

`user_ID`

This is the user name for connection to Oracle.

The User ID adapter command (`-US`) is required for data sources and data targets.

Syntax summary

This topic displays the syntax of the Oracle AQ Adapter commands for data sources and targets.

- [Data sources](#)
 - [Data targets](#)
 - [Oracle AQ Adapter example files](#)
-

Data sources

The following is the syntax of the Oracle AQ Adapter commands used for data sources:

```
-Q queue_name -C connect_string -US user_ID -PW {password|@filename}
[-GTX]
[-NODATA]
[-MDQ [mdq_file]]
[-DBNAME [database_name]]
[-I]
[-HDR]
[-CID value]
[-LSN {0|S|wait_time_in_sec}]
[-QTY {value|S}]
[-R name]
[-S]
[-T[E] [+]] [path] [level]]
```

Data targets

The following is the syntax of the Oracle AQ Adapter commands used for data targets:

```
-Q queue_name -C connect_string -US user_ID -PW {password|@filename}
[-GTX]
[-MDQ [mdq_file]]
[-DBNAME [database_name]]
[-I]
[-P level_number]
[-D wait_time_in_sec]
[-EQ except_queue_name]
[-E exp_time_in_sec]
[-CID value]
[-R {name|name,name,...|@full_path}]
[-S]
[-T[E] [+]] [path] [level]]
```

Oracle AQ Adapter example files

To access the adapter example files, navigate to `install_dir/examples/adapters/aq`.

An Oracle AQ type tree is provided. The **simple.mtt** type tree is suitable for receiving and sending simple messages.

- [Connection sharing and transactions](#)
-

Connection sharing and transactions

The Oracle AQ Adapters share connections among multiple Oracle AQ cards based upon the user name, password, service name, rollback option and thread of execution. This is done to improve the performance of the interface.

Data format

The Oracle AQ Adapter formats message data as infix delimited, using a vertical bar (|) as the delimiter. Groups (ADT types) are denoted with a left bracket as the initiator ([:]) and a right bracket as the terminator (:]) to facilitate detection of null groups. (A null group indicates that the entire ADT is null, not just individual members within the group.)

Only the nested ADTs are initiated or terminated; the top ADT group is not.

Because a vertical bar (|), left bracket ([), and right bracket (]) are special characters, when these characters are present in any message, they are released with an exclamation point (!) release character. This process should be transparent to the user.

Note: If an ADT type queue is used, one consisting of only simple types (no nested ADTs), the data itself must not contain any left bracket ([) or right bracket (]) special characters (the adapter releases them, but the type tree will not recognize them).

Using the Database Interface Designer with the Oracle adapters, you can create a type tree to reflect the message types and data format expected by the Oracle AQ Adapter.

Trace log

Use the Trace adapter command (-T) to create a trace file that reports adapter activity information, recording the events that occur while the adapter is retrieving and sending data. The trace command produces a log file with the default name **m4aq.log** file in the map execution directory.

The trace file contains information about connections to the Oracle AQ messaging system, message sizes, error messages, and adapter error messages.

Return codes and error messages

Return codes and messages are returned when the particular activity completes. Return codes and messages might also be recorded as specified in the audit logs, trace files, execution summary files, and so on.

- [Oracle AQ Adapter messages](#)

Oracle AQ Adapter messages

The following is a listing of all the codes and messages that can be returned as a result of using the Oracle AQ Adapter for sources or targets.

In addition to the following return codes and messages, the Oracle AQ Adapter can return Oracle AQ error codes. Consult the documentation from Oracle Corporation about the AQ messaging system for descriptions of these codes.

Note: Adapter return codes with positive numbers are warning codes that indicate a successful operation. Adapter return codes with negative numbers are error codes that indicate a failed operation.

Table 1. Oracle AQ Adapter error codes and messages

Return Code	Message
-1001	Invalid command line option
-1002	Missing command line argument
-1003	Invalid command line argument
-1004	Command line processing failed
-1005	Invalid queue name
-1101	Unsupported data type
-1102	Invalid release character position

PDF Adapter

The PDF Adapter works with Acrobat forms and templates to generate a PDF document or extract data from all or part of an input PDF document.

In Design Studio, use the PDF importer (File > Import > PDF) to generate a native XML schema type tree and an associated name-lookup file from the Acrobat form (AcroForm) of a PDF document. The name-lookup file is required to generate formatted PDF output. The name-lookup file maps each type in the schema to the corresponding AcroForm field name, as follows:

TX_type_name=AcroForm_field_name

You can use the native XML schema type tree for mapping, but do not edit the names in the schema. To change schema field names, edit the AcroForm and import the native schema (.xsd) type tree from the PDF document again. On output, the PDF Adapter can generate a formatted PDF file from an Acrobat form, or Acroform template. Without a template, the adapter can create a PDF file that contains unformatted text.

- [System requirements](#)

See the [release notes](#) for the PDF Adapter requirements.

- [PDF Adapter commands](#)

System requirements

See the [release notes](#) for the PDF Adapter requirements.

PDF Adapter commands

- [**Command aliases**](#)
The PDF Adapter works with Acrobat forms and templates to generate a PDF document or extract data from all or part of an input PDF document.
- [**Transaction scope and document persistence in memory**](#)
When a document is added using the -ADOC command, the PDF Adapter stores the document in memory. Documents stored in memory are identified by a document identifier that is either provided by the user, or generated by the PDF Adapter. The document persists in memory as specified by the transaction scope in the card settings.
- [**The data_from_map option**](#)
Some of the PDF Adapter action commands that follow include the *data_from_map* option. The *data_from_map* option represents the data that is passed as the third parameter of a **GET** function. It is not a parameter on the adapter command line.
- [**Add document \(-ADOC\) command**](#)
The **-ADOC** command adds a file on the local file system, or a remote file, accessible through **http** protocol, on the file system as a document in memory and returns the document ID to the map. Alternatively, the **-ADOC** command can add dynamic data from a map as a document in memory and return the document ID to the map.
- [**Remove document \(-RDOC\) command**](#)
The **-RDOC** command removes a document from memory. By default, documents are removed from memory when a map, or card, processing completes. Transaction scope card setting determines the document persistence in memory. Use this command if too many intermediate documents accumulate in memory during map processing.
- [**Replace document \(-UDOC\) command**](#)
The **-UDOC** command replaces an existing document in memory with a new one and returns the new document ID.
- [**Get document data \(-GDOC\) command**](#)
The **-GDOC** command returns data from a document in memory.
- [**Filename \(-URL\) command**](#)
- [**Schema \(-SCHEMA\) command**](#)
- [**Page \(-PAGE\) command**](#)
- [**PDF Template \(-TEMPLATE\) command**](#)
- [**Name lookup file \(-NAMES\) command**](#)
- [**Append data \(-APPEND\) command**](#)
- [**PDF transformation \(-PDF\) command**](#)
The **-PDF** command allows the map to transform map data in XML format to PDF data using the **GET** map rule. **PUT** receives map data in XML format and transforms it to formatted PDF data. This command allows the same with **GET**, so that it can return PDF data back to the map, by performing in-memory transformation. The XML format data must conform to the PDF importer generated, native XML schema type tree, for the corresponding PDF document

Command aliases

The PDF Adapter works with Acrobat forms and templates to generate a PDF document or extract data from all or part of an input PDF document.

Use **PDF** as the adapter command alias on input and output card overrides and in **GET** and **PUT** rules. For example:

Input source override execution command	-IAPDF card_num
Output target override execution command	-OAPDF card_num

Inbound adapter examples

Command line:

```
-U JohnDoe1040.pdf -S f1040.xsd
```

GET rule:

```
=PARSE(GET("PDF", "-S f1040.xsd", BlobIn))
```

Command line for reading pages:

```
-U txredbook.pdf -PAGE 22-23  
-U txredbook.pdf -PAGE 22
```

Outbound adapter commands

Command line with template:

```
-U My1040.pdf -N f1040.names -P f1040.pdf
```

PUT rule:

```
=PUT("PDF", "-U My1040.pdf -N f1040.names -P f1040.pdf -T", Input)
```

Transaction scope and document persistence in memory

When a document is added using the -ADOC command, the PDF Adapter stores the document in memory. Documents stored in memory are identified by a document identifier that is either provided by the user, or generated by the PDF Adapter. The document persists in memory as specified by the transaction scope in the card settings.

- When the transaction scope is MAP, the document is accessible in memory to later map rules in the same map. The document is removed from memory when the map completes processing.
- When the transaction scope is CARD, the document is accessible in memory to later map rules on the same card. The document is removed from memory when the card completes processing, and is not accessible to map rules in later cards.

The *data_from_map* option

Some of the PDF Adapter action commands that follow include the *data_from_map* option. The *data_from_map* option represents the data that is passed as the third parameter of a **GET** function. It is not a parameter on the adapter command line.

If both **-URL** and *data_from_map* options are provided, then *data_from_map* has precedence over the **-URL** option.

Add document (-ADOC) command

The **-ADOC** command adds a file on the local file system, or a remote file, accessible through **http** protocol, on the file system as a document in memory and returns the document ID to the map. Alternatively, the **-ADOC** command can add dynamic data from a map as a document in memory and return the document ID to the map.

-ADOC command syntax



-URL

Adds a single file, specified by a local file URL (**file:///**), or a remote file URL (**http://**), as a document in memory and returns a document ID to the map. The **-URL** keyword is optional, but you must specify either the **-URL**, or the *data_from_map* option on the **-ADOC** command. **doc_ID** is an optional parameter. If specified, the **-ADOC** command associates the user-specified identifier for the document in memory and returns the same. If not specified, the **-ADOC** command generates a unique document identifier for the document in memory and returns the same.

data_from_map

Adds dynamic data from a map as a document in memory and returns the document ID to the map. The *data_from_map* is optional, but you must specify either the **-URL**, or the *data_from_map* option in the **-ADOC** command.

Examples

The following example adds the `parsewds.txt` file as a document in memory and returns the document ID to the map:

```
=GET("PDF", "-URL file:///C:\parsewds.txt -ADOC")
```

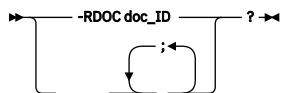
The following example adds the `Hello World!!` text string as a document in memory and returns **mydocid** to the map:

```
=GET("PDF", "-ADOC mydocid", "Hello World!!")
```

Remove document (-RDOC) command

The **-RDOC** command removes a document from memory. By default, documents are removed from memory when a map, or card, processing completes. Transaction scope card setting determines the document persistence in memory. Use this command if too many intermediate documents accumulate in memory during map processing.

-RDOC command line syntax



doc_ID

doc_ID indicates the document identifier of a document to be deleted from memory, if needed.

Replace document (-UDOC) command

The **-UDOC** command replaces an existing document in memory with a new one and returns the new document ID.

-UDOC command syntax



-URL

Specifies the local file URL (**file:///**), or remote file URL (**http://**) of the document that is to replace the document in memory. Either the **-URL** or the *data_from_map* option is required.

data_from_map

Replaces the document in memory with dynamic data from a map. Either the *data_from_map* or the -URL option is required.

Get document data (-GDOC) command

The **-GDOC** command returns data from a document in memory.

-GDOC command syntax

►► -GDOC doc_ID ►►

Filename (-URL) command

-U *filename.pdf*

-URL *filename.pdf*

The -URL command specifies a URL to a local file on the file system (**file:///**), a remote file on a host server (**http://**), or a document in memory (**memory://**). If you do not specify either a local or remote file URL, the adapter treats the PDF document file path as relative to the map directory. Use this command for data sources and data targets.

The -URL command is optional for **GET** rules, but the command is required on input cards. For example, a map can pass PDF Acroform data as a blob to the PDF Adapter, and the adapter can extract data from the blob in XML format.

The -URL command is required on output cards and **PUT** rules.

Schema (-SCHEMA) command

-S *schema.xsd*

-SCHEMA *schema.xsd*

Use this command for data sources and data targets.

On input cards and **GET** rules, the -SCHEMA command specifies the name of the XML schema file that the **GET** transaction is to return with the PDF data. You must specify either -SCHEMA or -PAGE on input cards and **GET** rules. The -SCHEMA and -PAGE command are mutually exclusive.

On output cards and **PUT** rules, the -SCHEMA command specifies the XML schema used to parse XML data when generating a PDF document. The -SCHEMA command is optional on output cards and **PUT** rules. If you omit both the -SCHEMA and -TEMPLATE commands, the adapter creates a PDF document of unformatted text.

Page (-PAGE) command

-PAGE { *nn* | *nn-nn* }

Specifies a page or a range of pages to read from a PDF document and send to the map as a text blob. Use this command on data sources. This command is optional. You must specify either -SCHEMA or -PAGE on input cards and **GET** rules. The -SCHEMA and -PAGE command are mutually exclusive.

PDF Template (-TEMPLATE) command

-P *template_name.pdf*

-TEMPLATE *template_name.pdf*

The PDF template to use to generate a PDF document. Use this command for data targets. This command is optional. If you omit both the -SCHEMA and -TEMPLATE commands, the adapter creates a PDF document containing unformatted text.

Name lookup file (-NAMES) command

-N *filename.names*

-NAMES *filename.names*

The name-lookup file created by importing a PDF document. The .names file maps the AcroForm field names to the types in the schema. Use this command for data targets. This command is required on output cards and **PUT** rules to create a formatted PDF document. If you omit the -NAMES command, the generated PDF document contains unformatted text.

Append data (-APPEND) command

-APPEND

Appends data to the specified PDF document. Use this command for data targets. This command is optional.

PDF transformation (-PDF) command

The **-PDF** command allows the map to transform map data in XML format to PDF data using the **GET** map rule. **PUT** receives map data in XML format and transforms it to formatted PDF data. This command allows the same with **GET**, so that it can return PDF data back to the map, by performing in-memory transformation. The XML format data must conform to the PDF importer generated, native XML schema type tree, for the corresponding PDF document

This command is useful in environments where file system access is restricted to the users, and PDF transformations must be performed in memory and then the transformed output to be returned to the map for further processing. A stream of PDF data is passed to the map with GET:

```
=GET("PDF","-PDF -NAMES memory://<names-document-id> -TEMPLATE memory://<template-document-id>",<xml-map-data>)
```

A template must be added to memory, if it is not a remote file, using the **-ADOC** command when performing in-memory transformation of XML formatted data to PDF formatted data. The **-ADOC** command returns a user specified document ID, or system ID for the template.

Note: The document in memory is referenced as: **memory://**

This three step process is shown here:

- **template-document-id=GET("PDF","-ADOC",<template-file-data-as-stream>)** or **=GET("PDF","-ADOC <my-template-documentid>",<template-file-data-as-stream>)**
- **names-document-id=GET("PDF","-ADOC",<names-file-data-as-stream>)** or **=GET("PDF","-ADOC <my-names-documentid>",<names-file-data-as-stream>)**
- **pdf-data-out=GET("PDF","-PDF -NAMES memory://<names-document-id> -TEMPLATE memory://<template-document-id>",<xml-map-data>)**

If you have to specify **-SCHEMA** (or **-S**), then you can do the same for the schema files.

Store output document in memory

Without the **-PDF** command, instead of streaming to a map, output operation can keep the output document in memory until it is fetched with **-GDOC** option. This is a two step process:

- **=PUT("PDF","-URL memory://<my-output-doc-id> -NAMES memory://<names-document-id> -TEMPLATE memory://<template-document-id>",<xml-map-data>)** or the adapter command line can be used with the output card.
- **pdf-data-out=GET("PDF","-GDOC <my-output-doc-id>")**

Default File URL

It is required to specify **memory://** for memory documents, **http://** for remote documents, or **file:///** for local file system documents using the **-URL (-U)**, **-NAMES (-N)**, **-SCHEMA (-S)** and **-TEMPLATE (-P)** commands.

Reference XML data

-URL memory://<xml-data-document-id> can be specified. XML data is in memory and can be referenced instead of passing map data as a third parameter of GET. The third parameter has precedence over the **-URL** option.

RabbitMQ Adapter

This documentation describes the RabbitMQ Adapter.

- [**Overview**](#)
The RabbitMQ adapter provides access to RabbitMQ message broker exchanges and queues.
- [**Introduction**](#)
The RabbitMQ adapter is used in maps and flows to send messages to the specified exchange or to receive messages from the specified queue.
- [**Prerequisites**](#)
- [**Authenticating connections**](#)
The RabbitMQ adapter supports username and password-based authentication to establish the connections to the RabbitMQ broker. The username and password credentials need to be specified in the User and Password properties in the connection definition for the adapter. The provided credentials are mandatory to authenticate AMQP connections established on the Port property and HTTP connections established on the Management Port property.
- [**Adapter properties and commands**](#)
This section lists the properties and commands supported by the adapter.
- [**Design Time Operations**](#)
- [**Examples**](#)

Overview

The RabbitMQ adapter provides access to RabbitMQ message broker exchanges and queues.

Introduction

The RabbitMQ adapter is used in maps and flows to send messages to the specified exchange or to receive messages from the specified queue.

The following features are supported by the RabbitMQ adapter:

- Testing of configured connections.
- Listing of available queues and exchanges.
- Generating schemas.
- Receiving messages from the specified queue with the support for multiple receiving guarantee modes.
- Sending messages to the specified exchange using the specified routing key with the support for multiple delivery guarantee modes.
- Accessing the message payload and metadata in individual messages including native message headers.
- Using TLS for secure connections.

Prerequisites

It is recommended that the RabbitMQ broker is up to date with the latest available patches. For more information about officially supported RabbitMQ releases and the latest available patches, see <https://www.rabbitmq.com/versions.html>.

The RabbitMQ client library utilized by the adapter to connect to the RabbitMQ broker is included with the adapter library and does not need to be installed separately in the adapter's runtime environment.

Configure the network connectivity from the adapter's runtime environment to the broker. The value of the Host property in the connection definition for the adapter must be either of the following options:

- DNS-resolvable host name.
- IP address of the broker.

For the adapter to establish an AMQP connection and a HTTP connection with the broker, the port numbers in Port and Management Port properties in the connection definition for the adapter must be active and accessible on the broker from the adapter's runtime environment.

Authenticating connections

The RabbitMQ adapter supports username and password-based authentication to establish the connections to the RabbitMQ broker. The username and password credentials need to be specified in the User and Password properties in the connection definition for the adapter. The provided credentials are mandatory to authenticate AMQP connections established on the Port property and HTTP connections established on the Management Port property.

In addition to the authentication of the user based on the username and password credentials, the adapter supports certificate-based authentication with the RabbitMQ broker by utilizing TLS-secured AMQP and HTTP connections.

- The adapter can be configured to request the RabbitMQ broker to provide a certificate that the adapter validates using the truststore information specified in the Truststore Location and Truststore Password properties.
- The Verify Hostname property can be set to instruct the adapter to verify, if the host name specified in the connection is also listed in the certificate presented by the broker. The adapter can also be configured to present its certificate using the keystore information specified in the Keystore Location, Keystore Password, and Key Password properties when the broker is configured to request clients to provide their certificate when connecting to it. The Key Password is applicable when the private key in the keystore is protected by a password different from the keystore password.
- [RabbitMQ adapter as a source](#)
- [RabbitMQ adapter as a target](#)

RabbitMQ adapter as a source

The RabbitMQ adapter as a source:

- Can be used as a data source for input card in maps, and a source node in flows. In this capacity, the adapter connects to the broker and receive messages from the queue specified in the Queue Name property.
- Can be configured to act as a watch for the source nodes in flows. In this capacity, the adapter is utilized from the listener service and registers itself with the RabbitMQ broker to notify the messages arriving on the queue specified in the Queue Name property. The adapter captures the messages and reports them as events which in turn are used to trigger instances of the flow in which the watch is defined. When an instance of the flow runs, the input node in the flow on which the watch is defined operates on the data collected from the event that triggered the flow instance without a need to connect to the broker again.
- Receives messages from the specified queue with support of multiple Receiving Guarantee Mode. In the simplest cases, the adapter can be configured to read messages with automatic acknowledgment which removes messages from the queue immediately and with automatic negative acknowledgment which always restores messages on the queue. The adapter can also be configured to acknowledge messages positively or negatively to the broker and to receive messages under AMQP transactions and commit or roll back the transactions based on the reported execution status in the map or flow.
- Provides support for accessing payloads and message properties in the received messages including custom message headers which all can be used in the transformation logic defined for the map or flow. When providing message properties along with the payload data, the adapter reports them together in form of JSON documents. The adapter supports encoding binary message payloads as base64 or hex-digit pairs encoded values in JSON documents that cannot be represented as text.

RabbitMQ adapter as a target

The RabbitMQ adapter as a target:

- Can be used as data target for output card in maps, and a target node in flows. In this capacity, the adapter connects to the broker and sends messages to the exchange specified in the Exchange Name property using the routing key specified in the Routing Key property.
- Sends messages to the specified exchange using the specified routing key with the support of multiple Delivery Guarantee Mode. In the simplest cases, the adapter can be configured to send messages in "fire-and-forget" mode in which case it does not receive or expect to receive any confirmation from the broker. The adapter

can also be configured to request confirmation on successful delivery from the broker and act on the confirmation status and to send messages under AMQP transactions and commit or roll back the transactions based on the reported execution status in the map or flow.

- Provides support for setting payloads and message properties for the produced messages which can be provided manually or derived by the transformation logic defined for the map or flow. When providing message properties along with the payload data, the adapter reports them together in form of JSON documents. The adapter supports encoding binary message payloads as base64 or hex-digit pairs encoded values in JSON documents that cannot be represented as text.

Adapter properties and commands

This section lists the properties and commands supported by the adapter.

Host

Specifies the host name where the RabbitMQ broker is installed and running. This is a valid DNS name of the host or its IP address. This is a mandatory property. The corresponding adapter command is `-H host` (or `-HOST host`).

Port

Specifies the port on which the RabbitMQ broker is listening for connections from clients such as the RabbitMQ adapter and communicating with them using the AMQP protocol. Messages are consumed and produced on the connection established on this port. The default value is 5672. If TLS Connections property is set to value AMQP or AMQP/HTTP, the adapter establishes secure AMQP connection to the broker. The port number must match the port on which RabbitMQ accepts secure AMQP connections. Default port is 5671. The corresponding adapter command is `-P port` (or `-PORT port`).

Management Port

Specifies the port on which the RabbitMQ broker is accepting API calls over HTTP for performing management operations. The adapter connects to this port on the broker to obtain the lists of available exchanges and queues to display in the web UI. The default value is 15672. If TLS Connections property is set to value HTTP or AMQP/HTTP, the adapter establishes secure HTTP (HTTPS) connection to the broker. The port number must match the port on which RabbitMQ accepts secure HTTP connections. Default port is 15671. The corresponding adapter command is `-MP port` (or `-MANAGEMENTPORT port`).

User

Specifies the username to use for authentication when connecting to the message broker. This is a mandatory property. The corresponding adapter command is `-USR username` (or `-USERNAME username`).

Password

Specifies the password value to use for authentication when connecting to the message broker. The corresponding adapter command is `-PWD password` (or `-PASSWORD password`).

Virtual Host

Specifies the virtual host for the connection defined in the broker. The user must be authorized to access that virtual host. The default value is `/`, which is the default virtual host defined in the broker. The corresponding adapter command is `-VH vhost` (or `-VIRTUALHOST vhost`).

Connection Name

Specifies the optional name to use for identifying the connection. The broker includes this value in the broker log messages which makes it easier to locate log messages that apply to the connection established by the adapter. The corresponding adapter command is `-CN name` (or `-CONNECTIONNAME name`).

TLS Connection

Specifies the types of connections to utilize TLS for security. The corresponding adapter command is `-TLSC connections` (or `-TLS CONNECTIONS connections`). The following values are supported:

- **None**: TLS is never used. This is the default value. The corresponding adapter command value is `none`.
- **AMQP**: TLS for AMQP connections only. AMQP connections are used for producing and consuming messages. The corresponding adapter command value is `amqp`.
- **HTTP**: TLS for HTTP connections only. HTTP connections are used for listing available queues and exchanges in the web UI. The corresponding adapter command value is `http`.
- **AMQP/HTTP**: TLS for both AMQP and HTTP connections. The corresponding adapter command value is `amqp_http`.

Truststore Location

Specifies the path to the truststore in PKCS#12 or JKS format that the adapter uses to validate the broker certificate when TLS is enabled. The truststore must contain the chain of certificates up to the CA certificate that is used to sign the broker certificate. The corresponding adapter command is `-TSL path` (or `-TRUSTSTORELOCATION path`).

Truststore Password

Specifies the password for the adapter to use to access the specified truststore when TLS is enabled. The corresponding adapter command is `-TSP password` (or `-TRUSTSTOREPASSWORD password`).

Verify Hostname

Verifies the hostname specified for the connection is listed in the certificate presented by the broker when TLS is enabled. If it is not listed in the certificate, the connection fails. The corresponding adapter command is `-VH` (or `-VERIFYHOSTNAME`).

Keystore Location

Specifies the path to the keystore in PKCS#12 or JKS format that the adapter uses to retrieve the private key and certificate with the public key. Which is required, if the broker is configured to request clients to provide a certificate for TLS connections (mutual TLS). The corresponding adapter command is `-KSL path` (or `-KEYSTORELOCATION path`).

Keystore Password

Specifies the password for the adapter to access the specified keystore when TLS is enabled and mutual TLS is configured. The corresponding adapter command is `-KSP password` (or `-KEYSTOREPASSWORD password`).

Key Password

Specifies the password for the adapter to access the private key in the specified keystore when TLS is enabled and mutual TLS is configured. The password is required, if the private key is secured with a password different from the keystore password. The corresponding adapter command is `-KP password` (or `-KEYPASSWORD password`).

Queue Name

Specifies the name of the queue from which to consume messages. Click the Fetch button when configuring this property in the web UI to show the available queue and select a queue from the list. You can enter in the name of the queue manually. The corresponding adapter command is `-Q name` (or `-QUEUE name`).

Exchange Name

Specifies the name of the exchange to produce the messages. Click the Fetch button when configuring this property in the web UI to show the available exchanges and select an exchange from the list. You can type in the name of the exchange manually. The corresponding adapter command is `-E name` (or `-EXCHANGE name`).

Routing Key

Specifies the routing key for the broker to use along with the exchange name to determine the queue or queues to which the produced messages should be routed and stored. The corresponding adapter command is `-RK key` (or `-ROUTINGKEY key`).

Receiving Guarantee Mode

Specifies the guarantee mode to request when consuming messages from the queue. The corresponding adapter command is `-RGM mode` (or `-RECEIVINGGUARANTEEMODE mode`). The following values are supported:

- **Transactions (AMQP TX):** The adapter consumes messages under an AMQP (TX) transaction. If the adapter is requested to commit the transaction in the map or flow, it positively acknowledges the messages consumed under the transaction and commits the AMQP transaction after which the broker removes the messages from the queue. If the adapter is requested to roll back the transaction in the map or flow, it rolls back the AMQP transaction which results in the messages remaining on the queue. This is the default option. The corresponding adapter command value is `tx`.
- **Acknowledgments Only:** The adapter consumes messages outside of an AMQP (TX) transaction. If the adapter is requested to commit the transaction defined in the map or flow context, it positively acknowledges the messages consumed under the transaction and the broker removes them from the queue. If the adapter is requested to roll back the transaction in the map or flow, it negatively acknowledges (nacks) the messages consumed under the transaction with redelivery enabled which results in the messages remaining on the queue. The corresponding adapter command value is `acknowledgments_only`.
- **Read Only:** When a message is consumed by an adapter with redelivery enabled, the adapter negatively acknowledges the message regardless of whether it is requested to commit or rollback the transaction, so the message always remains on the queue. The corresponding adapter command value is `read_only`.
- **None:** The adapter consumes messages in auto-acknowledge mode. This means that the messages are permanently removed from the queue after consumed by the adapter regardless of whether the adapter is subsequently requested to commit or rollback the transaction in the map or flow context. The corresponding adapter command value is `none`.

Delivery Guarantee Mode

Specifies the guarantee mode to request when producing messages to the exchange. The corresponding adapter command is `-DGM mode` (or `-DELIVERYGUARANTEEMODE mode`). The following values are supported:

- **Transactions (AMQP TX):** The adapter produces messages to the exchange under an AMQP (TX) transaction. If the adapter is requested to commit the transaction in the map or flow context, it commits the AMQP transaction after which the broker stores the messages. If the adapter is requested to roll back the transaction in the map or flow context, it rolls back the AMQP transaction which results in the messages not being stored on the queue(s). Note that for the true transactional behavior to be enforced all the messages need to be routed to the same queue, it means the exchange and the routing key used for producing the messages must result in all messages in the transaction being routed and stored on the same queue. This is the default option. The corresponding adapter command value is `tx`.
- **Publisher Confirms:** The adapter requests the broker to send confirmations about successful acceptance of each message. If the adapter is subsequently requested to commit the transaction in the map or flow context, it checks if all the messages in the transaction were positively acknowledged (acked) by the broker (and waits for any pending confirmations if necessary) and performs no further action. If any messages were negatively acknowledged (nacked) by the broker, the adapter logs them, if the logging is enabled and throws an exception to indicate the problem. The messages that were already positively acknowledged by the broker are stored on the broker and those that were negatively acknowledged are not. The corresponding adapter command value is `publisher_confirms`.
- **None:** This is a 'fire and forget' mode. The adapter publishes messages and does not perform any further checking to verify if the messages were successfully accepted and stored by the broker. The corresponding adapter command value is `none`.

Mandatory Routing

Specifies the mandatory routing for the messages for the specified exchange name and routing key. When this property is enabled and a message not routed to its destination, the broker returns the messages to the adapter asynchronously and the adapter is notified about this throws an exception and causes the map or flow to fail.

When this property is enabled and Delivery Guarantee Mode is set to Transactions (AMQP TX), the adapter is notified about non-routable messages during the execution of the AMQP commit call. If that happens, the commit will complete successfully and the messages that were successfully routed will be committed but the adapter will still throw exception to indicate that some messages could not be routed and were returned by the broker.

When this property is enabled and Delivery Guarantee Mode is set to Publisher Confirms, the adapter is be notified about non-routable messages prior to being notified with positive or negative confirmation for each message. Since the adapter always waits for the confirmations from the broker in this mode, it will be timely notified of any non-routable and returned messages and will throw an exception. By default, this property is enabled.

If an alternate exchange is defined for the exchange to which the messages are published and the message could not be routed to the original exchange, it will be re-routed by the broker to the alternate exchange and will not be considered as non-routable. If this property is enabled, the broker will not return the message to the adapter as the message will be considered successfully routed. Returned messages are always logged by the adapter in the adapter log when logging is enabled. The corresponding adapter command is `-MR` (or `-MANDATORYROUTING`).

Consumer Tag

Specifies the consumer tag for the consumer. Tag to assign to the consumer created by the adapter for consuming messages from the broker to serve as events for triggering flow instances. Therefore, this property only applies to the consumer the adapter creates when used in a source node in a flow for that a watch is enabled. When the value is not specified for this property, the broker automatically generates a consumer tag value for the consumer. The corresponding adapter command is `-CTAG tag` (or `-CONSUMERTAG tag`).

Message Header

Specifies the JSON schema version that represents messages. The default value `None` means that message metadata will not be processed, only the message payload. The adapter does not do any JSON processing in this case and the entire data passed to and from the adapter represents the message payload binary data. The corresponding adapter command value is `0`. The value `Version 1` indicates the message JSON structure version 1, which is covered in detail in the Generating Schema section. The corresponding adapter command value is `1`. The corresponding adapter command is `-HDR version` (or `-HEADER version`).

Charset

Specifies the character set to use for conversion between the raw bytes exchanged with the adapter framework and the Java strings required for the message header JSON fields. By default, the charset encoding from the current system locale is used. The corresponding adapter command is `-CS charset` (or `-CHARSET charset`).

Message Header Element

Specifies the message header fields. Comma-separated list of message header fields to include in the generated JSON schema and in the generated JSON representation of messages when consuming messages. Any extra whitespaces around the values are not ignored and are considered parts of those values. The property applies only when the Message Header property is set to a value other than `None`. The corresponding adapter command is `-MHE list` (or `-MESSAGEHEADERELEMENTS list`).

For example, if this property is set to the value `app_id,priority,message_id,user_id`, the JSON representation of the message will be:

```
{
    "app_id": "",
    "priority": 0,
    "message_id": "",
    "user_id": ""
}
```

It is important to note that the order of fields in the property value matches the order of elements in the generated JSON. Also, the payload element must be set among the property's fields to be included in JSON.

Native Headers

Specifies the native message headers. Comma-separated list of native message headers to include in the headers element in the generated JSON schema and in the generated JSON representation of messages when consuming messages. The native headers are required to be captured. They are required to be captured when the Message Header Elements property is not set, in that case all message header elements are automatically included and when it is set to a comma-separated list of values which includes the headers values. It is important to note that any extra whitespaces around the values in the list are not ignored and are considered parts of those values. The corresponding adapter command is -NH *list* (or -NATIVEHEADERS *list*).

For example, if generating a schema and the property Message Header Elements is set to value message_id,headers,payload, and the Native Header Fields is set to value title,size, as part of the schema generation process the adapter retrieves message from the queue that contains title header of string type and size header of numeric integer type then the generated JSON schema will be:

```
{
    "message_id": "",
    "headers": [
        "title": "",
        "size": 0
    ],
    "payload": ""
}
```

Native Headers Element Type

Specifies the type to use when generating schema and consuming messages. You can define the type of the headers element in the JSON messages representation. When producing messages, the adapter automatically detects the type used in the JSON data provided to it. The corresponding adapter command is -NHET *type* (or -NATIVEHEADERSELEMENTTYPE *type*). The supported values are:

- Object:** The headers element is a JSON object. This is the default value. The corresponding adapter command value is object.
- Array:** The headers element is a JSON array. The corresponding adapter command value is array.

When the headers element is of array type, it is a JSON array with an arbitrary number of JSON objects that contains the key and value elements. In most cases, string types are appropriate when all headers for the message can be represented by the same atomic type:

```
"headers": [
    {
        "key": "header1_name",
        "value": "header1_value"
    }
]
```

When the headers element is of object type, it is a JSON object with the number of elements that matches the number of headers in the message and where each of those elements has name and value that match the corresponding header's key and value. It is appropriate to use when the headers for the message have different types, such as a string, integer, floating-point decimal number, Boolean, and list of string values:

```
"headers": {
    "header1_name": "header1_value",
    "header2_name": 10,
    "header3_name": 10.05,
    "header4_name": true,
    "header5_name": [
        "header5_element1_value"
    ]
}
```

Message Payload Encoding

Specifies the encoding of the value of the payload element in the JSON message representation. The corresponding adapter command is -MPE *encoding* (or -MESSAGEPAYLOADENCODING *encoding*). The supported values are:

- Text:** The value is a string representation of the message payload. It assumes that the message payload is text encoded using the current system locale encoding or the encoding specified in the Charset property. The corresponding adapter command value is text.
- JSON:** The value is in JSON format. This can be an atomic JSON value, a JSON object, or a JSON array. It assumes that the message payload is JSON text encoded using the current system locale encoding or the encoding specified in the Charset property. The corresponding adapter command value is json.
- Hex Pairs:** The value is a string consisting of consecutive hex-digit pairs, where each pair corresponds to a single byte in the message payload's binary content. The corresponding adapter command value is hex_pairs.
- Base64:** The value is a string that represents base64 encoded message payload's binary content. The corresponding adapter command value is base64.

Application Id

Specifies the application identifier message property value to set for the produced messages. The value can also be set on a per-message basis using the app_id element in the JSON message structure. When both values are specified, the app_id JSON element value takes preference. The corresponding adapter command is -AID *id* (or -APPLICATIONID *id*).

Content Type

Specifies the content type message property value to set for the produced messages, for example gzip. The value can also be set on a per-message basis using the content_type element in the JSON message structure. When both values are specified, the content_type JSON element value takes preference. The corresponding adapter command is -CTYPE *type* (or -CONTENTTYPE *type*).

Content Encoding

Specifies the content encoding message property value to set for the produced messages, for example text/plain or application/json. The value can also be set on a per-message basis using the content_encoding element in the JSON message structure. When both values are specified, the content_encoding JSON element value takes preference. The corresponding adapter command is -CENC *encoding* (or -CONTENTENCODING *encoding*).

Correlation Id

Specifies the correlation identifier message property value to set for the produced messages. The value can also be set on a per-message basis using the correlation_id element in the JSON message structure. When both values are specified, the correlation_id JSON element value takes preference. The corresponding adapter command is -CID *id* (or -CORRELATIONID *id*).

Delivery Mode

Specifies the mode of delivery to set for the produced messages. The value can also be set on a per-message basis using the delivery_mode element in the JSON message structure. When both values are specified, the delivery_mode JSON element value takes preference. The corresponding adapter command is -DM *mode* (or -DELIVERYMODE *mode*). The following values are supported for this property:

- **Transient:** For instructing the broker to store messages in memory. This is the default value. The corresponding adapter command value is transient.
- **Persistent:** For instructing the broker to persist messages on disk. The corresponding adapter command value is permanent.

Reply To

Specifies the name of the response queue to set for the produced messages. The value can also be set on a per-message basis using the reply_to element in the JSON message structure. When both values are specified, the reply_to JSON element value takes preference. The corresponding adapter command is -RTO *name* (or -REPLYTO *name*).

Expiration

Specifies the message expiration duration in milliseconds to set for the produced messages. The value can also be set on a per-message basis using the expiration element in the JSON message structure. When both values are specified, the expiration JSON element value takes preference. The corresponding adapter command is -EXP *duration* (or -EXPIRATION *duration*).

Message Id

Specifies the message identifier message property value to set for the produced messages. The value can also be set on a per-message basis using the message_id element in the JSON message structure. When both values are specified, the message_id JSON element value takes preference. The corresponding adapter command is -MID *id* (or -MESSAGEID *id*).

Priority

Specifies the priority to set for the produced messages. The value can also be set on a per-message basis using the priority element in the JSON message structure. When both values are specified, the priority JSON element value takes preference. The corresponding adapter command is -PRI *priority* (or -PRIORITY *priority*).

Timestamp

Specifies the timestamp message property value to set for the produced messages. It is a string value in the yyyy-MM-dd HH:mm:ss format representing a timestamp with a second precision and using the timezone from the current system locale. The value can also be set on a per-message basis using the timestamp element in the JSON message structure. When both values are specified, the timestamp JSON element value takes preference. The corresponding adapter command is -TS *timestamp* (or -TIMESTAMP *timestamp*).

Type

Specifies the user defined message type to set for the produced messages. The value can also be set on a per-message basis using the type element in the JSON message structure. When both values are specified, the type JSON element value takes preference. The corresponding adapter command is -TYP *type* (or -TYPE *type*).

User Id

Specifies the user identifier message property value to set for the produced messages. When set, RabbitMQ verifies that it matches the user id associated with the current connection. The value can also be set on a per-message basis using the user_id element in the JSON message structure. When both values are specified, the user_id JSON element value takes preference. The corresponding adapter command is -UID *id* (or -USERID *id*).

Limit

Specifies the total number of messages to consume. The default value is 1. The special value S indicates all available messages. When the adapter is utilized as a listener for a flow node with a watch this property does not affects. In that case, the adapter waits indefinitely for a new message to arrive for as long as it is in the running state. The corresponding adapter command is -QTY *limit*.

Timeout

Specifies the time duration (in seconds) to wait for before retrieving the message. The default value is S. This is a special value that indicates unlimited (infinite) wait. The value 0 means no wait. When the adapter is utilized as a listener for a flow node with a watch this property does not affects. In that case, the adapter waits indefinitely for a new message to arrive for as long as it is in the running state. The corresponding adapter command is -LSN *timeout*.

Logging

Specifies the level of logging to use for the log (trace) file produced by the adapter.

The value Information means log informational messages, the value Errors Only means log error messages only, and the value Verbose means log debug and trace level messages along with the informational and error messages.

The corresponding adapter command is:

-T[V|E][+] [*log_file*]

-T -> Log adapter informational messages.

-TE -> Log only adapter errors.

-TV -> Use verbose (debug) logging. The log file records all activity that occurs while the adapter is producing or consuming messages.

+ -> Appends the trace information to the existing log file. Omit this argument to create a new log file.

log_file -> Specifies the location of the log file to which to write log messages. If not specified, the default log file name m4rabbitmq.mtr is used and the file is stored to the directory in which the executed compiled map resides.

Append Log

Flag indicating what to do if the specified log file already exists. When set to true, the log messages are appended to the file. When set to false, the file is truncated, and the messages are written to the empty file. The default value is true.

Log File Path

Specifies the location of the log file to which to write log messages. If not specified, the default log file name m4rabbitmq.mtr is used and the file is stored to the directory in which the executed compiled map resides.

Design Time Operations

The following functions are performed by the adapter in the web UI to assist with the design process when designing maps or flows that utilize the RabbitMQ adapter:

- Testing connection.
- Listing the available queues and exchanges.
- Generating schema.

- **Testing Connections**

When defining a RabbitMQ connection in the web UI, the Test option is provided to verify the connections settings by attempting to connect to the RabbitMQ broker specified in the connection definition.

- **Listing queues and exchanges**

When defining properties for a RabbitMQ action, card, or node in the UI, the values for the Queue Name and Exchange Name properties can be entered manually or the Fetch option can be selected to retrieve the list of available queues and exchanges in the broker and then a value selected from the list. To retrieve the list, the adapter establishes HTTP connection to the broker using the settings from the connection definition associated with that action, card, or node.

- **Generating Schema**

When defining schema for a RabbitMQ action, card or node in the UI, an existing schema in the project can be selected or a new schema can be generated and automatically selected. When generating the schema, if the Message Header property is set to value None, adapter generates a simple classic schema with a single item to represent the message payload as raw binary data.

Testing Connections

When defining a RabbitMQ connection in the web UI, the Test option is provided to verify the connections settings by attempting to connect to the RabbitMQ broker specified in the connection definition.

For implementing the Test function, the adapter tries to establish AMQP connection to the RabbitMQ host at the port specified in the Port property. The user credentials and the TLS settings if enabled for the AMQP connection are utilized when establishing the connection and are therefore verified by the test operation.

The Management Port property is not used when performing the test operation from the UI, so the ability to establish HTTP connection to the broker is not verified by the test operation. The testing of HTTP connection is performed indirectly from the UI by listing the available queues and exchanges in the UI page on which the action, card, or node properties for the adapter are defined. Since those operations are performed using management API calls to the broker that require HTTP connection to the broker performing them will result in testing the Management Port property along with the credentials and TLS settings, if those have been enabled for the HTTP connections.

Listing queues and exchanges

When defining properties for a RabbitMQ action, card, or node in the UI, the values for the Queue Name and Exchange Name properties can be entered manually or the Fetch option can be selected to retrieve the list of available queues and exchanges in the broker and then a value selected from the list. To retrieve the list, the adapter establishes HTTP connection to the broker using the settings from the connection definition associated with that action, card, or node.

Generating Schema

When defining schema for a RabbitMQ action, card or node in the UI, an existing schema in the project can be selected or a new schema can be generated and automatically selected. When generating the schema, if the Message Header property is set to value None, adapter generates a simple classic schema with a single item to represent the message payload as raw binary data.

If the Message Header property is set to Version 1, the adapter generates JSON schema to represent the message metadata and payload.

The following example shows default Version 1 message JSON structure:

```
{  
    "version": 0,  
    "delivery_tag": 0,  
    "redelivered": true,  
    "exchange": "",  
    "routing_key": "",  
    "consumer_tag": "",  
    "app_id": "",  
    "content_type": "",  
    "content_encoding": "",  
    "correlation_id": "",  
    "delivery_mode": 0,  
    "reply_to": "",  
    "expiration": "",  
    "headers": {},  
    "message_id": "",  
    "priority": 0,  
    "timestamp": "",  
    "type": "",  
    "user_id": "",  
    "payload": ""  
}
```

By default, the adapter includes all the fields in the generated JSON schema. A reduced set of fields to include in the schema can be specified using the Message Header Elements property. When the property is set the adapter includes only the elements specified in the comma-separated list of values set in the property. Refer to the

Message Header Elements property description for more information.

The JSON type of the headers element can be array or object. When generating schema and consuming messages the selection for the type to use in the JSON generated by the adapter is made using the Native Headers Element Type property. For more information, see the Native Headers Element Type property description.

The JSON type of the payload element is string by default and captures the message payload in text format. It can also be any other JSON type if the message payload is JSON and the adapter is configured to capture the JSON structure. When generating schema and consuming messages the selection for the JSON type to use in the JSON generated by the adapter is made using the Message Payload Format property. For more information, see the Message Payload Format property description.

When generating schema, the adapter in certain cases needs to establish AMQP connection and retrieve a message from the specified queue, so that it can inspect the structure of the message and generate correct schema. In those cases, the adapter expects at least one message to be available on the queue. It proceeds to consume that message and then negatively acknowledges it so that it is restored on the queue. Note that this results in the message technically being redelivered and will result in the broker setting the redelivered property on the message to true.

The following conditions are required for connecting to the broker and retrieving a message from the queue when generating a schema:

- The Message Header is set to a value other than None. If set to None, the adapter generates a simple fixed schema with a single item to represent the message payload.
- Schema generation is initiated while configuring a source action, card, or node. The Queue Name property must be set in that case. When configuring a target action, card or node, the default simple schema or JSON schema with all elements is generated depending on the Message Header value.
- One or both of the following applies:
 - The JSON message structure is requested to have the headers element.
 - The Message Payload Format property is set to the value JSON. It is important to note that the headers element is implicitly requested when the Message Header Elements property is not set, in that case it is automatically included. It is explicitly requested when the Message Header Element property is set to a list of values including headers.

The following table provides a brief description about individual JSON elements in the JSON message structure.

Element	JSON type	Description
version	number	Version. Set the value 1 for the header version 1 .
delivery_tag	number	Delivery tag, applicable when consuming messages.
redelivered	boolean	Redelivered flag, applicable when consuming messages.
exchange	string	Exchange name, applicable when consuming messages.
routing_key	string	Routing key, applicable when consuming messages.
consumer_tag	string	Consumer tag, applicable when consuming messages with a flow listener.
app_id	string	Application id. Custom value assigned by the user.
content_type	string	Message content type, such as gzip .
content_encoding	string	Message content encoding, such as text/plain .
correlation_id	string	Correlation id. Custom value assigned by the user.
delivery_mode	number	Delivery mode. Value 1 for transient, Delivery mode. Value 2 for persistent.
reply_to	string	Queue name for response messages.
expiration	string	Message expiration. To set string representation of a number representing expiration time in milliseconds.
headers	object or array	Native headers associated with the message. The type is controlled using the Native Headers Element Type property.
message_id	string	Message id. Custom value assigned by the user.
priority	integer	Message priority as assigned by the user.
timestamp	string	Message timestamp in yyyy-MM-dd HH:mm:ss format, in the current system locale's time zone.
type	string	Message type. Custom value assigned by the user.
user_id	string	User id. When set, checking is performed to ensure it matches the user id of the current connection.
payload	string or JSON	Message payload. The type is controlled by using the Message Payload Encoding property.

Examples

- [Example of the GET map function](#)
- [Example of the PUT map function](#)

Example of the GET map function

In the following example, assume that the adapter is used in a GET function:

- When the adapter is used to retrieve all messages from a RabbitMQ queue `queue1` on the RabbitMQ broker running on the same host on which the map runs. Messages are acknowledged as part of committing the source transaction. Logging is enabled, with verbose level and default log file `m4rabbitmq.mtr` created in the map directory:

```
GET("RABBITMQ", "-H localhost -P 5672 -Q queue1 -RGM tx -QTY S -LSN 0 -TV")
```

Example of the PUT map function

In the following example, assume that the adapter is used in a PUT function:

- When the adapter is used to write a message to an exchange with the specified routing key on the RabbitMQ broker running on the same host on which the map runs. In this case, the built-in `amq.direct` exchange is used with the routing key `queue1` configured in the broker to route the message to the queue with the same name. The

input_data represents the message produced for the exchange, and this is a reference to a type from another card in the map. The log file rabbitmq.log is created in the directory in which the map runs, it is created in append mode, and only messages of the error severity are logged:

```
PUT("RABBITMQ", "-H localhost -P 5672 -E amq.direct -RK queue1 -TE+ rabbitmq.log", input_data)
```

Redis Adapter

This documentation describes the Redis Adapter.

- [Overview](#)
The Redis adapter provides access to Redis servers.
- [Introduction](#)
The Redis adapter supports all the data structures supported by Redis, and many operations for the data structures.
- [Redis adapter as a source](#)
The adapter can be specified as a data source for input cards in maps, and for source nodes in flows. All Redis commands are non-blocking except for SUBSCRIBE, PSUBSCRIBE, BLPOP, and BRPOP. For these commands, a time out can be specified.
- [Redis adapter as a target](#)
- [Supported Redis adapter](#)
The following commands are supported by the adapter. Any command can be invoked in any context because data can either be provided in the card data, or via the Data property.
- [Adapter properties and commands](#)
This section provides the detailed description of the adapter properties for defining connections, actions, and cards.
- [Examples](#)

Overview

The Redis adapter provides access to Redis servers.

Introduction

The Redis adapter supports all the data structures supported by Redis, and many operations for the data structures.

It can be used to set values in Redis data structures, get values from Redis and to subscribe to channels when acting as a listener for flow execution.

The supported versions of the Redis server are 3.0.1 and later.

Redis adapter as a source

The adapter can be specified as a data source for input cards in maps, and for source nodes in flows. All Redis commands are non-blocking except for SUBSCRIBE, PSUBSCRIBE, BLPOP, and BRPOP. For these commands, a time out can be specified.

The adapter can also be configured to act as a watch for the source nodes in flows. This is supported for **SUBSCRIBE**, **PSUBSCRIBE**, **BLPOP**, and **BRPOP** commands. The adapter is utilized from the listener service and executes the selected command. Once data is available on the channel, or in the list (If **BLPOP** or **BRPOP** are called) the adapter captures the messages and reports them as events which in turn are used to trigger instances of the flow in which the watch is defined. When an instance of the flow runs, the input node in the flow on which the watch is defined operates on the data collected from the event that triggered that flow instance, without a need to connect to the broker again.

For commands that return structured data, such as hashes, sets, and sorted sets, the data can either be returned in delimited form, or as JSON. This is determined by the setting of the JSON property.

Some commands both require data and return data. There are several ways to call such commands:

Invoke the adapter from an input card or source node and pass the request data in the Data property.

- Invoke the adapter from a request node in a flow.
- Invoke the adapter from a GET map function, providing the request data as the third argument.

Redis adapter as a target

The adapter can be specified as a data target for output cards in maps and for target nodes in flows.

Supported Redis adapter

The following commands are supported by the adapter. Any command can be invoked in any context because data can either be provided in the card data, or via the Data property.

Primitive commands

- **GET**
- **SET [EX=expireTime | NX]**
- **APPEND**
- **DEL**
- **EXISTS**
- **INCR**
- **INCRBY**
- **DECR**
- **DECRBY**

List commands

- **LPOP**
- **RPOP**
- **BLPOP**
- **BRPOP**
- **LPUSH**
- **RPUSH**

Set commands

- **SADD**
- **SMEMBERS**

Sorted set commands

- **ZADD [XX|NX]**
- **ZCARD**
- **ZCOUNT**
- **ZRANGE [BYScore | REV | WITHSCORES]**
- **ZREM**
- **ZRANK [WITHSCORE]**
- **ZREVRANK [WITHSCORE]**
- **ZSCORE**

Hash commands

- **HSET**
- **HGET**
- **HMGET**
- **HGETALL**
- **HEXISTS**
- **HDEL**
- **HSETNX**
- **HKEYS**
- **HINCRBY**

Channel (Pub/Sub) commands

- **SUBSCRIBE channel(s)**
- **PSUBSCRIBE pattern(s)**

- **PUBLISH** channel data

Adapter properties and commands

This section provides the detailed description of the adapter properties for defining connections, actions, and cards.

Host

Specifies the host name or TCP/IP address of the Redis server. This is a mandatory property. The corresponding adapter command is **-H name** (or **-HOST name**).

Port

Specifies the Redis sever port. The default value is 6379. The corresponding adapter command is **-P number** (or **-PORT number**).

Password

Specifies the password for connection to the server if required. The corresponding adapter command is **-PWD password** (or **-PASSWORD password**).

Database ID

Specifies the ID of the database to connect to, if required by the Redis server. The corresponding adapter command is **-DID id** (or **-DATABASE id**).

Enable SSL

User can enable this property if the server requires an SSL/TLS connection. The corresponding adapter command is **-S** (or **-SSL**).

Certificate Authority Key File

Specifies the file containing the authority key certificate, if used. The corresponding adapter command is **-CAF filename** (or **-CAFILE filename**).

Private Key File

Specifies the location of the private key file to use when mutual TLS is enabled. The corresponding adapter command is **-KF filepath** (or **-KEYFILE filepath**).

Public Key Certificate File

Specifies the file containing the public key certificate when mutual TLS is enabled. The corresponding adapter command is **-KF filepath** (or **-CERTFILE filepath**).

Server Name Indicator

Specifies the DNS hostname for server name indication. The corresponding adapter command is **-SNI**.

Verify Hostname

Instructs the adapter to verify the hostname value in the server certificate presented by the Redis server during the SSL/TLS handshake. The corresponding adapter command is **-V** (or **-VERIFY**).

Command

Specifies the command to execute. The corresponding adapter command is **-C command** (or **-CMD command**).

Key

Specifies the Redis key. This is required for all commands except **SUBSCRIBE**, **PSUBSCRIBE**, and **PUBLISH**. The corresponding adapter command is **-K key** (or **-KEY key**).

Channel

Specifies the name of the channel for pub/sub operations. The corresponding adapter command is **-CH name** (or **-CHANNEL name**).

Data

Specifies the data. The corresponding adapter command is **-D data** (or **-DATA data**).

When providing data to the adapter, it can be passed to the input terminals of a target or request node, or to the output card of a map. Alternatively, it can be passed via the Data property. This is convenient if calling a command that requires request data, such as **HMGET** or sorted set commands. For more information, see the Data Formats section.

JSON

If this option is enabled, the data sent to, or received from Redis is sent as JSON data. The corresponding adapter command is **-J** (or **-JSON**).

Options

Specifies the options to modify the behavior of certain commands. Multiple options should be separated with a | character. The corresponding adapter command is **-O** (or **-OPTIONS**).

The supported options are as follows:

Command	Option	Description
SET	EX=expiration	Expiration time in seconds.
SET	NX	Set if the key does not already exist.
ZADD	XX	Only update elements that already exist.
ZADD	NX	Only add new elements.
ZRANGE	BYSCORE	Start and stop are scores rather than index.
ZRANGE	REV	Reverse the order of the results.
ZRANGE	WITHSCORES	Return the scores with the elements.
ZRANK	WITHSCORES	Return the score with the rank.
ZREVRANGE	WITHSCORES	Return the score with the rank.

Charset

Specifies the character set of the data. Default value is UTF-8.

The character set of any string data. This is used when converting from strings to bytes to send to Redis, or when creating a string from bytes in Redis. If not specified, the default system encoding is used. The corresponding adapter command is **-CS charset** (or **-CHARSET charset**).

Timeout

The blocking time in seconds for **BLPOP**, **BRPOP**, **SUBSCRIBE**, and **PSUBSCRIBE** operations. If set to 0, the operation will block until an object is available. The corresponding adapter command is **-TO timeout** (or **-TIMEOUT timeout**).

Logging

This property specifies the level of logging to use for the log (trace) file produced by the adapter. The default is Off. The value Information means log informational, the value Errors Only means log error messages only, and the value Verbose means log debug and trace level messages along with the informational and error messages.

The corresponding adapter command is:

-T [E|V] [+] [file_path]

-T -> Log adapter informational messages.

-TE -> Log only adapter errors.

-TV -> Use verbose (debug) logging. The log file records all activity that occurs while the adapter is producing or consuming messages.

+ -> Appends the trace information to the existing log file. Omit this argument to create a new log file.

file_path -> The m4redis.mtr log file in the map directory.

Data Formats

For primitive operations, the data is passed as strings with no formatting. Data passed to operations is sent in database-delimited format (| delimiter, \n terminator, ! release). If the JSON option is enabled, then data is instead passed and returned as JSON objects/arrays. This table lists what is required for inputs and outputs for each command. For input data, lists should be specified terminating each entry with \n terminator. If multiple values are required, delimit the values with | or \n characters.

Command	In	Out	JSON
GET	-	Bytes	-
SET	Bytes	-	-
APPEND	Bytes	-	-
DEL	-	-	-
EXISTS	-	true or false	-
INCR	-	Integer	-
INCRBY	Integer	Integer	-
DECR	-	Integer	-
DECRBY	Integer	Integer	-
LPOP	-	Bytes	-
RPOP	-	Bytes	-
BLPOP	-	Bytes	-
BRPOP	-	Bytes	-
LPUSH	Bytes	-	-
RPUSH	Bytes	-	-
SADD	Bytes	-	-
SMEMBERS	-	List of members	Out
HSET	List of field/values	-	In
HSETNX	List of field/values	-	In
HGET	Key	Value	-
HMGET	List of fields		In/Out
HMGETALL	-	List of field/values	Out
HEXISTS	Key	true or false	-
HDEL	Key	-	
HKEYS	-	List of Keys	Out
HINCRBY	Field/Integer	Integer	In
ZADD	List of member/scores	-	In
ZCARD	-	Integer	-
ZCOUNT	Min/Max	Integer	In
ZRANGE	Start/Stop	List of members	In/Out
ZRANGE with scores	Start/Stop	List of member/scores	In/Out
ZREM	List of members	-	In
ZRANK	Member	Integer	
ZRANK with score	Member	Rank/Score	Out
ZREVRANK	Member	Integer	-
ZREVRANK with score	Member	Rank/Score	Out
ZSCORE	Member	Score	-
PUBLISH	Bytes	-	-
SUBSCRIBE	-	Bytes	-
PSUBSCRIBE	-	Bytes	-

Schemas can be generated from the Map Designer when the Redis adapter is used in an input or output card. It generates the appropriate schema based on:

- Whether the adapter is used in an input or output card.
- The selected command.
- Whether the JSON option is enabled.

Examples

Examples of the GET map function

In the following example, assume that the adapter is used in a GET function.

This command will execute the **ZRANGE** command for key "sortedset:1", specifying the start as 2 and stop as 5 (via the **-DATA** command). The **-OPTIONS** commands instructs the adapter to return the scores together with the elements, and to reverse the order of the data.

```
GET("REDIS", "-H redis-host -P 6379 -CMD ZRANGE -KEY sortedset:1 -DATA 2|5 -OPTIONS withscores|rev")
```

Examples of the PUT map function

In the following example, assume that the adapter is used in a PUT function.

This command will set the key "mykey" to the value provided as the 3rd parameter of the PUT function.

```
PUT("REDIS", "-H redis-host -P 6379 -CMD SET -KEY mykey", "This is my data!")
```

ServiceNow Adapter

The ServiceNow adapter provides support for accessing ServiceNow REST API resources.

The ServiceNow adapter provides access to the following ServiceNow REST APIs:

- Table API
- Aggregate API

- [Introduction](#)

The ServiceNow adapter is used to perform data integration with ServiceNow by invoking ServiceNow APIs to perform read and write operations on the tables used to store data for those resources. For more information about ServiceNow REST API, refer to the ServiceNow documentation.

- [Adapter Properties and Commands](#)

This section lists the properties supported by the adapter.

- [Examples](#)

This chapter provides GET and PUT examples.

Introduction

The ServiceNow adapter is used to perform data integration with ServiceNow by invoking ServiceNow APIs to perform read and write operations on the tables used to store data for those resources. For more information about ServiceNow REST API, refer to the ServiceNow documentation.

- [Authentication Connections](#)

Authentication Connections

The adapter supports the following authentication methods:

- Basic authentication. The user provides username and password values and the adapter automatically provides them to ServiceNow to perform the authentication. ServiceNow adapter supports Basic authentication by default.
- OAuth 2.0 authentication. It requires to provide access token as captured outside of the application. However, when the token expires, the following actions for the adapter to take can be configured using the Access Token Expiry Action property:
 - Report Error
 - Refresh Token
 - Request New Token Using Username and Password
 - Request New Token Using Client Credentials.

ServiceNow adapter as a source

ServiceNow adapter as a source:

- Retrieves the record(s).
- Computes aggregates.

ServiceNow adapter as a target

ServiceNow adapter as a target:

- Creates the record(s).
- Update the record(s).
- Deletes the record(s).

Adapter Properties and Commands

This section lists the properties supported by the adapter.

- **[Instance Name](#)**
- **[Authentication](#)**
Specifies the authentication methods to authorize the user access to the ServiceNow REST APIs/endpoints.
- **[Username](#)**
Specifies the username for the connection.
- **[Password](#)**
Specifies the password for the connection.
- **[Access Token](#)**
Specifies an access token to access the ServiceNow REST APIs using OAuth 2.0 Authentication.
- **[Access Token Expiry Action](#)**
Specifies the action to be taken by the adapter, if the provided access token is reported expired or invalid.
- **[Consumer Key](#)**
- **[Consumer Secret](#)**
Specifies the consumer secret value to use when performing Request New Token expiry action.
- **[Refresh Token](#)**
Specifies the refresh token to perform Refresh Token expiry action.
- **[Security Token](#)**
Specifies the security token value to perform the Request New Token expiry action.
- **[Token Storage File](#)**
Specifies the location of the file in the local file system in which to store the access token retrieved after performing Refresh Token or Request New Token expiry action.
- **[Encryption Key](#)**
Specifies the key to use for encrypting the access token with AES-256 encryption method when saving the token in the token storage file.
- **[Version](#)**
Specifies the service version of ServiceNow Table API.
- **[Endpoint](#)**
Specifies the endpoint entry in the configuration. This is a mandatory property. The corresponding adapter command is -E *value* (or -ENDPOINT *value*).
- **[Table Name](#)**
Specifies the ServiceNow table name to be acted upon when action Get Record, Insert Record, Update Record or Delete Record is selected.
- **[System Id](#)**
- **[Query Parameter](#)**
Specifies the parameters those would be passed to endpoint while invoking the ServiceNow REST API calls. The corresponding adapter command is -P (or -PROPERTIES). Query parameters should be specified in as a whitespace separated list of *param=value* pairs.
- **[Logging](#)**
This property specifies the level of logging to use for the log (trace) file produced by the adapter.
- **[Log File Path](#)**

Instance Name

Specifies a ServiceNow instance allocated to the user. Users can access an instance via web browser using each instance's unique URL such as <https://instancename.service-now.com>. This is a mandatory property. The corresponding adapter command is -INAME *instance* (or -INSTANCENAME *instance*).

Authentication

Specifies the authentication methods to authorize the user access to the ServiceNow REST APIs/endpoints.

The corresponding adapter command is -A *value* or -AUTH *value*. The following two authentication methods are supported:

- Basic - Authentication with username and password. The corresponding adapter command *value* is basic.
- OAuth 2.0 - Authentication with access token obtained through OAuth 2.0 workflow. The corresponding adapter command *value* is oauth2.

Username

Specifies the username for the connection.

The corresponding adapter command is -UN *username* (or -USERNAME *username*).

Password

Specifies the password for the connection.

The corresponding adapter command is -PWD *password* (or -PASSWORD *password*).

Access Token

Specifies an access token to access the ServiceNow REST APIs using OAuth 2.0 Authentication.

Refer to the ServiceNow documentation for information on how to configure OAuth 2.0 authentication and generate initial access token.. It can be generated by REST API tools, such as Postman. This is a mandatory property. The corresponding adapter command is -AT *token* (or -ACCESSTOKEN *token*).

Access Token Expiry Action

Specifies the action to be taken by the adapter, if the provided access token is reported expired or invalid.

The corresponding adapter command is -ATEA *value* (or -ACCESSTOKENEXPIRYACTION *value*). The following actions are supported:

- Report Error. The corresponding adapter command value is report_error.
 - Refresh Token. The corresponding adapter command value is refresh_token.
 - Request New Token Using Username and Password. The corresponding adapter command value is password.
 - Request New Token Using Client Credentials. The corresponding adapter command value is client_credentials.
-

Consumer Key

Specifies the consumer key value to use when performing Refresh Token or Request New Token expiry action. The corresponding adapter command is -CK *key* (or -CONSUMERKEY *key*).

Consumer Secret

Specifies the consumer secret value to use when performing Request New Token expiry action.

It is enabled when the user sets up an application under the Connected Apps in ServiceNow. This is a mandatory property. The corresponding adapter command is -CS secret (or -CONSUMERSECRET *secret*).

Refresh Token

Specifies the refresh token to perform Refresh Token expiry action.

It needs to be captured outside the adapter application to enable the adapter to refresh the access token without any user interaction. The corresponding adapter command is -RT *token* (or -REFRESHTOKEN *token*).

Security Token

Specifies the security token value to perform the Request New Token expiry action.

If it is present, it will be appended to the password in case of OAuth 2.0 authentication. The corresponding adapter command is -ST *token* (or -SECURITYTOKEN *token*) command.

Token Storage File

Specifies the location of the file in the local file system in which to store the access token retrieved after performing Refresh Token or Request New Token expiry action.

This is a mandatory property. The corresponding adapter command is -TF *filepath* (or -TOKENFILE *filepath*).

Encryption Key

Specifies the key to use for encrypting the access token with AES-256 encryption method when saving the token in the token storage file.

This is a mandatory property. The corresponding adapter command is -EK *key* (or -ENCRYPTIONKEY *key*).

Version

Specifies the service version of ServiceNow Table API.

All available service versions can be fetched during configuration of the adapter in the web UI. The corresponding adapter command is -V *version* (or -VERSION *version*).

Endpoint

Specifies the endpoint entry in the configuration. This is a mandatory property. The corresponding adapter command is **-E value** (or **-ENDPOINT value**).

Actions available in ServiceNow adapter are as follows:

- Table API
 - Retrieve records from a table. The corresponding adapter command *value* is `getTableRecords`.
 - Retrieve a record. The corresponding adapter command *value* is `getRecord`.
 - Insert a record in specified table. The corresponding adapter command *value* is `insertRecord`.
 - Update a specified record in the table. The corresponding adapter command *value* is `updateRecord`.
 - Delete a specified record. The corresponding adapter command *value* is `deleteRecord`.
- Aggregate API
 - Compute Aggregate. The corresponding adapter command *value* is `computeAggregate`.

Table Name

Specifies the ServiceNow table name to be acted upon when action Get Record, Insert Record, Update Record or Delete Record is selected.

This is a mandatory property. The corresponding adapter command is **-TN name** (or **-TABLENAME name**).

System Id

Specifies the system id of the record to fetch, update or delete. The corresponding adapter command is **-SID id** (or **-SYSID id**).

Query Parameter

Specifies the parameters those would be passed to endpoint while invoking the ServiceNow REST API calls. The corresponding adapter command is **-P** (or **-PROPERTIES**). Query parameters should be specified in as a whitespace separated list of *param=value* pairs.

For valid endpoint parameters of Table API and Aggregate API, refer to the ServiceNow documentation.

Logging

This property specifies the level of logging to use for the log (trace) file produced by the adapter.

The default is Off. The value Information means log informational, the value Errors Only means log error messages only, and the value Verbose means log debug and trace level messages along with the informational and error messages.

The corresponding adapter command is **-T [E|V] [+]** [file_path]

-T -> Log adapter informational messages.

-TE -> Log only adapter errors.

-TV -> Use verbose (debug) logging. The log file records all activity that occurs while the adapter is producing or consuming messages.

+ -> Appends the trace information to the existing log file. Omit this argument to create a new log file.

file_path -> The full path to the adapter trace log. If you omit this keyword, the adapter creates the m4svcnnow.mtr log file in the map directory.

Log File Path

Specifies the location of the log file to which to write log messages. If not specified, the default log file name m4svcnnow.mtr is used, and the file is stored to the directory in which the executed compiled map resides.

Examples

This chapter provides GET and PUT examples.

- [GET examples](#)
- [PUT examples](#)

GET examples

GET function example for fetching records from a table API

```
GET("SVCNOW", "-E getTableRecords -A basic -UN username -PWD password -INAME https://dev1instance.service-now.com/ -TN problem -V v1")
```

GET function example for fetching records from an aggregate API

```
GET("SVCNOW", "-E computeAggregate -A basic -UN username -PWD password -INAME https:// dev1instance.service-now.com/ -TN problem -P sysparm_avg_fields=problem_state")
```

PUT examples

PUT function example

```
PUT("SVCNOW", "-E insertRecord -A basic -UN username -PWD password -INAME https://dev1instance.service-now.com/ -TN problem", inputdatarequest)
```

The *inputdatarequest* must be in JSON request format.

REST Adapter

REST adapter provides the ability to invoke REST operations with JSON requests constructed in the map, and to produce JSON responses from REST operations for further processing in the map, that way adding support for integration with REST services from the maps. It supports the following key functions:

- HTTP and HTTPS protocol schemes.
- GET, POST, PUT, DELETE, PATCH and HEAD methods.
- Basic and OAuth 2.0 authentication modes.
- Custom request headers.
- Multipart data.
- [Authenticating connections](#)
- [REST Adapter properties and commands](#)
This section provides details about the properties and corresponding adapter commands.
- [SSL/TLS support in REST adapter](#)
The SSL/TLS provides stricter security for HTTPS connections involving very selected parties.
- [Pagination support in REST adapter](#)
- [Examples](#)
- [OAuth 2.0 authentication examples](#)
In case the access token provided in this authentication scheme expires, access token expiry action can be taken using below options:
- [Proxy server support](#)
This documentation describes the Proxy server support.

Authenticating connections

The adapter supports the following authentication methods:

- Basic authentication. The user provides username and password values and the adapter automatically assembles them into appropriate authorization header.
- OAuth 2.0 authentication. The user provides access token obtained through OAuth 2.0. The username and password can still be used to automatically refresh the access token.

REST Adapter as a source:

REST adapter as a source performs the specified GET or HEAD operation and returns the response.

REST Adapter as a target:

REST adapter as a target submits requests to the specified POST, PUT, PATCH or DELETE operation, as applicable.

REST Adapter properties and commands

This section provides details about the properties and corresponding adapter commands.

- [Configuration mode](#)
- [URL](#)
- [Method](#)
- [Headers](#)
- [Script Path](#)

- **Endpoint**
Specifies the endpoint entry in the configuration. This property is enabled only in Configuration Script and Configuration Package modes and is currently reserved for internal use only.
- **Package Id**
Specifies the identifier string for the deployment package. This property is enabled only in Configuration Package mode and is currently reserved for internal use only.
- **Authentication**
Specifies the authentication method to use for the REST calls. The corresponding adapter command is -A (or -AUTH).
- **Access Token**
- **Access Token Expiry Action**
Specifies the action to be taken by Rest Adapter if the access token provided is either expired or invalid.
- **Access Token URL**
- **Consumer Key**
- **Consumer Secret**
- **Refresh Token**
- **Security Token**
- **Token Storage File**
Specifies the location of the file in the local file system in which to store the access token retrieved after performing Refresh Token or Request New Token expiry action.
- **Encryption Key**
Specifies the key to use for encrypting the access token with AES-256 encryption method when saving the token in the token storage file.
- **Logging**
- **Append Log**
- **Log File Path**
Name of the log file to which to write log messages. If not specified, the default log file name m4rest.mtr is used, and the file is stored to the directory in which the executed compiled map resides.

Configuration mode

Specifies the mode of work for the adapter. The default value is Generic, and when selected, the URL, Method and Headers properties are enabled and used to define which URL to connect to, which method to perform on the resource specified in the URL and which request headers (if any) to set on the request. The values Configuration Script and Configuration Package are used for implementing custom adapter extensions and are currently reserved for internal use only.

URL

Specifies the URL of the resource (endpoint). The URL can include query parameters. The corresponding adapter command is -U (or -URL).

Method

Specifies the method to perform on the resource. The corresponding adapter command is -M (or -METHOD). The supported values are:

- GET (Adapter command value: get)
- Head (Adapter command value: head)
- PUT (Adapter command value: put)
- GET (Adapter command value: get)
- POST (Adapter command value: post)
- PATCH (Adapter command value: patch)
- DELETE (Adapter command value: delete)

Headers

Specifies one or more HTTP request headers. This is an optional property. It can be specified by command with spaces separated key value pairs. For example:
header1=value1 header2=value2.

To specify REST Adapter Header values that contain a space character, see [technote](#).

Script Path

Provides the script path. This property is available, when you set Configuration Mode to Configuration Script and is currently reserved for internal use only. This is a mandatory property.

Endpoint

Specifies the endpoint entry in the configuration. This property is enabled only in Configuration Script and Configuration Package modes and is currently reserved for internal use only.

Actions available in REST adapter as follows:

- Get Record (Adapter command value: getRecord)
 - Insert Record (Only Target Mode) (Adapter command value: createRecord)
 - Update Record (Only Target Mode) (Adapter command value: updateRecord)
 - Delete Record (Only Target Mode) (Adapter command value: deleteRecord)
 - Execute Query (Adapter command value: query)
 - Execute QueryAll (Adapter command value: queryAll)
 - Query Performance (Adapter command value: queryPerformance)
 - Search String (Adapter command value: searchString)
-

Package Id

Specifies the identifier string for the deployment package. This property is enabled only in Configuration Package mode and is currently reserved for internal use only.

Authentication

Specifies the authentication method to use for the REST calls. The corresponding adapter command is -A (or -AUTH).

There are three methods as follows

- Authentication is not performed. The adapter command value is None.
 - Authentication with username and password. The adapter command value is Basic.
 - Authentication with access token obtained through OAuth 2.0 workflow. The adapter command value is OAuth 2.0.
-

Access Token

Specifies the access token string to use with OAuth2.0 authentication method. The corresponding adapter command is -AT (or -ACCESSTOKEN).

Access Token Expiry Action

Specifies the action to be taken by Rest Adapter if the access token provided is either expired or invalid.

The default value is Report Error. The following values are supported:

- Report Error (Adapter command value: reportError).
- Refresh Token (Adapter command value: refresh_token).
- Request New Token (Adapter command value: password).

This is a mandatory property. The corresponding adapter command is -ATEA or (-ACCESSTOKENEXPIRYACTION).

Access Token URL

Specifies the URL of the authentication server for providing access tokens. This property is enabled when Refresh Token and Request New Token expiry action is selected. The corresponding adapter command is -TURL (or -TOKENURL).

Consumer Key

Specifies the consumer key value to use when performing Refresh Token or Request New Token expiry action. The corresponding adapter command is -CK (or -CONSUMERKEY).

Consumer Secret

Specifies the consumer secret value to use when performing Request New Token expiry action. The corresponding adapter command is -CS (or -CONSUMERSECRET).

Refresh Token

Specifies the refresh token to provide when performing Refresh Token expiry action. The corresponding adapter command is -RT (or -REFRESHTOKEN).

Security Token

Specifies the security token value to perform when performing Request New Token expiry action. The corresponding adapter command is -ST (or -SECURITYTOKEN).

Token Storage File

Specifies the location of the file in the local file system in which to store the access token retrieved after performing Refresh Token or Request New Token expiry action.

The corresponding adapter command is -TF (or -TOKENFILE).

Encryption Key

Specifies the key to use for encrypting the access token with AES-256 encryption method when saving the token in the token storage file.

The corresponding adapter command is -EK (or -ENCRYPTIONKEY).

Logging

Specifies the level of logging to use for the log (trace) file produced by the adapter. The default is off. The value info means log informational and error messages, the value error means log error messages only, and the value verbose means log debug and trace level messages along with the informational and error messages.

Append Log

Flag indicating what to do if the specified log file already exists. The default value is true. When set to true, the log messages are appended to the file. When set to false, the file is truncated, and the messages are written to the empty file.

Log File Path

Name of the log file to which to write log messages. If not specified, the default log file name m4rest.mtr is used, and the file is stored to the directory in which the executed compiled map resides.

SSL/TLS support in REST adapter

The SSL/TLS provides stricter security for HTTPS connections involving very selected parties.

The REST adapter/node provides a way to refer an SSL certificate that will identify them.

The SSL handshake will not work if server and client certificates are not a match. By doing so, no credential is sent over the Internet, just encrypted data that will be decrypted correctly only if the certificates match.

- **Authenticate Server**

The Authenticate Server property is used to authenticate to the server.

- **Truststore file full path**

The Truststore file full path property indicates the full path for truststore file for SSL handshake.

- **Truststore Password**

The Truststore Password property specifies the password for truststore.

- **Authenticate Client**

The Authenticate Client property is used for client certificate validation.

- **Keystore file path**

The Keystore file path property is used to specify the keystore file full path.

- **Keystore Password**

The Keystore Password property specifies the password for keystore.

- **Key Password**

- **Verify host name**

Authenticate Server

The Authenticate Server property is used to authenticate to the server.

The Authenticate Server property is applicable only when the protocol specified is HTTPS. This notifies the adapter that client server certificate validation will take place during the SSL handshake.

The corresponding command for this property is -AS (or -AUTHSERVER).

Truststore file full path

The Truststore file full path property indicates the full path for truststore file for SSL handshake.

This property is applicable only when the protocol used is **HTTPS** and Authenticate Server property is specified.

The default path for Truststore file full path property is JAVA_HOME/lib/security/cacerts

The corresponding command for this property is -TSL [file_path] (or -TRUSTSTORELOCATION [file_path]).

Truststore Password

The Truststore Password property specifies the password for truststore.

This property is applicable only when the protocol used is **HTTPS** and Authenticate Server property is specified.

The default password for Truststore Password property is changeit.

The corresponding command for this property is -TSP [pwd] (or -TRUSTSTOREPASSWORD [pwd]).

Authenticate Client

The Authenticate Client property is used for client certificate validation.

The Authenticate Client property is applicable only when the protocol used is **HTTPS**. This property notifies the adapter that client certificate validation will take place during SSL handshake.

The corresponding command for this property is -AC (or -AUTHCLIENT).

Keystore file path

The Keystore file path property is used to specify the keystore file full path.

This property is applicable only when the protocol used is **HTTPS** and Authenticate Client property is specified.

The default path for Keystore file path property is JAVA_HOME/lib/security/cacerts

The corresponding command for this property is -KSL [file_path] (or -KEYSTORELOCATION [file_path]).

Keystore Password

The Keystore Password property specifies the password for keystore.

This property is applicable only when the protocol used is **HTTPS** and Authenticate Client property is specified.

The default password for Keystore Password property is changeit.

The corresponding command for this property is -KSP [pwd] (or -KEYSTOREPASSWORD [pwd]).

Key Password

The Key Password property specifies the password for key.

This property is applicable only when the protocol used is **HTTPS** and Authenticate Client property is specified and ensure that the Key Password is different from the Keystore Password.

The default password for Key Password property is changeit.

The corresponding command for this property is -KP [pwd] (or -KEYPASSWORD [pwd]).

Verify host name

The Verify host name property is used to instruct the adapter to verify the hostname value in the server certificate presented by the REST endpoint server during the SSL handshake.

The property is applicable only when the protocol used is **HTTPS** and Authenticate Client property is specified.

The corresponding command for this property is -VH (or -VERIFYHOSTNAME).

Pagination support in REST adapter

Pagination turns big archives of data into smaller, more digestible pieces.

Many REST APIs provide a way to retrieve data using multiple calls and a pagination mechanism to get additional pages of data with each call to the API.

Pagination support in REST Adapter in the context of a burst mode of operation is where each call returns data that is provided to the map or flow which then processes the data and returns to the source for more data to get the next burst.

- **Pagination Type**

The there are four patterns that are commonly used for implementing pagination in REST APIs and they can be specified using the Pagination Type property.

- **Records per Call**

The Records per Call property specifies the number of records to fetch per call.

- **Limit**

The Limit property specifies the total number of records to return across all calls.

- **First Offset**

The First Offset property specifies the offset value to provide in the first call.

- **Number of Records Location**

The Number of Records Location property specifies the location in response as where to find the total number of records.

- **Number of Records Path**

The Number of Records Path property specifies the header name or the path where the total number of records are found.

- **First Page**

The First Page property specifies the page value to provide in the first call.

- **Number of Pages Location**

The Number of Pages Location property specifies the response location where the total number of pages are found.

- **Number of Pages Path**

The Number of Pages Path property specifies the header name or the path where the total number of pages are found.

- **Next Page Link Location**

The Next Page Link Location property specifies the name of the header or the path in the body for the field containing an URL for the next page. The URL may be absolute or relative.

- **Next Page Link Path**

The Next Page Link Path property specifies the header name or path where the URL for the next page is found.

- **Base URL**

The Base URL property specifies the URL to which all relative links for next pagination call would be appended.

- **Next Page Token Location**

The Next Page Token Location property specifies the name of the header or the path in the body for the field containing the token for the next page.

- **Next Page Token Path**

The Next Page Token Path property specifies the header name or path where the token for the next page is found.

- **Pagination Token Location**

The Pagination Token Location property specifies the location where the pagination token for subsequent calls would be configured.

- **Pagination Token Name**

The Pagination Token Name property specifies the name of the pagination token to be passed in the next subsequent calls.

- **Header and Parameter Special Values**

Pagination Type

There are four patterns that are commonly used for implementing pagination in REST APIs and they can be specified using the Pagination Type property.

The four patterns are as follows:

- Offset (offset)

- Page (page)

- Next Page Link (link)

- Next Page Token (token)

The corresponding command for this property is -PT [none | offset | page | link | token | USEENDPOINTDEF] (or -PAGINATIONTYPE [none | offset | page | link | token | USEENDPOINTDEF]).

Note: The **USEENDPOINTDEF** type would only be specified when the configuration mode is not a generic mode.

Records per Call

The Records per Call property specifies the number of records to fetch per call.

The corresponding command for this property is -RPC [num] (or -RECORDSPERCALL [num]).

Limit

The Limit property specifies the total number of records to return across all calls.

This property is applicable only when the Pagination Type is set to Offset.

The corresponding command for this property is -L [num] (or -LIMIT [num]).

First Offset

The First Offset property specifies the offset value to provide in the first call.

This property is applicable only when the Pagination Type is set to Offset.

The corresponding command for this property is -FO [num] (or -FIRSTOFFSET [num]).

Number of Records Location

The Number of Records Location property specifies the location in response as where to find the total number of records.

This property is applicable only when the Pagination Type is set to Offset.

The corresponding command for this property is -NRL [headers|body] (or -NUMRECORDSLOCATION [headers|body]).

Number of Records Path

The Number of Records Path property specifies the header name or the path where the total number of records are found.

This property is applicable only when the Pagination Type is set to Offset.

The corresponding command for this property is -NRP [path|name] (or -NUMRECORDPATH [path|name]).

First Page

The First Page property specifies the page value to provide in the first call.

This property is applicable only when the Pagination Type is set to Page.

The corresponding command for this property is -FP [num] (or -FIRSTPAGE [num]).

Number of Pages Location

The Number of Pages Location property specifies the response location where the total number of pages are found.

This property is applicable only when the Pagination Type is set to Page.

The corresponding command for this property is -NPL [headers|body] (or -NUMOFPAGESLOCATION [headers|body]).

Number of Pages Path

The Number of Pages Path property specifies the header name or the path where the total number of pages are found.

This property is applicable only when the Pagination Type is set to Page.

The corresponding command for this property is -NPP [path|name] (or -NUMOFPAGESPATH [path|name]).

Next Page Link Location

The Next Page Link Location property specifies the name of the header or the path in the body for the field containing an URL for the next page. The URL may be absolute or relative.

This property is applicable only when the Pagination Type is set to Next Page Link.

The corresponding command for this property is -NLL [headers|body] (or -NEXTLINKLOCATION [headers|body]).

Next Page Link Path

The Next Page Link Path property specifies the header name or path where the URL for the next page is found.

This property is applicable only when the Pagination Type is set to Next Page Link.

The corresponding command for this property is -NLP [path|name] (or -NEXTLINKPATH [path|name]).

Base URL

The Base URL property specifies the URL to which all relative links for next pagination call would be appended.

This property is applicable only when the Pagination Type is set to Next Page Link.

The corresponding command for this property is -BURL [URL] (or -BASEURL [URL]).

Next Page Token Location

The Next Page Token Location property specifies the name of the header or the path in the body for the field containing the token for the next page.

This property is applicable only when the Pagination Type is set to Next Page Token.

The corresponding command for this property is -NTL [headers|body] (or -NEXTTOKENLOCATION [headers|body]).

Next Page Token Path

The Next Page Token Path property specifies the header name or path where the token for the next page is found.

This property is applicable only when the Pagination Type is set to Next Page Token.

The corresponding command for this property is -NTP [path|name] (or -NEXTTOKENPATH [path|name]).

Pagination Token Location

The Pagination Token Location property specifies the location where the pagination token for subsequent calls would be configured.

This property is applicable only when the Pagination Type is set to Next Page Token.

The corresponding command for this property is -PTL [headers|query] (or -PAGINATIONTOKENLOCATION [headers|query]).

Pagination Token Name

The Pagination Token Name property specifies the name of the pagination token to be passed in the next subsequent calls.

This property is applicable only when the Pagination Type is set to Next Page Token.

The corresponding command for this property is -PTN [token_name] (or -PAGINATIONTOKENNAME [token_name]).

Header and Parameter Special Values

In paginated API calls the pagination parameters are provided either as query parameters, or (less commonly) as header values. Since these values change with each call to the API, these can be specified in the query parameter or header tables.

This is provided by special properties that have the syntax {{...}}. These special values are as follows:

- {{record_count}}
Specifies the value of records per call property.

For example: The {{record_count}} value provides a mechanism by which the Records per Call property can be easily changed or passed as a flow variable, without changing the value in the parameter or header definition.

- {{offset}}
Specifies the next offset value. On each call this will be incremented by the value of the Records per Call property.
 - {{page}}
Specifies the next page number. On each call this will be incremented.
 - {{pagination_token}}
Specifies the pagination token from the prior call. On the first call, the query parameter or header will not be set. On subsequent calls, it will be set to the value of the pagination token from the previous call.
-

Examples

- [No Authentication Examples](#)
 - [Basic Authentication examples](#)
-

No Authentication Examples

GET Example:

```
GET("REST", "-U restURL -M get -A none -H \"header1=value1 header2=value2\"")
```

PUT Example:

```
PUT("REST ", "-U restURL -M put -A none -H \"header1=value1 header2=value2\"")
```

Basic Authentication examples

GET Example:

```
GET("REST", "-U restURL -M get -A BASIC -UN username -PWD password -H \"header1=value1 header2=value2\"")
```

PUT Example:

```
PUT("REST ", "-U restURL -M put -A BASIC -UN username -PWD password -H \"header1=value1 header2=value2\"")
```

OAuth 2.0 authentication examples

In case the access token provided in this authentication scheme expires, access token expiry action can be taken using below options:

In case the access token provided in this authentication scheme expires, access token expiry action can be taken using below options:

REPORT ERROR

GET Example:

```
GET("REST", " -U restURL -M get -A OAUTH2 -ATEA reportError -AT accToken -H \"header1=value1 header2=value2\"")
```

PUT Example:

```
PUT("REST ", " -U restURL -M put -A OAUTH2 -ATEA reportError -AT accToken -H \"header1=value1 header2=value2\"")
```

REFRESH TOKEN

GET Example:

```
GET("REST", " -U restURL -M get -A OAUTH2 -ATEA refresh_token -TURL tokenURL -CK consumerKey -AT accToken -RT refreshToken -TF filepath -EK eKey -H \"header1=value1 header2=value2\"")
```

PUT Example:

```
PUT("REST ", " -U restURL -M put -A OAUTH2 -ATEA refresh_token -TURL tokenURL -CK consumerKey -AT accToken -RT refreshToken -TF filepath -EK eKey -H \"header1=value1 header2=value2\"")
```

REQUEST NEW TOKEN

GET Example:

```
GET("REST", " -U restURL -M get -A OAUTH2 -ATEA password -UN username -PWD password -TURL tokenURL -CK consumerKey -CS consumerSecret -AT accToken -ST secToken -RT refreshToken -TF filepath -EK eKey -H \"header1=value1 header2=value2\"")
```

PUT Example:

```
PUT("REST ", " -U restURL -M put -A OAUTH2 -ATEA password -UN username -PWD password -TURL tokenURL -CK consumerKey -CS consumerSecret -AT accToken -ST secToken -RT refreshToken -TF filepath -EK eKey -H \"header1=value1 header2=value2\"")
```

Proxy server support

This documentation describes the Proxy server support.

- [Proxy server support in REST Adapter](#)

The Services and REST Adapter provide the option to configure a proxy server URL. A proxy server serves as an intermediary between the client and the server. It receives REST requests from IBM Sterling Transformation Extender and acts on behalf of IBM Sterling Transformation Extender when connecting to the end server. The Proxy server is utilized when access to the destination server is restricted, and connection must be made exclusively through a proxy server.

- [Troubleshooting of proxy server](#)

Proxy server support in REST Adapter

The Services and REST Adapter provide the option to configure a proxy server URL. A proxy server serves as an intermediary between the client and the server. It receives REST requests from IBM Sterling Transformation Extender and acts on behalf of IBM Sterling Transformation Extender when connecting to the end server. The Proxy server is utilized when access to the destination server is restricted, and connection must be made exclusively through a proxy server.

- [Proxy configuration using environment variables](#)

This documentation describes proxy configuration using environment variables.

- [Proxy configuration using flow variables](#)

This documentation describes proxy configuration using flow variables.

- [Proxy configuration using properties](#)

This documentation describes proxy configuration using properties.

- [Proxy configuration using the command line](#)

This documentation describes proxy configuration using the command line.

Proxy configuration using environment variables

This documentation describes proxy configuration using environment variables.

HTTP_PROXY

To establish connections to any HTTP URLs through the REST adapter, you can set the `HTTP_PROXY` environment variable on the machine where IBM Sterling Transformation Extender is installed. This configuration allows the proxy server URL provided to be utilized for establishing connections.

Example: `HTTP_PROXY=http://proxy-server-ip:port`

HTTPS_PROXY

To establish connections to HTTPS URLs through the REST adapter, you can configure the `HTTPS_PROXY` environment variable on the machine where IBM Sterling Transformation Extender is installed. By setting this variable, the provided proxy server URL will be utilized for making connections to any HTTPS URLs.

Example: `HTTPS_PROXY=http://proxy-server-ip:port`

NO_PROXY

To bypass the proxy server for specific URLs in the machine where IBM Sterling Transformation Extender is installed, you can set the `NO_PROXY` environment variable. This variable should be configured with a comma-separated list of URLs that do not need to be passed through the proxy server URL specified in `HTTP_PROXY` or `HTTPS_PROXY`.

Example: `NO_PROXY=localhost,some-domain.com`

Note: For adding these flow variables in the docker installation, one must edit `setenv.sh` file in hip-server and hip-rest containers.

Proxy configuration using flow variables

This documentation describes proxy configuration using flow variables.

proxy_url

When deploying and running flows, you have the option to configure the "`proxy_url`" flow variable for both HTTPS and HTTP URL connections using the REST Adapter.

Proxy configuration using properties

This documentation describes proxy configuration using properties.

Proxy URL

You can configure the Proxy URL property in Service Builder and REST Adapter.

Proxy configuration using the command line

This documentation describes proxy configuration using the command line.

-PURL

In the command line mode of the REST Adapter, you can utilize the "**-PURL**" command to configure the proxy server URL.

Example: -PURL<https://proxy-server-ip:3128>

Troubleshooting of proxy server

To verify the proxy configuration checks, follow these steps:

- Check if the Proxy server URL can be accessed from the machine where IBM Sterling Transformation Extender is installed. Use the Telnet command to access the proxy server.
For example: `telnet <proxy-server-ip> <proxy-server-port>`
- If you are unable to access your port, ensure that it is enabled if you are behind a firewall.
- If your proxy server IP is not accessible, make sure to include the DNS entry of the proxy server details in the hosts file where IBM Sterling Transformation Extender is installed.
- If the REST API requests are not being passed through the proxy server, ensure that there is an entry for the end resource URLs in your host's file on the proxy server.

Salesforce Adapter

Salesforce Adapter provides support for accessing Salesforce service over REST API interface.

Using Salesforce Adapter, you can:

- Retrieve a record of a Salesforce object, such as Account, User, or custom object.
- Perform a query or search.
- Update or delete records.

Introduction

Salesforce Adapter is used to connect to REST APIs provided by Salesforce for easy integration and development without writing REST APIs frameworks in the application. Salesforce Adapter is an easy way to create and update underlying data model and standard objects.

- [Authenticating connections](#)
- [Properties and commands](#)
This chapter provides a detailed description of the adapter properties.
- [Examples](#)

Authenticating connections

OAuth 2.0 authentication. The user provides access token obtained through OAuth 2.0 workflow. The username and password can still be used to automatically refresh the access token. Before making REST API calls through Salesforce adapter, you must authenticate the application user using OAuth 2.0. Refer to Salesforce product documentation.

Salesforce adapter as a source:

The Salesforce adapter as a source is used to:

- Get a record.
- Execute a Query
- Check the query performance.
- Search a String.

Salesforce adapter as a target:

The Salesforce adapter as a target is used to:

- Insert a record.
- Update a record.
- Delete a record.
- [Salesforce adapter as source](#)

- [Salesforce adapter as a target](#)
-

Salesforce adapter as source

Salesforce adapter as a target

Properties and commands

This chapter provides a detailed description of the adapter properties.

- **Instance URL**
Specifies a server instance of organization. User can view their instance URL when they open their Salesforce Org screen. This is a mandatory property. The corresponding adapter command is -IURL (or -INSTANCEURL).
 - **Access token**
Specifies an access token to access the Salesforce REST APIs using OAuth 2.0 Authentication. Refer to [Generate an Initial Access Token](#) to generate initial access token.
 - **Version**
 - **Endpoint**
Specifies the endpoint entry in the configuration. This is a mandatory property. The corresponding adapter command is -E (or -ENDPOINT).
 - **Access Token Expiry Action**
This property specifies the action to be taken by Salesforce Adapter if the access token provided is either expired or invalid.
 - **Consumer key**
Specifies the consumer key value to use when performing Refresh Token or Request New Token expiry action.
 - **Consumer secret**
Specifies the consumer secret value to use when performing Request New Token expiry action. It is enabled when the user sets up an application under the Connected Apps in Salesforce.
 - **Refresh token**
Specifies the refresh token to provide when performing Refresh Token expiry action.
 - **Security Token**
Specifies the security token value to perform when performing Request New Token expiry action.
 - **Token Storage File**
Specifies the location of the file in the local file system in which to store the access token retrieved after performing Refresh Token or Request New Token expiry action.
 - **Encryption key**
Specifies the key to use for encrypting the access token with AES-256 encryption method when saving the token in the token storage file.
 - **Object Name**
This property specifies the object name to be acted upon when action Get Record, Insert Record, Update Record or Delete Record is selected.
 - **Id**
Specifies the record ID that is used for fetching, updating or deleting any record.
 - **SOQL Query**
SOQL Query is enabled, when you select the Execute Query, Execute QueryAll, or Query Performance action.
 - **Search string**
Specifies the search parameter that is used for simplified searching in Salesforce data instead of a SOQL clause.
 - **Logging**
Specifies the level of logging to use for the log (trace) file produced by the adapter.
 - **Append Log**
The Append Log is a flag that indicates what to do if the specified log file already exists.
 - **Log File Path**
Name of the log file to which to write log messages.
-

Instance URL

Specifies a server instance of organization. User can view their instance URL when they open their Salesforce Org screen. This is a mandatory property. The corresponding adapter command is -IURL (or -INSTANCEURL).

Access token

Specifies an access token to access the Salesforce REST APIs using OAuth 2.0 Authentication. Refer to [Generate an Initial Access Token](#) to generate initial access token.

After setting up the application under Connected Apps in Salesforce, you need the access token. It can be generated REST API tools, such as Postman. This is a mandatory property. The corresponding adapter command is -AT or (-ACCESTOKEN)

Version

Specifies the service version of Salesforce REST API which you want to connect to your Salesforce Org. All available service versions can be fetched in the UI listing. This is a mandatory property. The corresponding adapter command is -SV (or -SERVICEVERSION).

Endpoint

Specifies the endpoint entry in the configuration. This is a mandatory property. The corresponding adapter command is -E (or -ENDPOINT).

Actions available in Salesforce adapter as follows:

- Get Record (Adapter command value: getRecord).
 - Insert Record (Only Target Mode) (Adapter command value: createRecord).
 - Update Record (Only Target Mode) (Adapter command value: updateRecord).
 - Delete Record (Only Target Mode) (Adapter command value: deleteRecord).
 - Execute Query (Adapter command value: query).
 - Execute QueryAll (Adapter command value: queryAll).
 - Query Performance (Adapter command value: queryPerformance).
 - Search String (Adapter command value: searchString).
-

Access Token Expiry Action

This property specifies the action to be taken by Salesforce Adapter if the access token provided is either expired or invalid.

The default value is Report Error. The following values are supported:

- Report Error (Adapter command value: reportError).
- Refresh Token (Adapter command value: refresh_token).
- Request New Token (Adapter command value: password).

This is a mandatory property. The corresponding adapter command is -ATEA or (-ACCESSTOKENEXPIRYACTION).

Consumer key

Specifies the consumer key value to use when performing Refresh Token or Request New Token expiry action.

This is a mandatory property. The corresponding adapter command is -CK (or -CONSUMERKEY).

Consumer secret

Specifies the consumer secret value to use when performing Request New Token expiry action. It is enabled when the user sets up an application under the Connected Apps in Salesforce.

This is a mandatory property. The corresponding adapter command is -CS (or -CONSUMERSECRET).

Refresh token

Specifies the refresh token to provide when performing Refresh Token expiry action.

The corresponding adapter command is -RT (or -REFRESHTOKEN).

Security Token

Specifies the security token value to perform when performing Request New Token expiry action.

The corresponding adapter command is -ST or -SECURITYTOKEN command.

Token Storage File

Specifies the location of the file in the local file system in which to store the access token retrieved after performing Refresh Token or Request New Token expiry action.

This is a mandatory property. The corresponding adapter command is -TF (or -TOKENFILE).

Encryption key

Specifies the key to use for encrypting the access token with AES-256 encryption method when saving the token in the token storage file.

This is a mandatory property. The corresponding adapter command is -EK (or -ENCRYPTIONKEY).

Object Name

This property specifies the object name to be acted upon when action Get Record, Insert Record, Update Record or Delete Record is selected.

The default value is Get Record. This is a mandatory property. The corresponding adapter command is -SON (or -SOBJECTNAME).

Id

Specifies the record ID that is used for fetching, updating or deleting any record.

This is a mandatory property. The corresponding adapter command is -SOID (or -SOBJECTID).

SOQL Query

SOQL Query is enabled, when you select the Execute Query, Execute QueryAll, or Query Performance action.

Make sure that it contains valid SOQL query string. This is a mandatory property. The corresponding adapter command Is -SOQL (or -SOQLQUERY).

Refer to Salesforce Object Query Language (SOQL) for valid SOQL syntax.

Search string

Specifies the search parameter that is used for simplified searching in Salesforce data instead of a SOQL clause.

The corresponding adapter command -SS (or -SEARCHSTRING).

Logging

Specifies the level of logging to use for the log (trace) file produced by the adapter.

The default is off. The value info means log informational and error messages, the value error means log error messages only, and the value verbose means log debug and trace level messages along with the informational and error messages.

Append Log

The Append Log is a flag that indicates what to do if the specified log file already exists.

The default value is true. When set to true, the log messages are appended to the file. When set to false, the file is truncated, and the messages are written to the empty file.

Log File Path

Name of the log file to which to write log messages.

If not specified, the default log file name m4sforce.trc is used, and the file is stored to the directory in which the executed compiled map resides.

Examples

- [GET map function](#)
- [PUT map function](#)

GET map function

Access token expiry action (-ATEA) is set to reportError, the adapter will throw 401 – Unauthorized error, if token is invalid or expired.

```
GET("SFORCE", " -E getRecord -ATEA reportError -AT access_token -IURL https://resourceful-hawk-wrhtdc-dev-ed.my.salesforce.com -SV 46.0 -SON Account")
```

To refresh the access token when it expires, set ATEA to 'refresh_token' (Refresh Token). -RT is provided with refresh token captured outside HIP/ITX along with consumer key:

```
GET("SFORCE", " -E getRecord -ATEA refresh_token -CK consumerKey -AT accessToken -RT refreshToken -TF filepath -EK ekey -IURL https://resourceful-hawk-wrhtdc-dev-ed.my.salesforce.com -SV 46.0 -SON Account")
```

To get new token, if it expires, ATEA is set to 'password' (Request New Token) along with username, password, consumer key and consumer secret. Optionally security token, if provided by Salesforce:

```
GET("SFORCE", " -E getRecord -ATEA password -UN username -PWD pwd -CK consumerKey -CS consumersecret -AT accessToken -ST secToken -RT refreshToken -TF filepath -EK ekey -IURL https://resourceful-hawk-wrhtdc-dev-ed.my.salesforce.com -SV 46.0 -SON Account")
```

PUT map function

```
PUT("SFORCE ", " -E createRecord -ATEA reportError -AT accessToken -IURL https://resourceful-hawk-wrhtdc-dev-ed.my.salesforce.com -SV 45.0 -SON Account ",  
inputdata)
```

Secure File Transfer Protocol (SFTP) Adapter

Overview

With the Secure File Transfer Protocol (SFTP) adapter, you can send and fetch files securely to and from an SFTP server. When you use the SFTP adapter as a source, you can read data from a file as well as fetch a list of files present in any directory on an SFTP server location. When you use the SFTP adapter as a target, you can send data to a specified file on the SFTP server. You can either:

- Create a new file.
- Overwrite an existing file.
- Append data to an existing file.

If you have configured the adapter to overwrite a file or append data to a file that does not exist—but if the directory structure in the file path is valid—the file is automatically created. If the directory structure in the specified file path does not exist, the write operation fails unless you have configured the adapter to create a new directory in case the specified file path does not exist. In that case, the adapter automatically creates the missing directories.

Introduction

The SFTP is a protocol for secure file data transfer over computer networks. The SFTP adapter conforms to this protocol and can be configured to act as an SFTP client to transfer and access file data to and from SFTP servers.

Command alias

Specify adapter commands by using a command string on the command line or by creating a command file that contains adapter commands. The command syntax is:

```
-IA[alias]card_num
```

```
-OA[alias]card_num
```

In the command syntax, -IA is the Input Source Override execution command and -OA is the Output Target Override execution command, *alias* is the adapter alias, and *card_num* is the number of the map card. The SFTP adapter alias and corresponding execution commands are listed in the following table.

Adapter	Alias	As Input	As Output
SFTP	SFTP	-IASFTP	-OASFTP

Authenticating connections

The Secure File Transfer Protocol (SFTP) is a protocol for secure file data transfer over computer networks. The SFTP adapter conforms to this protocol and can be configured to act as an SFTP client to transfer and access file data to and from SFTP servers.

The SFTP adapter offers the following two methods to authenticate a user to connect to an SFTP server:

- Password-based – You need to provide the correct password.
- Keypair-based – You need to enter the private key file path, public key file path, keypair name, and passphrase.

You can select any one of the two authentication methods as the preferred one. The adapter first attempts the preferred authentication method. If that attempt is not successful, the adapter then tries for the other method to authenticate.

Note: For Keypair-based authentication method, only the private-key file path is required. The public-key file path, keypair name, and passphrase are optional. Username is required for both the authentication methods.

After successfully connecting to an SFTP server, you can send and receive files to and from the server.

SFTP adapter as a source

SFTP adapter as a source supports:

- File list – Use this mode to fetch all the files from any directory on an SFTP server location.
- File data – Use this mode to fetch the content of a specified file from an SFTP server location.

SFTP adapter as a target

SFTP adapter as a target supports:

- Create – Use this mode to create a new file in the SFTP server location and write data to it, or report an error if the file already exists.
- Overwrite – Use this mode to overwrite a file with any new data if the file already exists; or otherwise to create a file and write data to it—if the specified file does not exist.
- Append – Use this mode to append any new data to an existing file on the SFTP server location, or otherwise to create a new file and write data to it—if no file of the specified name exists.

Note: When you fetch a file, the file is left on the server or removed based on the value specified in the File Success Action property.

Adapter properties and commands

This section lists the properties supported by the adapter.

Host

Specifies the host name or IP address of the SFTP server. This is a mandatory property. The corresponding adapter command is `-HOSTname` (or `-HOSTNAMEname`).

Port

Specifies the port on which the SFTP server is listening for incoming connections. The default value is 22. The corresponding adapter command is `-PORT number` (or `-PORTNUMBER number`).

Preferred authentication

Specifies the authentication method to try first when establishing a connection. The supported values are Password and Public Key. The default value is Password. The corresponding adapter command is `-PA method` (or `-PREFERREDAUTHENTICATION method`) where the `method` is one of password or public key.

User

Specifies the username for the connection. This is a mandatory property. The corresponding adapter command is `-USR name` (or `-USERNAME name`).

Password

Specifies the password value to use when performing password-based authentication. The corresponding adapter command is `-PWD value` (or `-PASSWORD value`).

Known hosts file path

Specifies the location of the known hosts file. The value is optional, and when specified results in enforcing strict host name checking against the specified file when establishing connections. If not specified, it results in omitting strict host name-checking. The corresponding adapter command is `-KHFP path` (or `-KNOWNHOSTSFILEPATH path`).

Key pair name

Specifies the name of the keypair to use when performing keypair based authentication. The value is optional and when not specified the name identifying the key pair is derived from the private key file name. The corresponding adapter command is `-KPN name` (or `-KEYPAIRNAME name`).

Private key file path

Specifies the location of the private key file to use when performing keypair based authentication. The corresponding adapter command is `-PRKFP path` (or `-PRIVATEKEYFILEPATH path`).

Passphrase

Specifies the passphrase is used to access the key data when performing keypair based authentication. The value is optional, and when not specified, it is assumed that the key is not passphrase protected. The corresponding adapter command is `-PSP value` (or `-PASSPHRASE value`).

Connection timeout

Specifies the amount of time, in seconds, to wait for the server to respond when establishing a connection, before considering the connection attempt unsuccessful. The default value is 30. The value 0 indicates no timeout. The corresponding adapter command is `-CT duration` (or `-CONNECTIONTIMEOUT duration`).

Retry connection count

Specifies the number of times to retry establishing a connection to the server before giving up and reporting the error. The default value is 0, which indicates no retries. This setting is not applicable when performing Test connection operation in the design UI, and the value 0 is automatically enforced in that case. The corresponding adapter command is `-RCC count` (or `-RETRYCONNECTIONCOUNTcount`).

Retry connection interval

Specifies the amount of time, in seconds, to wait between successive connection retries. The default value is 60. This setting is applicable only when the Retry Connection Count is set to a value greater than 0.

The corresponding adapter command is `-RCI duration` (or `-RETRYCONNECTIONINTERVAL duration`).

Path

Specifies the path of the directory or file to access on the SFTP server. Forward slash (/) character is used as a file separator. If the specified path value starts with a forward slash, it is treated as an absolute path; otherwise, it is treated as a path relative to the current default directory for the connection. The corresponding adapter command is `-P path` or `(-PATH path)`.

Create directory

Specifies the setting which when enabled results in the automatic creation of all missing directories in the specified directory path. When not enabled, the adapter reports an error if the specified directory structure does not exist on the SFTP server. The setting is applicable only when the adapter is used as a target. The corresponding adapter command is `-CD` (or `-CREATEDIRECTORY`).

Read mode

This property specifies the mode in which to use the adapter as a source. The File Data mode is used to read the content of the file specified in the Path setting. The File List mode is used to retrieve the list of absolute paths of the files located in the directory specified in the Path setting. The default value is File Data. The corresponding adapter command is -RM *mode* (or -READMODE *mode*), where *mode* is one of file_data or file_list.

Write mode

Specifies the mode in which to use the adapter as a source. The File Data mode is used to read the content of the file specified in the Path setting. The File List mode is used to retrieve the list of absolute paths of the files located in the directory specified in the Path setting. The default value is File Data. The corresponding adapter command is -WM *mode* (or -WRITEMODE *mode*), where *mode* is one of create, overwrite or append.

File Success Action

Specifies the success action to be taken by the adapter on the file as what to do with it after the successful run of a transaction. The default value is Keep.

The corresponding adapter command is -FILESUCCESSACTION (or -FSA). The following actions are supported:

- Keep: Keeps the file at the source after the successful run of a transaction. The corresponding adapter command value is keep.
- Delete Always: Deletes the file at the source in case of successful run of a transaction. The corresponding adapter command value is delete_always.
- Delete If Empty: Deletes the file at the source in case of a successful run. Checks, if the file is empty or not. If the file is empty, it is deleted, otherwise it is left in place. The corresponding adapter command value is delete_if_empty.

Failure Action

Specifies the failure action to be taken by the adapter on the file as where to place it after the failure run of a map. The default value is Commit.

The following actions are supported:

- Commit: This option will commit the transaction in case of transaction failure.
- Rollback: This option will rollback the transaction in case of transaction failure.

The following list shows the outcomes for different combinations of File Success Action and Failure Action property values:

- keep + commit = keep
- delete always + commit = delete
- delete if empty + commit (when file has data) = keep
- delete if empty + commit (when file does not have data) = delete
- keep + rollback = keep
- delete + rollback = keep
- delete if empty + rollback (when file has data) = keep
- delete if empty + rollback (when file does not have data) = keep

Logging

This property specifies the level of logging to use for the log (trace) file produced by the adapter. The default is Off. The value Information means log informational, the value Errors Only means log error messages only, and the value Verbose means log debug and trace level messages along with the informational and error messages.

The corresponding adapter command is:

-T [E|V] [+] [file_path]

-T -> Log adapter informational messages.

-TE -> Log only adapter errors.

-TV -> Use verbose (debug) logging. The log file records all activity that occurs while the adapter is producing or consuming messages.

+ -> Appends the trace information to the existing log file. Omit this argument to create a new log file.

file_path -> The full path to the adapter trace log. If you omit this keyword, the adapter creates the m4sftp.mtr log file in the map directory.

Append log

Flag indicating what to do if the specified log file already exists. When set to true, the log messages are appended to the file. When set to false, the file is truncated, and the messages are written to the empty file. The default value is true.

Log file path

Specifies the location of the log file to which to write log messages. If not specified, the default log file name m4sftp.mtr is used, and the file is stored to the directory in which the executed compiled map resides.

Convert Data

Specifies to convert data to parsed form. If enabled, support for converting CSV, Avro, Excel, Parquet and JSON documents into sets of data records in the CSV format. This means that the data produced by the converters can be loaded directly into relational databases.

Data Format

Specifies the data format. This is a mandatory property. Select one of the following as per the data format type. This property field is applicable when Convert Data property toggle is enabled.

- CSV
- Excel
- Avro
- Parquet
- JSON

Header

If enabled, specifies whether the CSV data contains a header row.

Quote

Specifies the quote character. Default value is ".

Delimiter

Specifies the field delimiter character. Default value is ,.

- Escape**
Specifies the escape character.
- Separator**
Specifies the row separator characters. Default value is \r\n.
- Character Set**
Specifies the character set of the data. Default value is UTF-8.
- Limit**
Specifies the maximum number of records to return.
- Sample Size**
Specifies the number of records to analyze when determining the data structure. Default value is 100.
- Worksheet Index**
Specifies the index of the worksheet. The index of the first worksheet is 1 (default).
- Select Source Tables**
Select which arrays of objects in the AVRO data you wish to map to tables in the target.
- Find Array**
If enabled, specifies that the output will iterate on the first array found in the JSON document.
- Array Path**
Specifies the path of the JSON array within the document. Omit if the document is an array. For example, "/resources".

Examples

Examples of the GET map function

In the following example, assume that the adapter is used in a GET function.

- When the adapter is used to fetch file content:

```
GET("SFTP", "-USR testUser -HOST host1 -PORT 22 -PA Password -PWD pwd1 -P /myDirectory/MyFile.txt -RM file_data -TV
/tmp/logfile.log")
```

- When the adapter is used to fetch file list:

```
GET("SFTP", "-USR testUser -HOST host1 -PORT 22 -PA Password -PWD pwd1 -P /myDirectory/ -RM file_list -TV
/tmp/logfile.log")
```

- When the adapter is used to File Success Action:

```
GET("SFTP", "-USR testUser -HOST host1 -PORT 22 -PA Password -PWD pwd1 -P /myDirectory/MyFile.txt -RM file_data -FSA keep
-TV/tmp/logfile.log")
```

Examples of the PUT map function

In the following example, assume that the adapter is used in a PUT function.

- When the adapter is used to create a new file, if the file is not present:

```
PUT ("SFTP", "-USR testUser -HOST host1 -PORT 22 -PA Password -PWD pwd1 -P /myDirectory/MyFile.txt -WM create -TV
/tmp/logfile.log", in_data_type)
```

- When the adapter is used to overwrite an existing file:

```
PUT ("SFTP", "-USR testUser -HOST host1 -PORT 22 -PA Password -PWD pwd1 -P /myDirectory/MyFile.txt -WM overwrite -TV
/tmp/logfile.log", in_data_type)
```

- When the adapter is used to append data to an existing file:

```
PUT ("SFTP", "-USR testUser -HOST host1 -PORT 22 -PA Password -PWD pwd1 -P /myDirectory/MyFile.txt -WM append -TV
/tmp/logfile.log", in_data_type)
```

- When the adapter is requested to automatically create missing directories:

```
PUT ("SFTP", "-USR testUser -HOST host1 -PORT 22 -PA Password -PWD pwd1 -P /myDirectory/MyFile.txt -WM create -CD -TV
/tmp/logfile.log", in_data_type)
```

Example of RUN() function in a map rule

- Here is an example of using RUN() function in a map rule to run a map map1 and override its output card 1 to use SFTP adapter:

```
=RUN("map1", "-OASFTP1 '-HOST server1 -PORT 10022 -USR user1 -PWD password1 -PATH /tmp/test.txt -WM overwrite -TV'")
```

- Here is the same example, but this time using command server from command line on Linux to run the map map1.lnx and perform the same override:

```
dtxcmdsv map1.lnx "-OASFTP1 '-HOST server1 -PORT 10022 -USR user1 -PWD password1 -PATH /tmp/test.txt -WM overwrite -TV'"
```

SNMP Adapter overview

The SNMP Adapter is used to create and send custom-defined traps to an SNMP manager. A *trap* is a message indicating that an event has occurred on the host running the network resource.

The generic term *SNMP manager* is used in this documentation when referencing SNMP-enabled network management software.

A typical use scenario is a data-sensitive trigger or a particular condition that occurs during a map execution. For example, using conditional logic and the `PUT` function in a map, a trap is sent when a datum exceeds a specified value.

The SNMP Adapter can be used only in an output card or `PUT` function (data target). For example,

```
=IF (Price:InputStream > 1000000,  
PUT ("SNMP", "-T", "Threshold exceeded (" +  
TEXT(Price:InputStream) + ")"), NONE)
```

A trap community must be defined in the `dtx.acl` file that specifies which IP address or domains are to receive trap notifications. The trap community identifies the community and hosts that can receive the traps. You can specify an IP address that is in IPv4 (for example, 1.2.3.4) or IPv6 (for example, a:b:c:d:0:1:2:3) format.

IPv4 format example:

```
trap = { trap-community = mycommunity  
hosts = 192.168.100.1,147.137.16.1
```

IPv6 format example:

```
trap = { trap-community = mycommunity  
hosts = 1080:0:0:0:8:700:200C:417A, 1081:0:0:0:8:800:200C:418B
```

- [System requirements](#)
- [Command alias](#)
- [SNMP Adapter commands](#)
- [Return codes and error messages](#)

System requirements

The minimum system requirements and operating system requirements for the SNMP Adapter are detailed in the release notes.

Command alias

Adapter commands can be specified by using a command string on the command line or creating a command file that contains adapter commands. The execution command syntax is:

```
-IA[alias] card_num  
-OA[alias] card_num
```

where `-IA` is the Input Source Override execution command and `-OA` is the Output Target Override execution command, `alias` is the adapter alias, and `card_num` is the number of the input or output card. The following table shows the adapter alias and its execution command.

Adapter	Alias	As Input	As Output
SNMP	SNMP	<code>-IASNMP</code> <i>card_num</i>	<code>-OASNMP</code> <i>card_num</i>

SNMP Adapter commands

This documentation describes the functions and use of the SNMP Adapter commands and their options.

- [List of commands](#)
- [Context name \(-CONTEXTNAME or -N\)](#)
- [Port \(-PORT or -P\)](#)
- [Security level \(-SECURITYLEVEL or -L\)](#)
- [Trace \(-TRACE or -T\)](#)
- [User \(-USER or -U\)](#)
- [Version \(-VERSION or -V\)](#)
- [XML \(-XML\)](#)
- [Example](#)
- [Example files](#)

List of commands

The following table lists valid commands for the SNMP Adapter, the command syntax, and if the command is supported (✓) for use with data sources, targets, or a combination.

Commands can be specified on the adapter's command line or through adapter properties. The commands are not case sensitive but each value is case sensitive.

Command	Syntax	Source	Target
Context name (-CONTEXTNAME or -N)	<code>-CONTEXTNAME</code> <i>context_name</i>		✓
Port (-PORT or -P)	<code>-PORT</code> <i>port_number</i>		✓
Security level (-SECURITYLEVEL or -L)	<code>-SECURITYLEVEL</code> 0 1 3		✓
Trace (-TRACE or -T)	<code>-T[E][+]</code> [<i>full_path</i>]		✓
User (-USER or -U)	<code>-USER</code> <i>user_name</i>		✓

Command	Syntax	Source	Target
Version (-VERSION or -V)	-VERSION 1 2 3		✓
XML (-XML)	-XML		✓

Context name (-CONTEXTNAME or -N)

Use the context name (-CONTEXTNAME or -N) adapter command to specify the name of the SNMPv3 context that is authorized.

Option	Description
<code>context_name</code>	The context name specified in the dtx.uacl file.

Port (-PORT or -P)

Use the Port adapter command (-PORT) to specify the port number to receive trap notifications. The SNMP manager installed on the machine that is connected to that port will receive trap notifications.

Option	Description
<code>port_number</code>	The port number to receive trap notifications. The default port number is 162 .

Security level (-SECURITYLEVEL or -L)

Use the security level (-SECURITYLEVEL or -L) adapter command to set the level of security for communication in the SNMPv3 network, as specified in the **dtx.uacl** file.

Option	Description
<code>security_level</code>	One of the following values:
0	No authentication and no privacy (noAuthNoPriv).
1	Authentication and no privacy (authNoPriv).
3	Authentication and privacy (authPriv).

Trace (-TRACE or -T)

Use the Trace adapter command (-TRACE or -T) to produce a diagnostics file that contains detailed information about SNMP Adapter activity.

The default filename is **m4snmp.mtr** and it is created in the map directory unless otherwise specified.

-T [E] [+] [*full_path*]

Option	Description
<code>E</code>	Produce a trace file containing only the adapter errors that occurred during map execution.
<code>+</code>	Append trace information to the existing trace file.
<code>full_path</code>	Creates a trace file with the specified name in the specified directory.

You can override the adapter command line trace options dynamically using the Management Console. See "Dynamic Adapter Tracing" in the Launcher documentation for detailed information.

User (-USER or -U)

Use the user (-USER or -U) adapter command to specify the name of the SNMPv3 user that is authorized.

Option	Description
<code>user_name</code>	The user name specified in the dtx.uacl file.

Version (-VERSION or -V)

Use the version (-VERSION or -V) adapter command to specify the version of the SNMP network protocol.

Option

Description

version

One of the following values:

- 1 SNMPv1. This is the default version.
- 2 SNMPv2c
- 3 SNMPv3

XML (-XML)

Use the XML adapter command (-XML) to specify that the data being sent to the SNMP Adapter is in XML format.

The default message format is plain text.

The format of the XML message is as follows:

```
<Message>
<String><! [CDATA[ string data ]]></String>
<Integer>integer value</Integer>
<Binary>base64 encoded binary data</Binary>
<Priority>integer value [0-6]</Priority>
</Message>
```

This format allows for binary data and an integer variable to be sent in the trap message. This enables an error code to be sent in addition to the data, and any other kind of data that the permitting binary data allows. If the XML format is not used, then only the text variable is sent in the trap.

A single MIB variable, the string or binary value, is limited to 64K due to a limitation in the SNMP protocol.

Example

For the PUT Target setting in an output card, select SNMP. In the **PUT > Target > Command** field, enter:

```
PUT ("SNMP", "-T+ output.txt -XML -
PORT 162", PACKAGE(TrapInfo))
```

This command specifies the following:

- Appends trace information to **output.txt**.
- Receives input from type tree in XML format.
- Sends traps to SNMP managers connected to port **162**.
- Sends the output from an output card named **TrapInfo**.

Example files

An example program is provided in the *install_dir\examples\adapters\sntp* directory. This example demonstrates how to send SNMP traps using the SNMP Adapter.

Return codes and error messages

Return codes and error messages are returned when the particular activity completes. Return codes and error messages might also be recorded as specified in the audit logs, trace files, and execution summary files.

- [SNMP Adapter messages](#)

SNMP Adapter messages

The following lists all the codes and messages that can be returned as a result of using the SNMP Adapter for sources or targets.

Adapter return codes with positive numbers are warning codes that indicate a successful operation. Adapter return codes with negative numbers are error codes that indicate a failed operation.

Return Code	Message
-------------	---------

Return Code	Message
0	Success
999	MPIRC_E_SNMP_EXCEPTION
998	MPIRC_E_SNMP_TOO_MANY_COMMANDS_EXCEPTION
997	MPIRC_E_SNMP_MISSING_CMDLINEARG_EXCEPTION
996	MPIRC_E_SNMP_INVALID_PORT_EXCEPTION

SOAP (WSDL) Adapter

SOAP defines the XML-based message format that applications use to communicate and inter-operate with each other over the Web.

The heterogeneous environment of the Web demands that applications support a common data encoding protocol and message format.

SOAP is a standard for encoding messages in XML that invoke functions in other applications running on any hardware platform regardless of different operating systems or programming languages. It is analogous to Remote Procedure Calls (RPC) used in many technologies such as Distributed Component Object Model (DCOM), but eliminates some of the complexities of using these interfaces.

The SOAP (WSDL) Adapter supports the SOAP 1.1 protocol as defined by <http://www.w3.org/TR/SOAP/>.

Use the SOAP (WSDL) Adapter to help to create SOAP request messages and to interpret response messages. The SOAP (WSDL) Adapter does not handle the delivery of messages and should be used with a transport adapter, such as HTTP, to deliver the request and receive the response message.

When you send a SOAP-based HTTP request, the default encoding of the content of the HTTP request is ISO-8859-1. The SOAP adapter requires a UTF-8 encoded response from the HTTP server, but the request can be in any Western or UTF-8 encoded format. See the description of the **-TYPE** command in the HTTP adapter documentation for details about how to specify the encoding.

The adapter has the following capabilities:

- Adds SOAP envelopes if they are not provided
- Parses the response message to extract scalar data elements
- Removes envelope and simplifies the form of SOAP responses to aid mapping of the response data
- Understands fault codes and interprets them as map failures

The SOAP (WSDL) Adapter does not provide a listener. The transport adapter provides this functionality.

- [System requirements](#)
- [Command alias](#)
- [SOAP Adapter commands](#)
- [Syntax summary](#)
- [Using the adapter](#)
- [Return codes and error messages](#)

System requirements

The minimum system requirements and operating system requirements for the SOAP (WSDL) Adapter are detailed in the release notes.

Command alias

Adapter commands can be specified by using a command string on the command line or by creating a command file that contains adapter commands. The execution command syntax is:

```
-IA[alias] card_num
-OA[alias] card_num
```

where **-IA** is the Input Source Override execution command and **-OA** is the Output Target Override execution command, **alias** is the adapter alias, and **card_num** is the number of the input or output card. The following table shows the adapter alias and its execution command.

Adapter	Alias	As Input	As Output
SOAP	SOAP	-IASOAPcard_num	-OASOAPcard_num

SOAP Adapter commands

This documentation describes the functions and use of the SOAP (WSDL) Adapter commands and their options.

- [List of commands](#)

List of commands

The following table lists valid commands for the SOAP (WSDL) Adapter, the command syntax, and whether the command is supported (✓) for use with data sources, targets, or any other combination.

Commands can be specified on the adapter's command line or through adapter properties. Note that the commands are not case sensitive but that each value is case sensitive.

Command	Syntax	Source	Target
Decode (-DECODE)	-DECODE	✓	
Encode (-ENCODE)	-ENCODE		✓
Header (-HDR)	-HDR	✓	
Raw (-RAW)	-RAW	✓	
Return (-RETURN)	-RETURN element	✓	
SOAPAction (-SA)	-SA soap_action		✓
Trace (-T)	-T[E][NX][+] [filename]	✓	✓
Transport (-TRANSPORT)	-TRANSPORT 'adapter_name (adapter_command_line)'	✓	✓

- [Decode \(-DECODE\)](#)
- [Encode \(-ENCODE\)](#)
- [Header \(-HDR\)](#)
- [Raw \(-RAW\)](#)
- [Return \(-RETURN\)](#)
- [SOAPAction \(-SA\)](#)
- [Trace \(-T\)](#)
- [Transport \(-TRANSPORT\)](#)

Decode (-DECODE)

Use the Decode adapter command (-DECODE) to instruct the adapter to decode the SOAP message, which involves stripping the envelope and SOAP envelope header, and handling SOAP faults.

-DECODE

For example, to decode a message received over HTTP:

```
GET ("SOAP", "-DECODE -TRANSPORT 'HTTP (-URL http:/myurl/)'" )
```

The -DECODE option is optional since it is implied by the invocation of the SOAP (WSDL) Adapter in an input card or GET function.

For more information on encoding and decoding data, see *Encoding and Decoding Data* in the Resource Adapters documentation.

Encode (-ENCODE)

Use the Encode adapter command (-ENCODE) to instruct the adapter to encode the SOAP message, which involves adding a SOAP envelope if none is provided.

-ENCODE

For example, to encode a SOAP message before transporting over HTTP:

```
PUT ("SOAP", "-ENCODE -TRANSPORT 'HTTP (-URL http:/myurl/)'  
SoapRequest )
```

The -ENCODE option is optional since it is implied by the invocation of the SOAP (WSDL) Adapter in an output card or PUT function.

For more information on encoding and decoding data, see *Encoding and Decoding Data* in the Resource Adapters documentation.

Header (-HDR)

Use the Header adapter command (-HDR) to specify that the SOAP envelope header be returned. This is an optional command. The default is to not return the SOAP header.

-HDR

See [Source Options Behavior](#) for more information on the behavior of the SOAP (WSDL) Adapter and its combinations of available options.

Raw (-RAW)

Use the Raw adapter command (-RAW) to specify that the response message be returned unmodified, except the SOAP envelope header is removed.

-RAW

If the **-RAW** command is specified, the only processing of the response data is:

The SOAP envelope header is stripped out. To keep the header specify **-HDR**.

Fault codes are interpreted

For example, using the **-RAW** command results in the data being returned as follows (intact, but without SOAP envelope header):

```
<?xml version="1.0" encoding="UTF-8" standalone="no" ?>
- <SOAP-ENV:Envelope SOAP-
  ENV:encodingStyle=http://schemas.xmlsoap.org/soap/encoding/
  xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
- <SOAP-ENV:Body>
- <SOAPSSDK1:AddResponse
  xmlns:SOAPSSDK1="http://tempuri.org/message/">
  <Result>290</Result>
</SOAPSSDK1:AddResponse>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

If the **-RAW** command is not specified, then the following is returned:

- The body of the request without the enclosing **<Body>...</Body>** tags
- Any namespace prefixes are removed
- Any attributes are removed

For example, the message shown above is returned as follows:

```
- <AddResponse>
  <Result>290</Result>
</AddResponse>
```

See [Source Options Behavior](#) for more information on the behavior of the SOAP (WSDL) Adapter and its combinations of available options.

Return (-RETURN)

Use the Return adapter command (**-RETURN**) to return a single scalar value from the response message. In many cases, a map only has interest in a single value. This option simplifies mapping in these types of cases. This command can only be used if the value of the element is a scalar value.

This command is incompatible with the **-HDR** command.

This is an optional command. The default is a null reply.

-RETURN element

Option

Description
element

Specify the path and the element to be returned using XPath syntax.

XPath is a language for addressing parts of an XML document. For more information on the XPath language, go to www.w3.org/TR/xpath.

For example, to specify /cat/dog as the XML element to be returned:

-RETURN /cat/dog

See [Source Options Behavior](#) for more information on the behavior of the SOAP (WSDL) Adapter and its combinations of available options.

- [Return option and XPath](#)

Return option and XPath

The element to return is specified using an XPath subject to the following limitations:

- The element to return must be a scalar type. That is, it cannot be an array or contain another element.
- Only fully qualified paths are accepted, for example:
/Message/StockQuote/Price
- Attributes of elements can be specified using the standard XPath '@' notation, for example:
/Message/StockQuote/@currency

While the SOAP standard does not forbid the use of attributes, it recommends not using them.

No other XPath syntax is supported

The limitations in the previous examples are taken into consideration in the following sample message:

```

<Message>
  <StockQuote>
    <Symbol>XYZ</Symbol>
    <Price currency="dollar">20</Price>
  </StockQuote>
</Message>

```

The names of the elements in the path correspond to the name in the returned data. Therefore, if the `-RAW` command is used, the names correspond to the element names in the raw response message. In this case, the elements are often qualified by namespace prefixes.

Consider the following example return message:

```

<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:SOAP-ENC="http://schemas.xmlsoap.org/soap/encoding/">
<SOAP-ENV:Body SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
<NS1:BabelFishResponse xmlns:NS1="urn:BorlandBabelIntf-IBorlandBabel">
<return xsi:type="xsd:string">Freeend</return>
</NS1:BabelFishResponse>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>

```

If we want to return only the `<return>` element, the following command(s) should be used:

```

-Raw -Return /SOAP-ENV:Envelope/SOAP-ENV:Body/
NS1:BabelFishResponse/return

```

Or, if the `-RAW` command is not specified:

```

-Return /BabelFishResponse/return

```

SOAPAction (-SA)

When the transport is HTTP, many Web services require the `SOAPAction` field to be added to the HTTP header of a SOAP request. Use the `SOAPAction` adapter command (`-SA`) to indicate to the server the intent of the request.

Use one of the following values:

- `NULL` (`-SA` with no argument), means that there is no indication of the intent
- `Empty string` (`-SA ""`), means that the intent of the request is provided by the HTTP request URI or Uniform Resource Identifier (no value is specified between the single quotes)
- `URI` (`-SA 'http://www.me.com/'`), where the intent is provided by the URI

The presence and content of the `SOAPAction` header field can be used by servers to appropriately filter SOAP request messages.

This is an optional command. The default is no `SOAPAction`.

```

-SA soap_action

```

Option

Description
<code>soap_action</code>

Specify the intent of the request for the SOAP message.

When using the HTTP adapter as the transport, the HTTP adapter command `-HDR+/-URL` must be used to instruct the HTTP adapter to add the specified `SOAPAction` to the HTTP header.

For example, to specify `urn:BorlandBabelIntf-IBorlandBabel#BabelFish` as the `SOAPAction`:

```

GET("SOAP", "-SA 'urn:BorlandBabelIntf-
IBorlandBabel#BabelFish' -TRANSPORT 'http(-HDR+ -HDR+ -URL
http://ww6.borland.com/webservices/BorlandBabel/
BorlandBabel.exe/soap/IBorlandBabel -T)'", requestData)

```

If the WSDL Importer is used to create the schemas for the SOAP request messages, the value that should be assigned to the `SOAP action` command can be found in the description of the corresponding operation.

See the Schema Importers documentation for more information on the WSDL Importer.

Trace (-T)

Use the `Trace` adapter command (`-TRACE` or `-T`) to produce a diagnostics file that contains detailed information about SOAP (WSDL) Adapter SOAP (WSDL) Adapter activity.

The default filename is `m4soap.mtr` and the file is created in the map directory unless otherwise specified.

```

-T[E] [NX] [+]
[filename]

```

Option	Description
E	Produce a trace file containing only the adapter errors that occurred during map execution.
NX	Excludes information about XML data from the trace file.
+	Append trace information to the existing trace file.
<i>filename</i>	Creates a trace file with the specified name in the specified directory.

You can override the adapter command line trace options dynamically using the Management Console.

Transport (-TRANSPORT)

Use the Transport adapter command (-TRANSPORT) to specify which transport adapter to use and the command line for the transport adapter.

The syntax of the command must include the single quotes and parentheses as shown below.

```
-TRANSPORT 'adapter_name (adapter_command_line)'
```

Option	Description
<i>adapter_name</i>	Specify which transport adapter to use.
<i>adapter_command_line</i>	Specify the command line for the transport adapter. You must enclose this option within parentheses () .

For example, to specify the HTTP adapter as the transport adapter, and to send the SOAP request to the http://localhost/WebService/test URL, enter the following command:

```
-TRANSPORT 'http (-URL http://localhost/WebService/test -T)'
```

Syntax summary

This documentation discusses the SOAP syntax summary and how it is used.

- [Data sources](#)
 - [Data targets](#)
 - [Source options behavior](#)
-

Data sources

The following is the command syntax for the SOAP (WSDL) Adapter commands used for data sources:

```
-TRANSPORT 'adapter_name (adapter_command_line)'
[-DECODE]
[-RAW]
[-HDR]
[-RETURN element]
[-T[E] [NX] [+]] [filename]
```

- [Example](#)
-

Example

For the GET Source setting in an input card, select SOAP. In the **GET > Source > Command** field, enter:

```
-T -RAW -TRANSPORT 'http (-URL http://www.myws.com/DoWS)'
```

This command specifies that the SOAP message:

- Received data from the HTTP URL <http://www.myws.com/DoWS> using the HTTP adapter (-TRANSPORT)
 - Returned data in full (-RAW)
 - Adapter trace is enabled (-T) and a diagnostics file is produced
-

Data targets

The following is the command syntax for the SOAP adapter commands used for data targets:

```
-TRANSPORT 'adapter_name (adapter_command_line)'  
[-ENCODE]  
[-SA soap_action]  
[-T[E] [NX] [+]] [filename]
```

- [Example](#)
-

Example

For the PUT Target setting in an output card, select SOAP. In the PUT>Target>Command field, enter:

```
-SA " -T -TRANSPORT 'http (-URL http://www.myws.com/DoWS)'
```

This command specifies that the SOAP message:

- Sent data over HTTP to the URL <http://www.myws.com/DoWS> using the HTTP adapter (-TRANSPORT)
 - SOAPAction is provided by the HTTP request URL (SA ")
 - Adapter trace is enabled (-T) and a diagnostics file is produced
-

Source options behavior

The following table defines the behavior of the SOAP (WSDL) Adapter based upon combinations of available options.

Commands	Behavior
No formatting commands	Only the contents of the body of the message are returned (this is the default). No attributes are returned and namespace prefixes are removed.
-RAW alone	If a header is in the message's envelope, it is removed, but the envelope and body are returned intact.
-HDR alone	Equivalent to -RAW and -HDR , since an envelope is required to ensure the well-formedness of the XML document. For more information on XML document standards go to www.w3.org/TR/1998/REC-xml-19980210 .
-RAW and -HDR	The SOAP request is returned in its entirety.
-RETURN XPath	Only the single scalar value is returned. See Return Option and XPath for more information.

Using the adapter

Use the SOAP (WSDL) Adapter to either invoke a Web service by sending a request message and processing the response, or in the implementation of a Web service.

Note that two adapters are used in communicating between the map and the Web service:

- The SOAP (WSDL) Adapter
- A transport adapter, such as an HTTP adapter

There are two primary usage scenarios for the SOAP (WSDL) Adapter:

- As a consumer of Web services
 - As a provider of Web services
 - [Consumer scenario](#)
 - [Provider scenario](#)
 - [Example files](#)
-

Consumer scenario

A common scenario is the flow of a request from a map, through SOAP and HTTP adapters to the Web Service, and the flow of the response back to the map from the Web Service.

The adapter is designed to be called in a GET map function call. The third parameter of the GET is the request message. For example:

```
Response = GET("SOAP", "-HDR -TRANSPORT 'HTTP(-URL http://  
www.stuff.com -T)'", PACKAGE(Request))
```

Adapter Action Sequence

The following describes the sequence of actions that occur in the consumer scenario.

- The request message is first passed to the SOAP (WSDL) Adapter.

- The request message might or might not have a SOAP envelope. If the envelope is not provided, the adapter adds it.
- The request message might or might not have a SOAP envelope header. The adapter does nothing with the header. If it is not provided in the request, then no header is sent.
- The adapter specified in the **-TRANSPORT** command is invoked to issue the request and receive a reply. Typically, this is the HTTP adapter that will then send an HTTP request and return the HTTP response.
- The SOAP (WSDL) Adapter is invoked a second time to process the reply.
- The adapter looks in the body to see if there is a Fault tag. If there is, the SOAP (WSDL) Adapter returns an error message that is comprised of the faultstring and the faultcode.
- The SOAP envelope is removed unless the **-RAW** command is specified.
- If the adapter is called with the **-HDR** command, and there is a SOAP envelope header, the header data will be returned.
- If the adapter is called with the **-RETURN** command, then all XML tags are removed and only the specified datum from the response message is returned. If the response message returns more than one datum, then the whole XML response message is returned with a warning code. This command is mutually exclusive with the **-HDR** command.

Provider scenario

The following descriptions explain using the SOAP (WSDL) Adapter with the HTTP adapter.

- [Response](#)

Response

The HTTP Adapter is invoked to send the response message.

Example files

The `install_dir\examples\web_services` directory contains sample files to be used with Web Services components, including the WSDL Importer, SOAP and HTTP adapters, to implement Web services strategies.

See `readme.txt` located in the example directory for more information about the examples.

Return codes and error messages

Return codes and messages are returned when the particular activity completes. Return codes and messages might also be recorded as specified in the audit logs, trace files and execution summary files.

- [Messages](#)

Messages

The following is a listing of all the codes and messages that can be returned as a result of using the SOAP (WSDL) Adapter for sources or targets.

Adapter return codes with positive numbers are warning codes that indicate a successful operation. Adapter return codes with negative numbers are error codes that indicate a failed operation.

Return Code	Message
-2001	General internal adapter error
-2002	Invalid command line option
-2003	Missing argument in command line option
-2004	Missing command line option
-2005	Invalid argument in command line option
-2006	FC A fault code has been generated.
-2007	Xpath data not found
-2008	Xpath duplicate data found
-2009	Stream error
-2010	

Socket Adapter overview

The Socket Adapter provides a means of passing data to and from IBM Transformation Extender through standard TCP/IP sockets. The adapter provides a listener interface that enables maps to be triggered upon arrival of data on a socket and therefore enables remote applications to pass data to IBM Transformation Extender without the need for additional middleware software.

The Socket Adapter is implemented as a dynamic-link library (DLL) on Windows platforms (**m4sock.dll**) and as a shared object on UNIX platforms (**m4sock.sl** or **m4sock.so**). It is assumed that message formats are binary.

The Socket Adapter supports the following:

- Conversations
- Client or Server mode

The adapter provides these functions:

- Gets one or more messages from a socket for input to IBM Transformation Extender
- Puts a message to a socket as output from IBM Transformation Extender
- Listens for messages and triggers maps to run as messages arrive

For the adapter commands that can be used with this adapter, see the list of adapter-specific commands.

- [System requirements](#)
- [How the adapter works](#)
- [Command alias](#)
- [Socket Adapter commands](#)
- [Syntax summary](#)
- [Troubleshooting](#)

System requirements

The minimum system requirements and operating system requirements for the Socket Adapter are detailed in the release notes. It is assumed that a Command Server has already been installed on the computer where the adapter is to be installed for runtime purposes.

How the adapter works

Because communication over sockets is through streams or bytes, the adapter needs some protocol to be able to distinguish where one message begins and another one ends.

To meet this need, the adapter provides the following options:

- Fixed size messages (**-FIXED**), where the length of each message is a known, fixed value.
- Sized messages (**-SIZED** or **-ISIZED**), where each message is preceded by a value representing the size of that message.
- File message (**-EOF**), where the shut down of the socket defines the message.
- Delimited messages (**-EOM**), where one or more specified characters indicate the end of the message.
- [Client or server mode](#)
- [Conversation support](#)

Client or server mode

The adapter operates either as the client or server of the socket connection. Server mode is only applicable to an input event. If the **mode** flag is not specified, the default is **client**.

In client mode, the listener or input card will connect to the specified remote socket.

In server mode, the listener creates the socket and waits for client connections to attach and send data. On the arrival of a message, a thread is spawned to read the message and process it.

Generally, if multiple cards in a map connect to the same port having the same **-HOST** and **-PORT** or **-SERVICE** then the same socket connection should be used in each card.

Conversation support

When a message is received on a socket connection for an input card or **GET** function, if an output card or **PUT** function in the same map specifies the same host and port number, the connection will be reused for the output card. A response message will be sent to the client that sent the request message.

The communication adapter supports conversations in several scenarios, including:

- **Input is an event** - Listener detects message, **GET** retrieves it, and an output card sends a response to the same socket.
- **Input is not an event - GET** retrieves message and an output card sends a response to the same socket.
- **=GET()** mapping function is used. The third parameter is data to send to a socket, the resulting message is returned from the **GET**. In this case, the same protocol must be used both for the write and the read.

When using the communication adapter in server mode (**-MODE server**), the conversation between client and server can be controlled by a variety of parameters. The most important of these is the Client Connection (**-CCON**) command, which determines how the conversation is ended.

The default value for **-CCON** is exclusive (**excl**), which means that the connection with a client is used exclusively by an instance of a map.

This can be with the Number of Event Messages command (**-NEM**) to initiate a conversational map instance. The **-NEM** command determines how many messages need to be received to start an instance of a map. If this map instance is run in burst mode, each burst might process one or more additional message from the client, and send one or more response back to the client. In this scenario, the map instance is ended when the client disconnects from the socket.

If **-CCON** is set to shared (**shared**) then a single client connection might be used by multiple map instances concurrently. In this scenario, the **-NEM** command cannot be used. The number of messages processed by a single map instance is determined by the Quantity (**-QTY**) and Listen (**-LSN**) commands. The listener receives the number of messages specified by these parameters and these messages are processed by a map instance, which can send one or more response messages to the client.

If the client continues to send messages while one map instance is running, another map instance might be started. Both map instances will use the same client connection. It is the responsibility of the client to associate the request messages with any responses that might be sent from these maps. That using the **-CCON (shared)** command does not preclude multiple clients connecting to the socket.

Command alias

Specify adapter commands by using a command string on the command line or by creating a command file that contains adapter commands. The command syntax is:

```
-IA[alias]card_num
-OA[alias]card_num
```

In the command syntax, **-IA** is the Input Source Override execution command and **-OA** is the Output Target Override execution command, *alias* is the adapter alias, and *card_num* is the number of the map card. The adapter alias and corresponding execution commands are listed below.

Adapter	Alias	As Input	As Output
Socket Adapter	SOCKET	-IASOCKET <i>card_num</i>	-OASOCKET <i>card_num</i>

Socket Adapter commands

This section of the documentation describes the functions and use of the Socket Adapter commands and their options.

- [List of commands](#)

List of commands

Socket Adapter protocol options specify how data is delimited during the transfer process. The following table lists the valid commands for the Socket Adapter, the command syntax, and if the command is supported (✓) for use with data sources, targets, or both.

The following commands can be specified on the adapter's command line or through adapter properties. The commands are not case sensitive, but each value is case sensitive. For ease of use, most commands have a short form and a long form.

Command	Syntax	Source	Target
Client Connection (-CCON)	-CCON excl shared	✓	
Data Header (-DHDR)	-DHDR 0x<hex terminator> -DHDR <i>terminator</i> -DHDR "terminator"	✓	
End Of File (-EOF)	-EOF	✓	✓
End Of Message (-EOM)	-EOM 0x<hex terminator> -EOM <i>terminator</i> -EOM "terminator"	✓	✓
Fixed (-FIXED)	-FIXED <i>size</i>	✓	✓
Host (-H or -HOST)	-HOST name or address	✓	✓
Inclusive Sized (-ISIZED)	-ISIZED	✓	✓
Listen (-LSN)	-LSN <i>seconds</i>	✓	
Mode (-M or -MODE)	-MODE server client	✓	
Number of Event Messages (-NEM)	-NEM <i>number</i>	✓	
Port (-P or -PORT)	-PORT <i>port number</i>	✓	✓
Quantity (-QTY)	-QTY <i>number</i>	✓	
Service (-S or -SERVICE)	-SERVICE <i>service name</i>	✓	✓
Sized (-SIZED)	-SIZED	✓	✓
Trace (-T or -TRACE)	-T[E V][+] [<i>file</i>]	✓	✓
Window Size (-WNDSIZE)	-WNDSIZE <i>size</i>		✓

- [Client Connection \(-CCON\)](#)
 - [Data header \(-DHDR\)](#)
 - [End Of File \(-EOF\)](#)
 - [End Of Message \(-EOM\)](#)
 - [Fixed \(-FIXED\)](#)
 - [Host \(-H or -HOST\)](#)
 - [Inclusive sized \(-ISIZED\)](#)
 - [Listen \(-LSN\)](#)
 - [Mode \(-M or -MODE\)](#)
 - [Number of event messages \(-NEM\)](#)
 - [Port \(-P or -PORT\)](#)
 - [Quantity \(-QTY\)](#)
 - [Service \(-S or -SERVICE\)](#)
 - [Sized \(-SIZED\)](#)
 - [Trace \(-T or -TRACE\)](#)
 - [Window Size \(-WNDSIZE\)](#)
-

Client Connection (-CCON)

The Client Connection adapter command determines if client connections are associated exclusively with a specific map instance, or might be shared between map instances.

`-CCON excl | shared`

Option	Description
<code>excl</code>	The client connection is exclusive to a particular map instance. This is the default client connection mode. Once a client has connected to the adapter that is running in server mode, that connection is associated with the map instance until the client disconnects. The map can involve a conversation with the client, where multiple requests are received from the client, and multiple responses returned. This option should be used in an environment where the client connects to the socket, engages in a conversation, and then disconnects.
<code>shared</code>	The client connection can be shared by multiple map instances. When used in collaboration with the <code>-QTY</code> and <code>-LSN</code> commands, the adapter can be configured to run a map once a number of messages have been received, as specified by the <code>-QTY</code> command, or when a listening time has elapsed, as specified by the <code>-LSN</code> command. This option should be used in an environment where the client connects to the socket, and then remains connected, feeding a continual stream of messages to the Launcher.

See "[Conversation Support](#)" for more information on using this command.

This command option can only be used in input cards, input events to the Launcher, that have a mode of server (`-MODE server`).

Data header (-DHDR)

The data header adapter command (`-DHDR`) discards the message header, which is terminated by the specified terminator.

`-DHDR 0x<hex terminator>`
`-DHDR terminator`
`-DHDR "terminator"`

Option	Description
<code>0x<hex terminator></code>	Hexadecimal value.
<code>terminator</code>	Character string.
<code>"terminator"</code>	Comma-delimited character string.

The "terminator" can contain white space and if the string itself contains quotation marks then you must double them. For example, "This is ""the terminator"".

End Of File (-EOF)

The End Of File (`-EOF`) adapter command defines a single message.

The adapter uses this command when communicating with other programs that invoke and detect calls to the **shutdown** function as defined by the Socket Application Programming Interface (API). The **shutdown** function disables any further send or receive operations on a socket connection. It is used to mark the end of one message.

On a data retrieval (**GET** or input card) operation, the adapter detects the End Of File marker after your program calls the **shutdown** function on the write end of the socket. All data that is received before the End Of File marker defines the single message that is used during map processing.

On a data routing (**PUT** or output card) operation, the adapter sends the End Of File marker to your program by calling the **shutdown** function on the write end of the socket. All data that is sent before the End Of File marker defines the single message that is used by your program.

When you use the End Of File command in a three-parameter **GET** rule, the following results occur. One message is first sent to the client. The message contains all of the data that is referenced by the third parameter. After one message is received from the client, the adapter then closes the socket connection.

When you use the End Of File command in a two-parameter **GET** rule or in an input card, you can also use it in a **PUT** rule or in an output card of the same map. The converse is true as well. When you use the End Of File command in a two-parameter **PUT** rule or in an output card, you can also use it in a **GET** rule or in an input card of the same map. After one data retrieval operation and one data routing operation that both use the End Of File command completes processing, your program can close the socket connection.

Because the End Of File command defines only one message for each **GET** and **PUT** operation, you cannot specify burst mode or the Quantity (**-QTY**) adapter command in the command line.

With the End Of File command used on the command line of an input card for a Launcher watch event, the map is triggered when the Socket Adapter receives one input message from any client or server connection. The adapter sends a reply to that same client or server connection by using the End Of File command on a data routing (**PUT**) operation or an output card. Like the three-parameter **GET** function, the triggered map receives one message and sends one message.

For a Launcher watch event, however, the triggered map can divide the one reply message into multiple data routing operations. For instance, you can design the output cards for modularized processing in the following way. Specify one output card to send the header portion of the message. Specify a second output card to send the body portion. You must specify the host name and port number in the command line on both output cards. You must also specify the End Of File command, but only on the second output card. The first output card causes the Socket Adapter to send the first portion of the message, which is the header. After the map invokes the second output card and the Socket Adapter detects the End Of File command, the adapter sends the last portion of the message, which is the body. It then calls the **shutdown** function to mark the end of the message. Whenever you use the Socket Adapter in this way, remember to use only the End Of File command on the last output card.

End Of Message (-EOM)

The End Of Message (-EOM) command indicates that a specific terminator marks the end of a message. This terminator must not appear in the message data. If this option is specified on output, the terminator characters are appended to the message data. If this option is specified on input, the data is parsed to find the terminator.

```
-EOM 0x<hex terminator>
-EOM terminator
-EOM "terminator"
```

Option

Description
0x<hex terminator>
Hexadecimal value.
terminator
Character string.
"terminator"
Comma-delimited character string.

Fixed (-FIXED)

The fixed command (-FIXED) specifies that the data is a fixed size (n bytes) specified in bytes. This many bytes will be read from or written to the socket.

```
-FIXED size
```

Option

Description
size

Specify that the data is a fixed size (n bytes) and that this many bytes will be read from or written to the socket.

Host (-H or -HOST)

Use the Host (-H or -HOST) command to specify the host name or TCP/IP address of the remote host. If the -HOST command is omitted then the name of the local machine is used.

```
-HOST name or address
```

Option

Description
name or address

The host name or TCP/IP address. For example, use *host_name* or use 192.168.1.120 for the IPv4 version of the IP address, or 1080:0:0:0:8:800:200C:417A, using the IPv6 version of the IP address.

Inclusive sized (-ISIZED)

The inclusive sized (-ISIZED) command is a variant of the -SIZED message type. Using the -SIZED message type, the first four bytes of a message represent the size of the message that follows. Using the -ISIZED message type, the first four bytes of data represent the size of the message that follows, but includes the first four bytes in the total length. Therefore, if the first four bytes of data indicate that the message size is 14 bytes, the message is complete after 10 more bytes.

Listen (-LSN)

Use the listen command (-LSN) to specify a time in seconds to wait for messages to arrive. If not specified, the adapter waits for an infinite amount of time.

The command -LSN 0 means do not wait at all. If no message is immediately available, the adapter returns a warning indicating that no messages were found.

-LSN seconds

Option

Description

seconds

The number of seconds to wait for messages to arrive.

Mode (-M or -MODE)

The mode command (-M or -MODE) sets the connection mode to either **server** or **client**.

-MODE server|client

Option

Description

server

The adapter waits for remote clients to connect and send messages that will trigger the map.

This option is valid only in input cards when the Launcher runs the map.

client

The adapter connects to the remote server and reads messages.

Number of event messages (-NEM)

Used only in Launcher scenarios, this command specifies the number of messages that will trigger maps to launch. The specified number of messages are read into memory and will be available when the map is run.

-NEM number

Option

Description

number

The number of messages that will trigger maps to run.

Port (-P or -PORT)

Use the Port command (-P or -PORT) to specify the TCP/IP port number. When the adapter is used in **server** mode, the port number specifies the port to which remote clients can connect. In **client** mode, the port number is the port on the remote host to which the adapter connects.

-PORT port number

Option

Description

port number

The TCP/IP port number.

Quantity (-QTY)

Use the Quantity command (-QTY) to capture the number of messages to be retrieved by the adapter. If you are running in burst mode, each burst might return less than -QTY messages, but the total number of messages returned by the adapter will not exceed the value specified by the -QTY command. If not specified, the default quantity is **1**.

-QTY number

Option

Description

number

The number of messages to be retrieved by the adapter.

Service (-S or -SERVICE)

Mutually exclusive of the port command (-PORT), use the service command (-S or -SERVICE) to specify the service name that is used to map a TCP/IP port number. The service command and the port command are interchangeable and cannot be used at the same time.

-SERVICE *service name*

Option

Description

service name

The service name that is used to map a TCP/IP port number.

Sized (-SIZED)

The Sized command (-SIZED) specifies that data be sized. The first four bytes of the message represent the size of the message that follows.

Trace (-T or -TRACE)

Use the Trace adapter command (-T or -TRACE) to produce a diagnostics file that contains detailed information about Socket Adapter activity.

The default filename is m4sockt.log and it is created in the map directory unless otherwise specified.

-T [E|V] [+] [*file*]

Option

Description

E

Produce a trace file containing only the adapter errors that occurred during map execution.

V

Verbose mode. Record in the log file all activity occurring while the adapter is retrieving data. The maximum size of the log file is 256 bytes. If you need to trace more or less bytes, set the TX_SOCKET_MAX_BYTES_IN_TRACE_MESSAGE environment variable.

+

Appends trace information to the existing log file; otherwise, the log file is recreated.

file

The log filename.

If a filename contains spaces or other special characters it must be enclosed in quotation marks. Example: "test-file.txt".

You can override the adapter command line trace options dynamically using the Management Console. See "Dynamic Adapter Tracing" in the *Launcher* documentation for detailed information.

Window Size (-WNDSIZE)

The Window Size command (-WNDSIZE) specifies the size in bytes of the buffer in which the End Of Message (EOM) is searched. If not specified, it defaults to 10,000 bytes.

This option is used only if -EOM is specified.

-WNDSIZE *size*

Option

Description

size

Specify the size in bytes of the buffer in which the End Of Message (EOM) is searched.

Syntax summary

This section of the documentation discusses the communication adapter syntax summary and how it is used.

- [Data sources](#)
- [Data targets](#)
- [Examples](#)

Data sources

The following is the command syntax of the communication adapter commands used for data sources:

```
[-CCON {excl|shared}]
[-DHDR 0x<hex terminator>|terminator|"terminator"]
[-HOST name or address]
[-LSN seconds]
[-MODE {server|client}]
[-NEM number]
[-PORT port number]
[-QTY number]
[-SERVICE service name]
[-SIZED|-ISIZED|-FIXED size|-EOF|
-EOM [0x<hex terminator>|terminator|"terminator"]]
[-T[E|V][+] [file]]
```

Data targets

The following is the command syntax of the communication adapter commands used for data targets:

```
[-HOST <name or address>]
[-PORT <port number>]
[-SERVICE <service name>]
[-SIZED|-ISIZED|-FIXED size|-EOF|
-EOM [0x<hex terminator>|terminator|"terminator"]]
[-T[E|V][+] [file]]
[-WNDSIZE size]
```

Examples

The following command example means connect on localhost on port 7352, read three sized messages, and generate a default trace file.

```
-PORT 7352 -SIZED -QTY 3 -TRACE
```

The following command example means start listening on host 192.168.75.32 and port 7352 for one EOM type message for 10000 seconds and generate a default trace file. The map is triggered after the adapter receives one message.

```
-HOST 192.168.75.32 -PORT 7352 -MODE SERVER -NEM 1-EOM "end" -
-QTY S -LSN 10000 -T
```

Troubleshooting

For information about return codes and error messages returned by the adapters, see "Return Codes and Error Messages" in the Resource Adapters documentation.

Standards Processing Engine Adapter

The Standards Processing Engine (SPE) adapter connects IBM Sterling Transformation Extender with Standards Processing Engine (SPE). Specify the SPE adapter on a IBM Sterling Transformation Extender map rule to run the map on SPE and to invoke the enveloping, de-enveloping, and transformation capabilities of a Pack solution.

Read the details about SPE and the associated Pack solutions in the Standards Processing Engine documentation.

- **[System requirements](#)**

The Standards Processing Engine (SPE) adapter function requires the installation of Standards Processing Engine and a solution Pack.

- **[SPE adapter commands and data harness functions](#)**

Standards Processing Engine (SPE) adapter commands fall into two categories: data harness functions and adapter action commands.

System requirements

The Standards Processing Engine (SPE) adapter function requires the installation of Standards Processing Engine and a solution Pack.

See the [IBM Sterling Transformation Extender system requirements](#) for details about the minimum system requirements of the SPE adapter. See the [Standards Processing Engine system requirements](#) for the minimum system requirements of SPE and the associated Pack solution.

SPE adapter commands and data harness functions

Standards Processing Engine (SPE) adapter commands fall into two categories: data harness functions and adapter action commands.

Data harness functions read and write to the underlying SPE database through a data harness, and are valid only when a map is running under SPE.

Adapter action commands are valid in maps that are running on IBM Sterling Transformation Extender Launcher, Command Server, or the TX Programming Interface. The commands:

- Start the SPE envelope and de-envelope functions
 - Run a map on SPE
 - Operate on documents in memory that pass data between IBM Sterling Transformation Extender and SPE
 - Data harness functions**
A IBM Sterling Transformation Extender map can read and write to the Standards Processing Engine (SPE) database through commands that invoke the SPE data harness. Data harness functions are appropriate only in maps that are running under SPE.
 - Data harness function syntax**
The Standards Processing Engine (SPE) adapter supports data harness functions that operate on SPE correlations, process data, code lists, delimiters, and the transaction register. You can use data harness functions only in IBM Sterling Transformation Extender maps that run under SPE.
 - SPE adapter action command overview**
This table summarizes the adapter action commands, syntax, and whether you can use the command with data sources, data targets, or both. You can specify the commands on the adapter command line or through map rules.
-

Data harness functions

A IBM Sterling Transformation Extender map can read and write to the Standards Processing Engine (SPE) database through commands that invoke the SPE data harness. Data harness functions are appropriate only in maps that are running under SPE.

You invoke the data harness functions by configuring either:

- The PUT and GET rules in a map
- An input card or output card command line that specifies the SPE adapter

By using the SPE adapter, maps can read SPE data on input cards and from GET rules, and write to SPE data from output cards and PUT rules. To execute the harness function as soon as the PUT rule is processed, use the -NOW argument. The -NOW argument executes the function even if the map fails.

When the PUT rule does not include the -NOW argument, the function executes when the map completes.

When the map runs successfully, the output from the adapter is associated with an SPE document for use by normal stream adapters.

Data harness function syntax

The Standards Processing Engine (SPE) adapter supports data harness functions that operate on SPE correlations, process data, code lists, delimiters, and the transaction register. You can use data harness functions only in IBM Sterling Transformation Extender maps that run under SPE.

Command Description

`getBinaryDocument`

Retrieves the specified binary document. This command is valid in a GET function.

`getEnvelope`

Retrieves the specified field from an envelope. This command is valid in a GET function.

`setCorrelation`

Updates the Standards Processing Engine correlation table with the specified name/value pair for the document or business process. This command is valid in a PUT function.

`getProcessData`

Retrieves process data from the XML document object model (DOM) specified by the XPath. *Process data* is intermediate data that accumulates in an XML document during document processing. You can use process data to determine the next step in document processing, such as which map to launch based on the result of parsing a document. This command is valid in a GET function.

`setProcessData`

Stores process data from the current field into the XML DOM that is specified by the XPath. *Process data* is intermediate data that accumulates in an XML document during document processing. You can use process data to determine the next step in document processing, such as which map to launch based on the result of parsing a document. This command is valid in a PUT function.

`getCodeListItemBySenderCode`

Retrieves a trading-partner code-list entry from SPE by sender code. A *trading partner code list* consists of one or many pairs of code values containing a sender code and a receiver code. Code pairs in code lists identify items in transactions between two or more trading partners. This command is valid in a GET function.

`getCodeListItemByReceiverCode`

Retrieves a trading-partner code list from SPE by receiver code. A trading partner code list consists of one or many pairs of code values containing a sender code and a receiver code. Code pairs in code lists identify items in transactions between two or more trading partners. This command is valid in a GET function.

`getTransactionRegister`

Compares the fields that you loaded into the Transaction Register with the input data, returns a code, and clears the Transaction Register. This command is valid in a GET function.

Code Meaning

0

Success; no duplicate data found.

1

Failure; duplicate data found.

8

No updates performed to this transaction register yet. Cannot perform a select before an update. The Transaction Register is empty. You must invoke **setTransactionRegister** to load the Transaction Register before you can invoke **getTransactionRegister**.

The following example:

```
PUT("SPE", "-t+ -c setTransactionRegister -k 1", "Reg 1")
GET("SPE", "-t+ -c getTransactionRegister")
GET("SPE", "-t+ -c getTransactionRegister")
```

1. Loads Transaction Register field 1 with the literal value Reg 1.
2. Compares the input document data to the content of the Transaction Register and returns 0 or 1.
3. Attempts to invoke **getTransactionRegister** again without loading the Transaction Register, resulting in return code 8.

The following example loads Transaction Register field 3 with the contents of the specified input data field:

```
PUT("SPE", "-t+ -c setTransactionRegister -k 3", PONumber Field:In1)
```

setTransactionRegister

Loads the specified Transaction Register field with the specified data. You can load up to six fields of data into memory for the Transaction Register. This command is valid in a PUT function.

After you run the **setTransactionRegister** command, you can run the **getTransactionRegister** command to check for duplicate input data.

getDelimiter

Gets a delimiter from an SPE document envelope. This command is valid in a GET function.

validateCodeWords

Validates a series of code words defined by a delimited string. This command takes the code words to validate from the `data_from_map` option instead of the **-KEY** argument. This command is valid in a **GET** function.

This example **PUT** rule uses the SPE adapter to create an element under process data that is called "getmap". The **-NOW** option creates the key even if the map fails:

```
=PUT("SPE", "-t+ -c setProcessData -k getmap -now", "data")
```

This example **GET** rule uses the SPE adapter to read the value of a key called "getmap" from the process data:

```
=GET rule " + VALID( GET("SPE", "-t+ -c getProcessData -k getmap") )
```

This example of a SPE adapter command line on a card creates a correlation that is called "adapter" when the card completes. The correlation includes all of the data that was built on the card:

```
-c setCorrelation -k adapter -t+
```

The SPE adapter supports the harness functions with the following command line arguments:

Table 1. SPE adapter command line arguments

Command Long Form	Command Short Form	Description
-COMMAND	-C	Command
-KEY	-K	Key
-CODE	-D	Sender/Receiver Code
-TABLE	-B	Code table name
-NOW	-N	Executes the harness function as soon as the PUT rule is processed, even if the map fails
-T	-T	Adapter trace (overwrite)
-T+	-T+	Adapter trace (append)
-TE	-TE	Adapter trace error (overwrite)
-TE+	-TE+	Adapter trace error (append)
-TV	-TV	Adapter trace verbatim (overwrite)
-TV+	-TV+	Adapter trace verbatim (append)

The default trace file name is M4spe.mtr.

This table describes the data harness functions and key values that you can use on the SPE adapter:

Table 2. SPE adapter data harness functions and key values

Command	Key
getBinaryDocument	The name of the document to retrieve.
setCorrelation	The name of the correlation to update.
getProcessData	The XPATH to select.
setProcessData	The XPATH to update.
getCodeListItemBySenderCode	The sender value for the code-list entry to retrieve. The command requires a sender code, code list table name, and column.
getCodeListItemByReceiverCode	The receiver value for the code-list entry to retrieve. The command requires a receiver code, code list table name, and column.
getEnvelope	The envelope field name.
getTransactionRegister	There are no keys for this command.
setTransactionRegister	Each entry is a number between 1 and 6 that indicates the field to update.
getDelimiter	An array of numbers that indicates the fields to select. The list is defined in <i>Table 4</i> .
validateCodeWords	This command takes the code words to validate from the <code>data_from_map</code> option instead of from the -KEY argument.

The following table lists the sender and receiver keys and values to use with the **getCodeListItemBySenderCode** and **getCodeListItemByReceiverCode** functions. You define the column names for the code list. The keys refer to the position of the column, not to a specific column name. This pattern continues for as many text columns as you have in the table.

Table 3. Sender and receiver keys
and values

Key	Value
COL_SENDER	0
COL_RECEIVER	1
COL_DESC	2
First_text_column	3
Second_text_column	4
Third_text_column	5
Fourth_text_column	6
Fifth_text_column	7
Sixth_text_column	8
Seventh_text_column	9
Eighth_text_column	10
Ninth_text_column	11

The following table lists the keys and values to use with the **getDelimiter** function.

Table 4. Delimiter keys and values

Key	Value
SEGMENT_TERMINATOR	0
TAG_DELIMITER	1
ELEMENT_DELIMITER	2
SUB_ELEMENT_DELIMITER	3
REPEATING_ELEMENT_DELIMITER	4
RELEASE_CHAR	5
DECIMAL_SEPARATOR	6

The following table lists the keys and values to use with the **setTransactionRegister** function.

Table 5. Transaction Register keys and values

Key	Value
COL_TEXT1	0
COL_TEXT2	1
COL_TEXT3	2
COL_TEXT4	3
COL_TEXT5	4
COL_TEXT6	5

SPE adapter action command overview

This table summarizes the adapter action commands, syntax, and whether you can use the command with data sources, data targets, or both. You can specify the commands on the adapter command line or through map rules.

The commands are not case-sensitive, but the command values are case-sensitive. Choices are separated by the OR symbol | . Optional parameters are enclosed in brackets [].

The topics that follow include the short forms of the commands and keywords, and provide the command syntax in railroad track format.

Table 1. SPE adapter command summary

Command	Syntax	Use with Data Source (GET)	Use with Data Target (PUT)
-ADOC	-URL file_URL -BATCH file_URL[file_URL] -ADOC [data_from_map]	Yes	No
-GDOC	-DOCID doc_ID -GDOC	Yes	No
-RDOC	{-DOCID doc_ID -BATCH {doc_ID [;doc_ID]}} -RDOC	Yes	Yes
-UDOC	-DOCID doc_ID [-URL file_URL] -UDOC [data_from_map]	Yes	No
-ENVELOPE	-DBURL {db_URL db_cfg_file} [-DBUSER user_name] [-DBPSWD password] -DBDRIVER driver_name [-DBSCHEMA schema_name] {-URL file_URL -DOCID doc_ID -BATCH {doc_ID [;doc_ID file_URL [;file_URL]}} -SENDER sender_ID -RECEIVER receiver_ID -ALIAS acceptor_lookup_alias [-EDISTANDARD edi_standard] [-SENDERQUAL sender_qualifier] [-RECEIVERQUAL receiver_qualifier] [-DEFERRED -GETDEFERRED -INTERCHANGE interchange_ID]] -ENVELOPE [-KEEP] [-ALLDOCS] [data_from_map]	Yes	No
-DEENVELOPE	-DBURL {db_URL db_cfg_file} [-DBUSER user_name] [-DBPSWD password] -DBDRIVER driver_name [-DBSCHEMA schema_name] {-URL file_URL -DOCID doc_ID -BATCH {doc_ID [;doc_ID file_URL [;file_URL]}} -DEENVELOPE [-KEEP] [-ALLDOCS] [data_from_map]	Yes	No
-TRANSFORM	-DBURL {db_URL db_cfg_file} [-DBUSER user_name] [-DBPSWD password] -DBDRIVER driver_name [-DBSCHEMA schema_name] -MAP {map_name_in_database map_file_URL} [-URL file_URL -DOCID doc_ID] -TRANSFORM [-KEEP] [-ALLDOCS] [data_from_map]	Yes	No

- Command aliases**

Use the SPE adapter alias to use the Standards Processing Engine (SPE) adapter on input and output cards and in **GET** and **PUT** map rules.

- The SPE database configuration file**

The required **-DBURL** parameter of the **-ENVELOPE**, **-DEENVELOPE**, and **-TRANSFORM** commands specifies either the URL of the Standards Processing Engine (SPE) database or the file URL of a database configuration file. The configuration file must specify the database connection URL and driver. If the database requires them, the configuration file also must specify the database user name, password, and schema.

- **[Database connection persistence](#)**
Database connection information remains in effect for the duration of a map. After you configure the database URL, driver, user name, password, and schema on a map rule or in the configuration file, you can omit them from subsequent rules in the same map. The database connection values remain in effect until the map completes or another map rule specifies a different configuration file or keyword value.
- **[The data_from_map option](#)**
Some of the Standards Processing Engine (SPE) adapter action commands that follow include the data_from_map option. The data_from_map option represents the data that is passed as the third parameter of a **GET** function. It is not a parameter on the adapter command line.
- **[Add document \(-ADOC\) command](#)**
The **-ADOC** command adds one or more files on the file system as a document in memory and returns the document ID to the map. Alternatively, the **-ADOC** command can add dynamic data from a map as a document in memory and return the document ID to the map.
- **[Get document data \(-GDOC\) command](#)**
The **-GDOC** command returns data from a document in memory.
- **[Remove document \(-RDOC\) command](#)**
The **-RDOC** command removes one or more documents from memory. By default, documents are removed from memory when map processing completes. Use this command if too many intermediate documents accumulate in memory during map processing.
- **[Replace document \(-UDOC\) command](#)**
The **-UDOC** command replaces an existing document in memory with a new one and returns the new document ID.
- **[Run a map \(-TRANSFORM\) command](#)**
The **-TRANSFORM (-TFM)** command runs a map on the Standards Processing Engine (SPE) and returns a response from SPE in XML format. The **-TRANSFORM** command can run a IBM Sterling Transformation Extender map, a Sterling B2B Integrator map, or an XSLT map.
- **[De-envelope a document \(-DEENVELOPE\) command](#)**
The **-DEENVELOPE (-DENV)** command de-envelopes a document as specified in the Standards Processing Engine (SPE) database and returns a response from SPE in XML format.
- **[Envelope a document \(-ENVELOPE\) command](#)**
The **-ENVELOPE (-ENV)** command envelopes a document as specified in the Standards Processing Engine (SPE) database and returns a response from SPE in XML format.

Command aliases

Use the SPE adapter alias to use the Standards Processing Engine (SPE) adapter on input and output cards and in **GET** and **PUT** map rules.

Specify adapter commands by using a command string on the command line or by creating a command file that contains adapter commands. The command syntax is:

```
-IA[alias]card_num
-OA[alias]card_num
```

where :

-IA
The Input Source Override execution command

-OA
The Output Target Override execution command

alias
The adapter alias

card_num
The number of the map card

To override an input card:

-IASPEcard_num

To override an output card:

-OASPEcard_num

The SPE database configuration file

The required **-DBURL** parameter of the **-ENVELOPE**, **-DEENVELOPE**, and **-TRANSFORM** commands specifies either the URL of the Standards Processing Engine (SPE) database or the file URL of a database configuration file. The configuration file must specify the database connection URL and driver. If the database requires them, the configuration file also must specify the database user name, password, and schema.

When the **-DBURL** keyword specifies the configuration file URL, the SPE adapter ignores any driver, user name, password, or schema that is specified on the adapter command line.

The format of the configuration file is:

```
javax.persistence.jdbc.url=db_URL
javax.persistence.jdbc.user=user_name
javax.persistence.jdbc.password=password
javax.persistence.jdbc.driver=driver_name
openjpa.jdbc.Schema=schema_name
```

You can specify an alias on the **-DBURL** keyword to support multiple configuration files, for example **-DBURL %dbcfgfile%**. You also can create aliases for the other keyword values in the adapter command line. See the Resource Registry documentation for details about resource aliases.

Database connection persistence

Database connection information remains in effect for the duration of a map. After you configure the database URL, driver, user name, password, and schema on a map rule or in the configuration file, you can omit them from subsequent rules in the same map. The database connection values remain in effect until the map completes or another map rule specifies a different configuration file or keyword value.

The data_from_map option

Some of the Standards Processing Engine (SPE) adapter action commands that follow include the data_from_map option. The data_from_map option represents the data that is passed as the third parameter of a **GET** function. It is not a parameter on the adapter command line.

Add document (-ADOC) command

The **-ADOC** command adds one or more files on the file system as a document in memory and returns the document ID to the map. Alternatively, the **-ADOC** command can add dynamic data from a map as a document in memory and return the document ID to the map.

-ADOC command syntax

```
>>-----+-- -ADOC-----><
  +- -URL--file_URL-----+
  |   .;-----.
  |   V-----.
  '--- -BATCH-----file_URL---'
    '- H-----'
```

-URL

Adds a single file, specified by a file URL, as a document in memory and returns a document ID to the map. The **-URL** keyword is optional, but you must specify one of the **-BATCH**, **-URL**, or **data_from_map** options on the **-ADOC** command.

-BATCH

-H

Adds multiple files, specified by a list of semi-colon-delimited (;) file URLs, to documents in memory and returns a list of semi-colon-delimited document IDs to the map. The **-BATCH** keyword is optional, but you must specify one of the **-BATCH**, **-URL**, or **data_from_map** options on the **-ADOC** command.

data_from_map

Adds dynamic data from a map as a document in memory and returns the document ID to the map. The **data_from_map** is optional, but you must specify one of the **-BATCH**, **-URL**, or **data_from_map** options on the **-ADOC** command.

Examples

The following example adds the parsewds.txt file as a document in memory and returns the document ID to the map:

```
GET("SPE", "-URL file:///C:/parsewds.txt -ADOC")
```

The following example adds the Hello World!! text string as a document in memory and returns the document ID to the map:

```
GET("SPE", "-ADOC", "Hello World!!")
```

Get document data (-GDOC) command

The **-GDOC** command returns data from a document in memory.

-GDOC command syntax

```
>>-- -DOCID---doc_ID-- -GDOC-----><
  '- -ID---'
```

-DOCID

-ID

Specifies the document ID of a document in memory from which data is returned. The **-DOCID** keyword is required.

Remove document (-RDOC) command

The **-RDOC** command removes one or more documents from memory. By default, documents are removed from memory when map processing completes. Use this command if too many intermediate documents accumulate in memory during map processing.

-RDOC command syntax

```
>>--- -DOCID---doc_ID-----+ -RDOC-----><
  | '- -ID---'
```

```
| .-----.
| V | |
'-- -BATCH-----doc_ID---'
`- H-----'
```

-DOCID

-ID

Removes a single file, specified by a document ID, from memory. You must specify either the -DOCID keyword or the -BATCH keyword on the **-RDOC** command.

-BATCH

-H

Removes multiple files, specified by a list of semi-colon-delimited (;) document IDs, from memory. You must specify either the -BATCH keyword or the -DOCID keyword on the **-RDOC** command.

Replace document (-UDOC) command

The **-UDOC** command replaces an existing document in memory with a new one and returns the new document ID.

-UDOC command syntax

```
>>-- -DOCID---doc_ID---+-----+
  '- ID----'      '- URL--file_URL--'

>-- -UDOC---+-----+-----><
  '-data_from_map-'
```

-DOCID

-ID

Specifies the ID of the document in memory that is to be replaced. The -DOCID keyword is required.

-URL

Specifies the file URL of the document that is to replace the document in memory. Either the -URL or the data_from_map option is required.

data_from_map

Replaces the document in memory with dynamic data from a map. Either the data_from_map or the -URL option is required.

Run a map (-TRANSFORM) command

The **-TRANSFORM (-TFM)** command runs a map on the Standards Processing Engine (SPE) and returns a response from SPE in XML format. The **-TRANSFORM** command can run a IBM Sterling Transformation Extender map, a Sterling B2B Integrator map, or an XSLT map.

The XML response file contains information about the documents after SPE processing. The metadata schema of the XML response is defined by the wtx_install_dir\Response.xsd file.

A Sterling B2B Integrator map or an XSLT map has a single input card and a single output card. The document data from a file, from memory, or from a map overrides the input card. The **-TRANSFORM** command overrides the output card with the final document after the transformation completes.

IBM Sterling Transformation Extender maps can have multiple input cards and multiple output cards. The document data from a file, from memory, or from a map overrides the first input card. The **-TRANSFORM** command overrides the last output card of a IBM Sterling Transformation Extender map and returns the final document after the transformation completes.

A IBM Sterling Transformation Extender map, XSLT map, or Sterling B2B Integrator map can run from the local file system or from the SPE database.

-TRANSFORM command syntax

```
>>-- -DBURL---+---+db_URL---+-----+
  '- DURL--'  '-db_cfg_file-'  +- -DBUSER--user_name-
  '- U--user_name----'

>-----+--- -DBDRIVER--driver_name-----+
  +- -DBPWD--password+
  '- P--password----'

>-----+-----+
  '- DBSCHEMA--schema_name-' 

>---+---+---+---+---+MAP---+map_name_in_database---+
  '- M---'  '-map_file_URL----'

>-----+---+---+TRANSFORM---+---+---+
  +- -URL--file_URL----+  '- -TFM-----'  '- -KEEP-
  '- -DOCID---doc_ID-
  '- -ID----'

>-----+---+---+---+---+---+ALLDOCS---+  '- data_from_map-' <

-DBURL
-DURL
```

Identifies the SPE database by either the database URL or the file URL of a configuration file that contains the database connection information. If you specify a configuration file, the SPE adapter uses only the database connection information in the configuration file. The adapter ignores any -DBDRIVER, -DBUSER, -DBPSWD, or -DBSCHEMA keywords on the adapter command line. The -DBURL keyword is required.

-DBUSER

-U

Specifies the SPE database user name. The -DBUSER keyword is required only for certain types of databases.

-DBPSWD

-P

Specifies the SPE database password. The keyword -DBPSWD is required only for certain types of databases.

-DBDRIVER

Specifies the name of the JDBC driver that connects to the SPE database. The -DBDRIVER keyword is required for all database types.

-DBSCHEMA

Specifies the name of the SPE database schema. The -DBSCHEMA keyword is required only for certain types of databases.

-MAP

-M

Specifies the name of the map in the SPE database or, for IBM Sterling Transformation Extender maps only, the file URL of the map on the local file system. The -MAP keyword is required.

-URL

Specifies the URL of a file that is to override the first input card of the map. The -URL keyword is optional, but you must specify one of the -URL, -DOCID, or data_from_map options.

-DOCID

-ID

Specifies the document ID of the document in memory that overrides the first input card of the map. The adapter passes the document data to the -TRANSFORM command as the PrimaryDocument. The -DOCID keyword is optional, but you must specify one of the -URL, -DOCID, or data_from_map options.

-KEEP

Retains the documents that are returned in the SPE response by storing the documents in the local file system with the document ID as the file name. The -KEEP keyword is optional. If you omit it, the SPE adapter does not save documents after the map or card transaction completes.

-ALLDOCS

Specifies that SPE is to return all documents in the XML response file, including type=INTERMEDIATE documents. The -ALLDOCS keyword is optional. If you omit it, SPE returns only type=FINAL documents in the XML response file.

data_from_map

Specifies the data bytes that are passed from a map that are to override the first input card of the map. The data_from_map is optional, but you must specify one of the -URL, -DOCID, or data_from_map options.

De-envelope a document (-DEENVELOPE) command

The **-DEENVELOPE (-DENV)** command de-envelopes a document as specified in the Standards Processing Engine (SPE) database and returns a response from SPE in XML format.

The XML response file contains the document IDs of the de-enveloped documents. The metadata schema of the XML response file is defined by the wtx_install_dir\Response.xsd file.

-DEENVELOPE command syntax

```
>>-- -DBURL---+---db_URL-----+-----+----->
  '- -DURL-'  '-db_cfg_file-'  +- -DBUSER--user_name+-
                                '- -U--user_name-----'

>-----+--- -DBDRIVER--driver_name----->
  +- -DBPSWD--password+
  '- -P--password-----'

>-----+----->
  '- -DBSCHEMA--schema_name-'

>--- -URL--file_URL-----+---+ -DEENVELOPE----->
  +- -DOCID---+---doc_ID-----+  '- -DENV-----'
  |  '- -ID----'  |
  |          .;-----.
  |          V      |
  |  '- -BATCH---+---file_URL---+'
  |  '- -H-----' | .;-----.
  |          V      |
  |  '--doc_ID---'

>-----+-----+-----+-----><
  '- -KEEP-'  '- -ALLDOCS-'  '-data_from_map-'
```

-DBURL

-DURL

Identifies the SPE database by either the database URL or the file URL of a configuration file that contains the database connection information. If you specify a configuration file, the SPE adapter uses only the database connection information in the configuration file. The adapter ignores any -DBDRIVER, -DBUSER, -DBPSWD, or -DBSCHEMA keywords on the adapter command line. The -DBURL keyword is required.

-DBUSER

-U

Specifies the SPE database user name. The -DBUSER keyword is required only for certain types of databases.

-DBPSWD

-P

Specifies the SPE database password. The -DBPSWD keyword is required only for certain types of databases.

-DBDRIVER

Specifies the name of the JDBC driver that connects to the SPE database. The -DBDRIVER keyword is required for all database types.

-DBSCHEMA

Specifies the name of the SPE database schema. The -DBSCHEMA keyword is required only for certain types of databases.

-URL

Specifies the file URL of the document to de-envelope. A relative path for the file URL resolves to the directory of the map that invokes the **-DEENVELOPE** command. The -URL keyword is optional, but you must specify one of the -URL, -DOCID, -BATCH or data_from_map options.

-DOCID

-ID

Specifies the document ID of a single document to de-envelope. The -DOCID keyword is optional, but you must specify one of the -URL, -DOCID, -BATCH or data_from_map options.

-BATCH

-H

Specifies a list of documents to de-envelope with semi-colon-delimited (;) file URLs or document IDs. The -BATCH keyword is optional, but you must specify one of the -URL, -DOCID, -BATCH or data_from_map options.

-KEEP

Retains the documents that are returned in the SPE response. The SPE adapter stores the documents in the local file system and uses the document ID as the file name. The -KEEP keyword is optional. If you omit it, the SPE adapter does not save documents after the map or card transaction completes.

-ALLDOCS

Specifies that SPE is to return all documents in the XML response file, including type=INTERMEDIATE documents. The -ALLDOCS keyword is optional. If you omit it, SPE returns only type=FINAL documents in the XML response file.

data_from_map

Specifies the data bytes to be de-enveloped. The data_from_map bytes are returned from the third parameter of a **GET** function in the map.

Envelope a document (-ENVELOPE) command

The **-ENVELOPE (-ENV)** command envelopes a document as specified in the Standards Processing Engine (SPE) database and returns a response from SPE in XML format.

The XML response file contains the document IDs of the enveloped documents. The metadata schema of the XML response is defined by the wtx_install_dir\Response.xsd file.

The **-ENVELOPE** command supports both deferred and immediate enveloping

-ENVELOPE command syntax

```
>>-- -DBURL---+--+db_URL-----+-----+----->
  '- -DURL--'  '-db_cfg_file-'  '+- -DBUSER--user_name-+
    '- -U--user_name-----'
```



```
>-----+--+ -DBDRIVER--driver_name----->
  +- -DBPSWD--password+
  '- -P--password----'
```



```
>-----+----->
  '- -DBSCHEMA--schema_name-'
```



```
>-----+--+ -SENDER---+---sender_ID--->
  +- -URL--file_URL-----+  '- -S-----'
  +-+ -DOCID---+---doc_ID-----+
  |  '- -ID----'          |
  |  .-.;-----.          |
  |  V                   |
  '-+ -BATCH---+---file_URL---+
  '- -H-----' | .-.;-----. |
  | V           | |
  '|---doc_ID---'|
```



```
>--- -RECEIVER---+---receiver_ID----->
  '- -R-----'
```



```
>--- -ALIAS---+---acceptor_lookup_alias----->
  '- -A-----'
```

```

>---+-----+-----+
++ -EDISTANDARD--edi_standard+
'-' -E--edi_standard-----'

>---+-----+-----+
++ -SENDERQUAL--sender_qualifier+
'-' -SNDQ--sender_qualifier-----'

>---+-----+-----+
++ -RECEIVERQUAL--receiver_qualifier+
'-' -RCVQ--receiver_qualifier-----'

>---+-----+-----+
'----+ -DEFERRED-----+'-----+
| '-' -DEF-----' | '+' -INTERCHANGE---+interchange_ID-
'----+ -GETDEFERRED--+ ' -I-----'
'-' -GETDEF-----'

>---+-----+-----+
'-' -ENV-----' '- -KEEP-' '- -ALLDOCS-'----->

>---+-----+-----><
' -data_from_map-'----->

```

-DBURL

Identifies the SPE database by either the database URL or the file URL of a configuration file that contains the database connection information. If you specify a configuration file, the SPE adapter uses only the database connection information in the configuration file. The adapter ignores any -DBDRIVER, -DBUSER, -DBPSWD, or -DBSCHEMA keywords on the adapter command line. The -DBURL keyword is required.

-DBUSER

-U

Specifies the SPE database user name. The -DBUSER keyword is required only for certain types of databases.

-DBPSWD

-P

Specifies the SPE database password. The -DBPSWD keyword is required only for certain types of databases.

-DBDRIVER

Specifies the name of the JDBC driver that connects to the SPE database. The -DBDRIVER keyword is required for all database types.

-DBSCHEMA

Specifies the name of the SPE database schema. The -DBSCHEMA keyword is required only for certain types of databases.

-URL

Specifies the file URL of the document to envelope. A relative path for the file URL resolves to the directory of the map that invokes the -ENVELOPE command. The -URL keyword is optional, but you must specify one of the -URL, -DOCID, -BATCH or data_from_map options.

-DOCID

-ID

Specifies the document ID of a single document to envelope. The -DOCID keyword is optional, but you must specify one of the -URL, -DOCID, -BATCH or data_from_map options.

-BATCH

-H

Identifies a group of documents to envelope by file URLs or document IDs. The -BATCH keyword is optional, but you must specify one of the -URL, -DOCID, -BATCH or data_from_map options.

-SENDER

-S

Specifies the document sender that is used to locate the envelope information in the SPE database. The -SENDER keyword is required.

-RECEIVER

-R

Specifies the document receiver that is used to locate the envelope information in the SPE database. The -RECEIVER keyword is required.

-ALIAS

-A

Specifies the acceptor look-up alias to locate the envelope information in the SPE database. The -ALIAS keyword is required.

-EDISTANDARD

-E

Specifies the EDI standard that is used to locate the envelope information in the SPE database. The -EDISTANDARD keyword is optional; however, you must specify the -EDISTANDARD keyword if you specify the -SENDERQUAL or -RECEIVERQUAL keywords.

-SENDERQUAL

-SNDQ

Specifies the document sender qualifier that is used to locate the envelope information in the SPE database. The -SENDERQUAL keyword is optional. If you specify the -SENDERQUAL keyword, you must specify the -EDISTANDARD keyword.

-RECEIVERQUAL

-RCVQ
Specifies the document receiver qualifier that is used to locate the envelope information in the SPE database. The -RECEIVERQUAL keyword is optional. If you specify the -RECEIVERQUAL keyword, you must specify the -EDISTANDARD keyword.

-DEFERRED

-DEF
Specifies deferred enveloping mode. The-DEFERRED keyword is optional. If you omit the keyword, enveloping is immediate. The -DEFERRED keyword and the -GETDEFERRED keyword are mutually exclusive.

-GETDEFERRED

-GETDEF
Gets the deferred documents from the SPE database to envelope them. The-GETDEFERRED keyword is optional. The -DEFERRED keyword and the -GETDEFERRED keyword are mutually exclusive.

-INTERCHANGE

-I
Specifies the customer-defined interchange identifier that is associated with the documents in the SPE database. You can use the interchange identifier later to select only the documents that were prepared with a particular identifier. The interchange identifier is optional with the -DEFERRED or the -GETDEFERRED keywords.

-KEEP
Retains the documents that are returned in the SPE response. The SPE adapter stores the documents in the local file system and uses the document ID as the file name. The -KEEP keyword is optional.

-ALLDOCS
Specifies that SPE is to return all documents in the XML response file, including type=INTERMEDIATE documents. The -ALLDOCS keyword is optional. If you omit it, SPE returns only type=FINAL documents in the XML response file.

data_from_map
Specifies the data bytes to be enveloped. The data_from_map bytes are returned from the third parameter of a **GET** function in the map. The data_from_map bytes are optional.

Archive (Tar) Adapter

This documentation introduces the Archive (Tar) adapter. You can use the adapter with a Command Server, Launcher, Software Development Kit, or map in a map rule.

- [Overview](#)
 - [System requirements](#)
 - [Command alias](#)
 - [Archive \(Tar\) Adapter commands](#)
 - [Syntax summary](#)
 - [Troubleshooting](#)
 - [Return codes and error messages](#)
-

Overview

Use the Archive (Tar) adapter to create, delete, update, add to, or extract from UNIX tape archive files. The Archive (Tar) adapter supports standard UNIX permissions, user ID attributes, and group ID attributes. This adapter is only for UNIX platforms and z/OS® UNIX Systems Services (USS). To archive files on Windows platforms, use the Archive (Zip) Adapter.

Converting data to ASCII on z/OS

The Archive (Tar) adapter cannot be used to convert your data to ASCII format when you are running your maps on a z/OS operating system on an IBM® z/Architecture® platform. To convert your data to ASCII format, you must use your map, along with the appropriate rules, to perform the conversion.

System requirements

The minimum system requirements and operating system requirements for the Archive (Tar) adapter are detailed in the release notes. It is assumed that a Command Server has already been installed on the computer where the adapter is to be installed for run-time purposes.

Command alias

Adapter commands can be specified by using a command string on the command line or by creating a command file that contains adapter commands. The execution command syntax is:

-IA[alias] card_num

-OA[alias] card_num

where -IA is the Input Source Override execution command and -OA is the Output Target Override execution command, alias is the adapter alias, and card_num is the number of the input or output card. The following table shows the adapter alias and its execution command.

Adapter	Alias	As Input	As Output
Archive (Tar)	TAR	-IATARcard_num	-OATARcard_num

Archive (Tar) Adapter commands

This is a description of the functions and use of the Archive (Tar) adapter commands and their options.

- [List of commands](#)

List of commands

The following table lists valid commands for the Archive (Tar) adapter, the command syntax, and whether the command is supported (✓) for use with data sources, targets, or both.

Command	Syntax	Source	Target
Tar File Archive (-ARCHIVE)	-ARCHIVE tarfile	✓	✓
Archive File (-FILE)	-FILE filename [-UNIQUE]	✓	✓
Group ID (-GID)	-GID gid_num	✓	✓
Tar File Member (-MEMBER)	-MEMBER filename [-D -X -A -C -DI -U]	✓	✓
File Permissions (-PERM)	-PERM perm_num	✓	✓
Trace (-T)	-T[E][+][full_path]	✓	✓
User ID (-UID)	-UID uid_num	✓	✓

You can append trace information to the existing trace file by using either the A or + option.

- [Tar File Archive \(-ARCHIVE\)](#)
- [Archive File \(-FILE\)](#)
- [Group ID \(-GID\)](#)
- [Tar File Member \(-MEMBER\)](#)
- [File Permissions \(-PERM\)](#)
- [Trace \(-T\)](#)
- [User ID \(-UID\)](#)

Tar File Archive (-ARCHIVE)

Use the Tar File Archive adapter command (-ARCHIVE) to specify which tar file will be accessed.

-ARCHIVE tarfile

Option

Description

tarfile

Required value. Specify the name of the tar file being used.

For example, to specify the **edi_in.tar** file to be accessed:

-ARCHIVE /vault/backup/edi_in.tar

Archive File (-FILE)

Use the Archive File adapter command (-FILE) to extract a member to a file or instruct the adapter to add the contents of a file on disk to the specified archive in the map directory. Data passed from the output card is ignored.

-FILE filename [-UNIQUE]

Option

Description

filename

Required value. Specify the name of the resulting archive file.

-UNIQUE

When extracting a member from the archive, this creates a file with a unique name to hold the data from the member.

When the Archive File adapter command (-FILE) is not specified, data passed to and from the adapter is through memory. For example, when extracting a member from an archive, the data is read into memory and then returned to the map. Similarly, when adding a member, the data is normally passed to the adapter in memory.

For example, to specify the **edi_08011999.dat** file as the resulting archive file:

```
-FILE edi_08011999.dat -UNIQUE
```

Group ID (-GID)

Use the Group ID adapter command (-GID) to set the group ID of the member to be archived or extracted.

```
-GID gid_num
```

Option

Description

gid_num

Specify a group ID to be archived or extracted.

When adding a member and -GID is not specified, the group ID is set to 0, the root, unless -FILE is specified, in which case the member inherits the GID of the file.

When extracting a member and -GID is not specified, but -FILE is specified, the group ID of the file to which the member is extracted is set to the same GID as the extracted member.

The Group ID adapter command (-GID) is only supported in environments that support group IDs.

Tar File Member (-MEMBER)

Use the Tar File Member adapter command (-MEMBER) to specify which tar file member is to be modified.

```
-MEMBER filename [-A|-C|-D|-X|-U]
```

Option

Description

filename

Required value. Specify the name of the member file.

-A

Add the member file specified by *filename* to the archive. This option is only valid for data targets.

-C

Create the new member file specified by *filename*. This option is only valid for data targets.

-D

Delete the member file specified by *filename* from the archive. This option is valid for either data sources or data targets.

-X

Extract the member file specified by *filename* from the archive. This option is only valid for data sources.

-U

Update the member file specified by *filename* in the archive. This option is only valid for data targets.

The Tar File Member adapter command (-MEMBER) also requires the use of the Tar File Archive adapter command (-ARCHIVE).

File Permissions (-PERM)

Use the File Permissions adapter command (-FILE) to set UNIX permissions on the file to be extracted or archived.

```
-PERM perm_num
```

Option

Description

perm_num

Specify the permission number. Standard UNIX octal codes are supported.

For example, to specify the UNIX permission number of **0777** on the file to be extracted or archived:

```
-PERM 0777
```

When adding a member through memory the default permissions of 0777 are used. When adding a member using the Archive File adapter command (-FILE), the permissions on the file are used.

When doing an extract, specify the -PERM command to set the permissions for the extracted file if the -FILE adapter command was used. If the -FILE adapter command is not specified, the -PERM command is ignored.

The File Permissions adapter command (-PERM) is only supported in environments that support permissions.

Trace (-T)

Use the Trace adapter command (-T) to produce a diagnostics file that contains detailed information about Archive (Tar) adapter activity.

The default filename is **m4tar.mtr** or **map_name.mtr**, depending on the options you select, and the file is created in the map directory unless otherwise specified.

-T[E] [+] [*full_path*]

Option

Description

E

Produce a trace file containing only the adapter errors that occurred during map execution. The adapter trace file is produced by default in the map directory, using the full name of the map file with an **.mtr** filename extension (**map_name.mtr**).

+

Append trace information to the existing trace file by using either the A or + option.

full_path

Creates a trace file with the specified name in the specified directory.

You can override the adapter command line trace options dynamically using the Management Console. See "Dynamic Adapter Tracing" in the *Launcher* documentation for detailed information.

User ID (-UID)

Use the User ID adapter command (-UID) to set the user ID of the member to be archived or extracted.

-UID uid_num

Option

Description

uid_num

Specify a user ID to be archived or extracted.

When adding a member and -UID is not specified, the user ID is set to 0, the root, unless -FILE is specified, in which case the member inherits the UID of the file.

When extracting a member and -UID is not specified, but -FILE is specified, the user ID of the file to which the member is extracted is set to the same UID as the extracted member.

The User ID adapter command (-UID) is only supported in environments that support IDs.

Syntax summary

This documentation discusses the Archive (Tar) syntax summary and how it is used.

- [Data sources](#)
- [Data targets](#)
- [Examples](#)

Data sources

The following is the command syntax of the Archive (Tar) adapter commands used for data sources:

```
-ARCHIVE tarfile -MEMBER filename [-D|-X]
[-FILE filename [-UNIQUE]]
[-PERM perm_num]
[-GID gid_num]
[-UID uid_num]
[-T[E] [+]] [full_path]]
```

Data targets

The following is the command syntax of the Archive (Tar) adapter commands used for data targets:

```
-ARCHIVE tarfile -MEMBER filename [-A|-C|-D|-U]
[-FILE filename [-UNIQUE]]
[-PERM perm_num]
[-GID gid_num]
[-UID uid_num]
[-T[E] [+]] [full_path]]
```

Examples

For the GET...Source setting in an input card, select Archive (Tar). In the GET...Source...Command field, enter:

```
-ARCHIVE in.tar -MEMBER data.txt -X -TA
```

This command extracts a member file named **data.txt** from a tar file named **in.tar** and appends the trace file.

To override output map card 1 using the Archive (Tar) adapter, add a file named **new.txt** to the **in.tar** tar file with permissions of 0777, enter:

```
-OATAR1 -ARCHIVE in.tar -MEMBER new.txt -A -PERM 0777
```

Troubleshooting

For information about error codes and messages returned by the adapters, see [Return codes and messages](#).

- [Adapter trace file](#)

Adapter trace file

The adapter trace file contains detailed information provided by the adapter, including recorded information about the actions that are taking place such as connections established and statements executed. The trace file is produced during the adapter execution and can be used as a debugging aid.

For information about -T, see [Archive \(Tar\) Commands](#).

If adapter tracing is enabled, but the trace file is not created, any of the following could be indicated:

- The path to the Archive (Tar) adapter is incorrect
- The adapter is not located on your path
- The Trace adapter command name is misspelled
- The case for the Trace adapter command is incorrect

If the trace file is created, the shared object has been found and is being loaded. If this is true, there might be an invalid adapter command line or a problem with the tar file.

- [Sample adapter trace file](#)

Sample adapter trace file

```
==== M4TAR Adapter Log ====
Performing initialization for M4TAR_PutArchive
Command line in <-C -MEMBER output1 -ARCHIVE tarout.tar -T>
2 arguments detected
Archiving <output1>
Opened archive 'tarout.tar' successfully
blocksize = 32
a rwxrwxrwx 0/0    119 output1
a output1 1 blocks
PutArchive succeeded

==== End run. Elapsed time: 00:00:00 ====

```

Return codes and error messages

Return codes and error messages are returned when the particular activity completes. Return codes and error messages might also be recorded as specified in the audit logs, trace files and execution summary files.

- [Messages](#)

Messages

This list contains all the codes and error messages that can be returned as a result of using Archive (Tar) adapter for sources or targets.

Adapter return codes with positive numbers are warning codes that indicate a successful operation. Adapter return codes with negative numbers are error codes that indicate a failed operation.

Return Code	Message
0	GetArchive succeeded
0	

```

PutArchive succeeded
-1   Can't open the file %s
-1   Unable to allocate %d bytes
-1   '%s' does not exist
-1   '%s' name too long
-1   '%s' not a file
-1   '%s' not found
-1   '%s' seek error
-1   '%s' unknown file type `%c'
-1   Data not available
-1   Failed to open archive (%s)
-1   Insufficient memory: allocation of %d bytes failed
-1   Insufficient memory: allocation of %d bytes failed
-1   Invalid command line - missing data file name
-1   Invalid command line - missing entry name
-1   Invalid option (-a, -c, or -u) request for M4TAR_GetArchive
-1   Member ` %s not found in archive
-1   Out of memory
-1   Unable to allocate memory (%Id bytes)
-1   Unable to create directory %s
-1   Unable to read archive
-1   Archive read error
-1   Bad directory structure
-1   Can't create ` %s'
-1   Can't delete ` %s'
-1   Can't delete scratch file
-1   Can't link ` %s' - data stay in ` %s'
-1   Can't open directory ` %s'
-1   Can't append ` %s'
-1   Delete option (-d) is for file archives only
-1   Directory checksum error
-1   Error extracting ` %s'
-1   Error reading ` %s'
-1   Tape blocksize error
-1   Tape close error
-1   Tape read error
-1   Tape read failed
-1   Tape seek failed
-1   Tape write error

```

The following is a listing of all the codes and messages that can be returned as a result of using the Archive (Tar) adapter on RS/6000® AIX® only:

Return Code
Message
-2 Invalid Entry Point. Must specify M4TAR_GetArchive or M4TAR_PutArchive.

TIBCO Rendezvous Adapter overview

The TIBCO Rendezvous Adapter facilitates mapping of data to and from the TIBCO Rendezvous product, as well as launching maps based upon a message arrival from a TIBCO Rendezvous messaging system.

The TIBCO Active Enterprise Release 7 or later, including Rendezvous, provides a substantial improvement in transport stability and efficiency as compared to previous releases. A main feature is that the Rendezvous sub-system is completely thread-aware, whereas, in previous releases the system was thread-safe, but not thread-aware. This necessitated adapter dependence upon an existing operating system transport such as TCP or pipes to implement thread-safe behavior. This restriction has been removed in the latest release.

Several central features of earlier releases have been made obsolete in TIBCO Rendezvous Release 7 or later: sessions, event manager contexts, signal events, and unitary message (that is, a message containing a single datum and no fields).

In earlier releases, the session was a key architectural element, encapsulating a TIBCO Rendezvous daemon connection, an event manager context, an event queue, and an event dispatcher-all in a single object. Release 6 separates these roles for greater flexibility and ease of use:

- Programs (the adapter) create transport objects to send outbound messages and receive inbound messages. Some types of transport connect to the TIBCO Rendezvous daemon; others do not.
- Programs (the adapter) create event queues for inbound messages and other events.
- The TIBCO Rendezvous event driver places events on the correct event queues.
- Programs (the adapter) dispatch events from their event queues.

When a program from an earlier release sends a unitary message and a program using Release 6 or later receives it, the receiver reformats the inbound message, placing the single datum in a field named `_data_` with no field identifier.

The subject of a message defines a mailbox name for which a message listener will accept and process received messages. To receive messages, a TIBCO Rendezvous listener must be registered to receive a subject of interest. The subject name might consist of a single element subject (`mySubject`) or hierarchical multiple element subjects (`mySubject.myData.myName`). Wildcards are supported for use with the Subject Name adapter command (-SBN). Subject names are, at most, 255 characters in length. The maximum element length is 252 characters.

The TIBCO Rendezvous Adapter supports both regular and certified message delivery based upon the specified adapter commands and the capability of the installed TIBCO Rendezvous product.

- ["TIBCO Rendezvous"](#)
- [System Requirements](#)
- [Distributed queuing](#)
- [TIBCO Rendezvous Command Aliases](#)
- [TIBCO Rendezvous Adapter commands](#)
- [Syntax summaries](#)
- [Troubleshooting](#)
- [Return Codes and Error Messages](#)

System Requirements

The minimum system requirements and operating system requirements for the TIBCO Rendezvous Adapter are detailed in the release notes. In addition to these, the following are additional requirements to install and run the TIBCO Rendezvous Adapter:

- TIBCO Rendezvous software must be installed on the same machine that is running the Command Server.
- If a remote Rendezvous daemon is not used, the workstation upon which the TIBCO Rendezvous Adapter and the Command Server are installed must have the TIBCO Rendezvous software installed and activated (rvd daemon running).
- If a remote Rendezvous daemon is used (using the `-DAEMON` adapter command), the TIBCO Rendezvous software must be installed, although the local Rendezvous daemon need not be activated.
- [TIBCO Rendezvous Message Content](#)
- [Data format for TIBCO Rendezvous messages](#)
- [TIBCO Rendezvous](#)

TIBCO Rendezvous Message Content

TIBCO Rendezvous messages are self-describing messages unless the Opaque adapter command (`-OPAQUE`) is specified.

A generic message contains an optional header and a message body. The header might contain the reply subject for messages as a data target sent to a certified receiver and a subject name.

The message body follows the optional header. The message body consists of either a single data object corresponding to a single, self-described, simple RV message or an RVMsg that implies a composite, complex RV message.

The RVMsg might contain multiple message content(s), each of which is either a single datum or a nested RVMsg. Each of these items has a corresponding XML tag that is automatically put in place because of the type tree initiator and terminator definitions. All messages start with the `<Message>` tag and terminate with a `</Message>` tag.

- [Simple Messages](#)
- [Complex messages](#)

Simple Messages

An example of a simple message consisting of the `Hello` string with a null field name has an internal adapter representation of the following:

```
<Message>
<Body.Datum><Tuple.Item><Name></Name><String>Hello
</String></Tuple.Item></Body.Datum>
</Message>
```

Complex messages

The following three data objects are included in complex message example:

Field Name	Data Type	Data
INT_VAL	4-byte integer	****
FLOAT_VAL	4-byte float	****
STRING_VAL	String	String Data

This complex message data has an internal adapter representation of the following:

```
<Message>
<Body.RVMsg>
<Content.Datum><Tuple.Item><Name>INT_VAL</Name><I32>****</I32></Tuple.Item></
Content.Datum>
<Content.Datum><Tuple.Item><Name>
</Name><F32>*****</F32></Tuple.Item>
</Content.Datum>
<Content.Datum><Tuple.Item><Name>STRING_VAL</Name>
<String>String Data</String></Tuple.Item>
</Content.Datum>
</Body.RVMsg>
</Message>
```

Data format for TIBCO Rendezvous messages

TIBCO Rendezvous Release 7 or later has implemented an expanded set of wire format data types that includes array types. This change is reflected in the new type tree template for the adapter: `tibco_rv.mtt` in the `install_dir\examples\adapters\tibrv` example directory. The new type tree includes the addition of new types as described in the following sections:

- [Binary.Datum Item Types](#)
- [Tuple Array of Binary.Datum Item Types](#)
- [Binary.Datum item types](#)
- [Tuple Array of Binary.Datum Item Types](#)

Binary Datum item types

For the new data types for the single datum fields, as well as their pre-Release 7 analogs, see the following table:

Note: The datatype `RVMMSG_ENCRYPTED` from previous releases of TIBCO is obsolete for Release 7 or later. With regards to backward compatibility, inbound messages from older programs containing fields of this type will produce an error when the receiving program attempts to extract them from the message.

Type Tree Field	TIBCO Wire Format Type Name	Storage	Type Tree Field	TIBCO Pre-7 Wire Format Type Name
I8	TIBRVMMSG_I8	8-bit signed integer	INT1	RVMMSG_INT
I16	TIBRVMMSG_I16	16-bit signed integer	INT2	RVMMSG_INT
I32	TIBRVMMSG_I32	32-bit signed integer	INT4	RVMMSG_INT
I64	TIBRVMMSG_I64	64-bit signed integer	n/a	n/a
U8	TIBRVMMSG_U8	8-bit unsigned integer	UINT1	RVMMSG_UINT
U16	TIBRVMMSG_U16	16-bit unsigned integer	UINT2	RVMMSG_UINT
U32	TIBRVMMSG_U32	32-bit unsigned integer	UINT4	RVMMSG_UINT
U64	TIBRVMMSG_U64	64-bit unsigned integer	n/a	n/a
F32	TIBRVMMSG_F32	32-bit floating point	REAL4	RVMMSG_REAL
IPPort16	TIBRVMMSG_IPPORT16	2-byte IP port	IPData2	RVMMSG_IPDATA
IPAddr32	TIBRVMMSG_IPADDR32	4-byte IP address	IPData4	RVMMSG_IPDATA

Note: Corresponding data types for all TIBCO Rendezvous message types, as well as the type tree files and replies for non-certified messages and composite messages, are provided with the TIBCO Rendezvous Adapter.

Tuple Array of Binary Datum Item Types

See the following table for the new types for the tuple fields that are all sized datums.

Type Tree Field	TIBCO Wire Format Type Name	Storage	TIBCO Pre-7 Wire Format Type Name
Tuple.I8Array	TIBRVMMSG_I8ARRAY	8-bit integer array	n/a
Tuple.I16Array	TIBRVMMSG_I16ARRAY	16-bit integer array	n/a
Tuple.I32Array	TIBRVMMSG_I32ARRAY	32-bit integer array	n/a
Tuple.I64Array	TIBRVMMSG_I64ARRAY	64-bit integer array	n/a
Tuple.U8Array	TIBRVMMSG_U8ARRAY	8-bit unsigned integer array	n/a
Tuple.U16Array	TIBRVMMSG_U16ARRAY	16-bit unsigned integer array	n/a
Tuple.U32Array	TIBRVMMSG_U32ARRAY	32-bit unsigned integer array	n/a
Tuple.U64Array	TIBRVMMSG_U64ARRAY	64-bit unsigned integer array	n/a
Tuple.F32Array	TIBRVMMSG_F32ARRAY	32-bit floating point array	n/a
Tuple.F64Array	TIBRVMMSG_F64ARRAY	64-bit floating point array	n/a

TIBCO Rendezvous

The TIBCO Rendezvous messaging system uses point-to-point and broadcast communications of self-describing messages based upon the message subject. The Rendezvous messaging system is a reliable, non-certified delivery system.

The TIBCO Rendezvous software retains outbound messages for a duration of 60 seconds, after which the message is destroyed. Therefore, TIBCO Rendezvous maintains no significant data latency. If a message listener is not registered or enabled when a message arrives, the message is lost.

- [High Data Rates](#)

High Data Rates

TIBCO strongly recommends that you not use certified message delivery in situations that require high data rates. In comparison with base-level reliable delivery, certified delivery requires additional processing time, exchanges more control messages and consumes more process memory (and optional disk storage as well). Usage of these resources grows in proportion to the following:

- the number of certified messages sent
- the size of the data in these certified messages
- the number of listeners that receive certified delivery

Therefore, for optimal performance, TIBCO strongly recommends limiting the use of certified delivery.

Distributed queuing

Distributed queuing combines features of Rendezvous certified messaging and fault-tolerance functionality to provide load balancing and fault-tolerant process redundancy in the TIBCO Active Enterprise environment. Specifically, the TIBCO Rendezvous Adapter can use distributed queues to implement one-of- n certified delivery in which there are n listeners defined in a distributed queue. For a description of distributed queue features and guidance in the selection of distributed queue configuration parameters, see the *TIBCO Rendezvous Concepts* documentation.

Each distributed queue member session has two distinct roles: as a listener member and as a potential scheduler. In the listener member role, queue member sessions listen for task messages and process inbound task messages as assigned by the scheduler. TIBCO Rendezvous fault tolerance software maintains one active scheduler in each queue. If the scheduler process terminates, another member assumes the role of scheduler. The queue member session in the scheduler role assigns inbound tasks to listener members in the queue.

The member sessions of a distributed queue share the same reusable correspondent name indicating that they are members of that queue. This correspondent name is specified with the XXXXDISTRIBUTEDQUEUECH3 adapter command (-DQ). Each member session of a distributed queue must listen for messages with the same subject. Even when multiple members listen, only one member processes the message for each inbound message (or task).

You can control distributed queue attributes by using the Distributed Queue adapter command (-DQ) and the Queue Options adapter command (-QO).

To implement multiple distributed queue listeners, create multiple instances of map components that contain the Distributed Queue adapter command (-DQ), specifying the same distributed queue name and the same subject name using the Subject Name adapter command (-SBN). The mapping functionality should be identical across the maps.

Distributed queues do not use ledger files and distributed queue member sessions automatically require old messages from certified senders.

TIBCO Rendezvous Command Aliases

Adapter commands can be specified by using an execution command string on the command line or by creating a command file that contains adapter commands dictating the desired execution settings. For information about all of the options you can use with the execution commands for adapters or how to create a command file, see the *Execution Commands documentation*.

Use the Input Source Override - Message execution command (`-IM`) and the Output Target Override - Message execution command (`-OM`) with the TIBCO Rendezvous Adapter alias. The execution command syntax is:

```
-IM[alias] card_num  
-OM[alias] card_num
```

where `-IM` is the Input Source - Override execution command and `-OM` is the Output Target - Override execution command, `alias` is the adapter alias, and `card_num` is the number of the input or output card, respectively. The following table shows the adapter alias and execution commands.

Adapter	Alias	As Input	As Output
TIBCO Rendezvous	RV	<code>-IMRV</code> <i>card_num</i>	<code>-OMRV</code> <i>card_num</i>

When using an adapter alias with the execution command, the TIBCO Rendezvous Adapter commands can be issued on the command line or in a command file. You can use the adapter commands to specify the message subject name, the TIBCO Rendezvous service name, the timeout period, the certified message delivery name, and more.

For example, the command string for the adapter in the TIBCO Rendezvous environment might be:

```
-IMRV1 '-SBN rvcm.test.in -REPLY rvcm.reply -CMN output_CM  
-LFN rvcmrpout.lgr -T'
```

This example command line retrieves incoming messages with the subject name defined as `rvcm.test.in` using the Subject Name adapter command (`-SBN rvcm.test.in`). The Reply adapter command (`-REPLY rvcm.reply`) specifies the subject name of `rvcm.reply` to the confirmation of the message receipt (the reply). The adapter listens for messages with the certified session named `output_CM` using the required Certified Session Name adapter command (`-CMN input_CM`). The ledger file named `rvcmrpout.lgr` is specified as the ledger file to log the TIBCO Rendezvous certified delivery for this particular certified session using the Ledger File Name command (`-LFN rvcmrpout.lgr`). The Trace adapter command (`-T`) indicates that a trace file will be created to report adapter activity information, recording the events that occur while the adapter is retrieving this data.

- [Wildcard support](#)
- [GET ▶ Source Settings](#)
- [PUT ▶ Target Settings](#)

Wildcard support

The TIBCO Rendezvous messaging system supports two wildcard characters for the Subject Name adapter command (`-SBN`). This allows subscribers to listen for collections of related subjects. It is invalid to send messages to wildcard subjects.

- An asterisk (*) represents a single element wildcard in a subject name. The element position is context-sensitive. The subject name specified as `rv.*.in` finds `rv.data.in`, but not `rv.run.data.in`.
- The right angle bracket (>) represents a wildcard for all elements that follow the > in a subject name. The subject name specified as `rv.test.>` returns `rv.test.in.any`, `rv.test.out`, and `rv.test`, but not `rv`, or `test.rv`.
- [Wildcard usage with the Launcher](#)
- [Using wildcards to generate unique message IDs](#)

Wildcard usage with the Launcher

The Launcher supports wildcard matching and expansion only with the ">" wildcard character. If a triggering map contains wildcards, within this map, any data source and data target specified with a wildcard is expanded at run time, based upon the command line that the adapter returns to the Launcher at the time of the trigger.

To specify a TIBCO Rendezvous subject name that has a wildcard and to perform expansion when triggering a map in a system, do not use the release character. The expansion of the subject name wildcard is limited to the TIBCO Rendezvous semantics of the ">" wildcard character.

For example, assume that a map specifies the data source as messages from `rv1.>`. You want to return the message in a file with a name matching the wildcard portion of the subject, appended with `.txt`. The output file name is specified as `*.txt`. The command line for the data source is `-SBN rv1.*`. When messages from the `rv1.abc.xyz` subject are received, the trigger function expands the subject name and the output map card file name is set to `abc.xyz.txt`.

Using wildcards to generate unique message IDs

When triggering on messages, output files with a unique name can be generated based upon the messages received as input by invoking the Launcher wildcard facility within the adapter. For data sources, specify the wildcard Message ID adapter command `-MSG *` for the `GET>.Source>.Command` setting in the Map Designer or Integration Flow Designer. The adapter generates a unique message ID for each triggered message received as an input. This data is returned with the wildcard character * that is expanded to form the output subject name.

GET ▶ Source Settings

Use the Map Designer **GET** settings to configure options for input map cards using the TIBCO Rendezvous Adapter. For the `GET>.Source` setting, select TIBCO RV to use the TIBCO Rendezvous Adapter as the data source.

- [GET ▶ Source ▶ Command Setting](#)
- [Source ▶ Transaction ▶ OnSuccess Setting](#)

- [Source ▶ Transaction ▶ OnFailure Setting](#)
 - [Source ▶ Transaction ▶ Scope Setting](#)
 - [Source ▶ Transaction ▶ Warnings Setting](#)
-

GET ▶ Source ▶ Command Setting

Use the **Command** setting to enter adapter commands. For the syntax of the commands, see the [List of Commands](#).

Source ▶ Transaction ▶ OnSuccess Setting

This setting is ignored by the TIBCO Rendezvous Adapter. There is no transactional support for this publish-and-subscribe messaging system. (There is no message queue.)

To achieve message tracking, use the certified adapter commands. The TIBCO Rendezvous certified adapter commands include:

- Certified Session Name (-CMN)
 - Confirm (-CONFIRM)
 - Ledger File Name (-LFN)
 - Reply (-REPLY)
-

Source ▶ Transaction ▶ OnFailure Setting

This setting is ignored by the TIBCO Rendezvous Adapter. There is no transactional support for this publish-and-subscribe messaging system. (There is no message queue.)

To achieve message tracking, use the certified adapter commands. The TIBCO Rendezvous certified adapter commands include:

- Certified Session Name (-CMN)
 - Confirm (-CONFIRM)
 - Ledger File Name (-LFN)
 - Reply (-REPLY)
-

Source ▶ Transaction ▶ Scope Setting

Use the **Scope** setting to specify when to check for success (**OnSuccess**) or failure (**OnFailure**) of a map, card, or burst in order to perform the rollback and retry options. The default value is **Map**.

Value	Description
Map	Check for success or failure at the completion of each map. If the map completes successfully, use the OnSuccess setting. If the map fails, use the OnFailure setting.
Burst	Check for success or failure at the completion of each burst. If the burst is successful, use the OnSuccess setting. If the burst fails, use the OnFailure setting.
Card	Check for success or failure at the completion of each card. If the card is processed successfully, use the OnSuccess setting. If the card fails, use the OnFailure setting.

Source ▶ Transaction ▶ Warnings Setting

Use the **Warnings** setting to choose whether to fail or ignore warning conditions for the specific input map card. The default value is **Ignore**.

Value	Description
Ignore	Ignore warnings for this map card.
Fail	Fail the map upon receiving a warning for this map card.

PUT ▶ Target Settings

Use the Map Designer **PUT** settings to configure options for output map cards using the TIBCO Rendezvous Adapter. For the **PUT ▶ Target** setting, select TIBCO RV to use the TIBCO Rendezvous Adapter as the data target.

- [PUT ▶ Target ▶ Command Setting](#)

- [Target ▶ Transaction ▶ OnSuccess Setting](#)
- [Target ▶ Transaction ▶ OnFailure Setting](#)
- [Target ▶ Transaction ▶ Scope Setting](#)
- [Target ▶ Transaction ▶ Warnings Setting](#)

PUT ▶ Target ▶ Command Setting

Use the **Command** setting to enter adapter commands. For syntax of these commands, see the [List of Commands](#).

Target ▶ Transaction ▶ OnSuccess Setting

Use the **OnSuccess** setting with adapter targets to prevent a message from being created unnecessarily. The default value is **Create**.

Value	Description
Create	Upon successful completion of the burst, map, card, or rule (determined by the Scope settings), create the message.
CreateOnContent	Upon successful completion of the burst, map, card, or rule (determined by the Scope settings), create the message only if it is not an empty message.

Target ▶ Transaction ▶ OnFailure Setting

This setting is ignored by the TIBCO Rendezvous Adapter. There is no transactional support for this publish-and-subscribe messaging system. (There is no message queue.)

Target ▶ Transaction ▶ Scope Setting

Use the **Scope** setting to specify when to check for success (**OnSuccess**) or failure (**OnFailure**) of a map, burst, or card in order to perform the rollback and retry options. The default value is **Map**.

Value	Description
Map	Check for success or failure at the completion of each map. If the map completes successfully, use the OnSuccess setting. If the map fails, use the OnFailure setting.
Burst	Check for success or failure at the completion of each burst. If the burst is successful, use the OnSuccess setting. If the burst fails, use the OnFailure setting.
Card	Check for success or failure at the completion of each card. If the card is processed successfully, use the OnSuccess setting. If the card fails, use the OnFailure setting.

Target ▶ Transaction ▶ Warnings Setting

Use the **Warnings** setting to choose whether to fail or ignore warning conditions for the specific output map card. The default value is **Ignore**.

Value	Description
Ignore	Ignore warnings for this map card.
Fail	Fail the map upon receiving a warning for this map card.

TIBCO Rendezvous Adapter commands

The following table lists valid commands for the TIBCO Rendezvous Adapter, the adapter command syntax, and whether the adapter command is supported (✓) for use with data sources, data targets, or with both.

TIBCO RV Command Name	Syntax	Source	Target
Certified Session Name (-CMN)	-CMN [session_name]	✓	✓
Confirmation (-CONFIRM)	-CONFIRM	✓	
Daemon (-DAEMON)	-DAEMON full_path	✓	✓
Distributed Queue (-DQ)	-DQ QName[, ScdWt[, Hrtbt][, Actvn]]	✓	
Error Subject Name (-ERN)	-ERN subject_name	✓	✓

TIBCO RV Command Name	Syntax	Source	Target
Header (-HDR)	-HDR	✓	
Listener Certified Session Name (-LCMN)	-LCMN <i>session_name</i>		✓
Ledger File Name (-LFN)	-LFN [<i>ledger_name</i>]	✓	✓
Listen (-LSN)	-LSN {0 S <i>wait_time_in_sec</i> }	✓	
Message (-MSG)	-MSG *	✓	
Network (-NETWORK)	-NETWORK <i>name</i>	✓	✓
No Old Messages (-NOM)	-NOM	✓	
Opaque Message Format (-OPAQUE)	-OPAQUE	✓	✓
Point-to-Point (-PP)	-PP	✓	
Queue Options (-QO)	-QO <i>LstWt</i> [, <i>AccTm</i> [, <i>CplTm</i> [, <i>Tasks</i>]]	✓	
Quantity (-QTY)	-QTY { <i>value</i> S}	✓	
Reply (-REPLY)	-REPLY <i>reply_name</i>		✓
Request Listen (-RQSTLSN)	-RQSTLSN <i>request_lsn_time</i>		✓
Request Reply (-RQSTRPL)	-RQSTRPL		✓
Subject Name (-SBN)	-SBN <i>subject_name</i>	✓	✓
Service (-SERVICE)	-SERVICE <i>name</i>	✓	✓
Trace (-T)	-T[E] [<i>full_path</i>]	✓	✓

- [Certified Session Name \(-CMN\)](#)
- [Confirmation \(-CONFIRM\)](#)
- [Daemon \(-DAEMON\)](#)
- [Distributed Queue \(-DQ\)](#)
- [Error Subject Name \(-ERN\)](#)
- [Header \(-HDR\)](#)
- [Listener Certified Session Name \(-LCMN\)](#)
- [Ledger File Name \(-LFN\)](#)
- [Listen \(-LSN\)](#)
- [Message \(-MSG\)](#)
- [Network \(-NETWORK\)](#)
- [No Old Messages \(-NOM\)](#)
- [Opaque Message Format \(-OPAQUE\)](#)
- [Point-to-Point \(-PP\)](#)
- [Queue Options \(-QO\)](#)
- [Quantity \(-QTY\)](#)
- [Reply \(-REPLY\)](#)
- [Request Listen \(-RQSTLSN\)](#)
- [Request Reply \(-RQSTRPL\)](#)
- [Subject Name \(-SBN\)](#)
- [Service \(-SERVICE\)](#)
- [Trace \(-T\)](#)

Certified Session Name (-CMN)

The Certified Session Name adapter command (-CMN) allows the sender or receiver to specify the certified session name for data sources and data targets.

-CMN [session_name]

Option

Description

session_name

This is a reusable name under which to register the certified session.

For example, to specify the certified session name **output_CM**, the syntax would be:

-CMN output_CM

The certified session name is the unique label for identifying a certified session. (This is not the same as the subject name.) The certified session name must be unique for each map and card because the certified session name establishes a point-to-point communication link between the sender and receiver that cannot be shared.

If the Certified Session Name adapter command (-CMN) does not specify the name of the certified session, a unique, non-reusable session name is assigned to the certified sender/receiver endpoint.

Note: You are not allowed to specify a NULL-certified session name while requiring that the certified sender retain unacknowledged messages. Omitting the argument for the Certified Session Name adapter command (-CMN) (which is equivalent to specifying a NULL-certified session name) automatically enforces the No Old Messages adapter command (-NOM) requirement for certified data sources.

If the Confirm adapter command (-CONFIRM) is not specified using the Certified Session Name adapter command (-CMN), the Rendezvous daemon generates and sends an automatic confirmation of message receipt to the sender, regardless of successful map completion.

If the Confirm adapter command (-CONFIRM) is specified using the Certified Session Name adapter command (-CMN), the confirmation of message receipt is sent only upon successful completion of the map.

Note: Use the Certified Session Name adapter command (-CMN) to bind the session name to the transport using the `tibrvbcmTransport_Create()` function.

Confirmation (-CONFIRM)

The Confirm adapter command (**-CONFIRM**) is used only on data sources receiving certified messages to specify that confirmation is explicit (non-automatic).

-CONFIRM

After successful completion of the map, the adapter sends a confirmation to the sender that the message was successfully received. The Certified Session Name adapter command (**-CMN**) must also be specified. If the Confirm adapter command (**-CONFIRM**) is not specified, confirmation occurs automatically upon receipt of a message.

Daemon (-DAEMON)

The Daemon adapter command (**-DAEMON**) is used on data sources or data targets to specify the TIBCO Rendezvous daemon location.

-DAEMON full_path

Option

Description

full_path

This is the full path of the TIBCO Rendezvous daemon location.

For example, to specify the location of the daemon on the current system, the syntax would be:

```
-DAEMON tcp:7501
```

Distributed Queue (-DQ)

Use the Distributed Queue adapter command (**-DQ**) to assign a correspondent name identifying data sources as members of a distributed queue and to set scheduling parameters.

-DQ QName[, ScdWt[, Hrtbt][, Actvn]]

Option

Description

Qname

This is a reusable correspondent name that defines a distributed queue.

ScdWt

This is the scheduler weight, which is a value that determines the queue member selected as the scheduler. The member with the greatest scheduler weight takes precedence. Acceptable values range between 1 and 65535, where the higher number has the greater scheduler weight.

Hrtbt

This is the scheduler heartbeat, which is an interval (in milliseconds) that is used to send messages from the scheduler session to inform other members that a member is acting as the scheduler. Acceptable values are unsigned 32-bit integers (except zero).

Actvn

This is the scheduler activation, which is the length of time (in milliseconds) that the heartbeat signal from the scheduler must be silent before the queue member with the greatest scheduler weight takes its place as the new scheduler. Acceptable values are unsigned 32-bit integers (except zero).

The member sessions of a distributed queue share the same reusable correspondent name (*Qname*), indicating that they are members of that queue. Each member of a distributed queue must use the same *Qname*.

The scheduler weight (*ScdWt*) represents the ability of the current session to fulfill the role of scheduler relative to other members of the same queue. If *ScdWt* is not specified, the default scheduler weight is 1.

All sessions in the queue must specify the same scheduler heartbeat value (*Hrtbt*) in milliseconds. If *Hrtbt* is not specified, the default heartbeat is 1000. *ScdWt* is required before you can specify *Hrtbt*.

All sessions in the queue must specify the same scheduler activation (*Actvn*) value in milliseconds. If *Actvn* is not specified, the default scheduler activation is 3000. *ScdWt* and *Hrtbt* are required before you can specify *Actvn*.

Note: The Distributed Queue adapter command (**-DQ**) must be accompanied by the Subject Name adapter command (**-SBN**) to specify the message subject name. For example, to implement a distributed queue named **dq.local** that will listen for messages from the subject **rv.msg** and set the scheduler weight to 5, the scheduler heartbeat to 2000 (2 seconds), and the scheduler activation to 6000 (6 seconds), the syntax would be:

```
-SBN rv.msg -DQ dq.local, 5, 2000, 6000
```

Note: The Distributed Queue adapter command (**-DQ**) cannot be used with the Certified Session Name adapter command (**-CMN**). To set listener options for members of a distributed queue, see the [Queue Options \(-QQ\)](#) adapter command.

Error Subject Name (-ERN)

Use the Error Subject Name adapter command (**-ERN**) to specify the subject to which a message is re-posted in the event of a map failure.

-ERN subject_name

Option

Description
<i>subject_name</i>
This is the TIBCO Rendezvous subject name for error messages.
If the message is not capable of being re-sent, a warning message is published to the specified error subject name. The error subject name specified with this command cannot be shared across input and output map cards, although the same subject name might be used in multiple input cards or multiple output cards.
Re-posted error messages are non-certified (reliable) messages. The initial map failure creates an adapter-generated error message with the -ERN-specified subject name and the following content: Map failed. Message to follow.
The message is then re-sent. If that re-posting is unsuccessful for any reason, a second error message with the -ERN-specified subject name is created with the following content: Message resend failure.
Map cards are processed in sequence; therefore, an input or output card using the Error Subject Name adapter command (-ERN) should be the first map card defined. If the -ERN map card follows a map card that causes a map failure, the -ERN map card is not invoked to re-send the message.
Note: The Error Subject Name adapter command (-ERN) is valid only on the TIBCO Rendezvous messaging system.

Header (-HDR)

Use the Header adapter command (-HDR) for data sources to specify to the adapter that header information should be included in the returned data.

-HDR

When the TIBCO Rendezvous Adapter is invoked without the Header adapter command (-HDR), the data consists of only the body of the message. The header information includes the optional subject name and reply name that allows the subject name and reply name data to be mapped when the Opaque adapter command (-OPAQUE) is specified.

The Header adapter command (-HDR) is not required for data targets because the adapter detects the presence of header data.

If the message contains any header information, the header information overrides the Subject Name adapter command (-SBN) and the Reply Subject Name adapter command (-REPLY).

Listener Certified Session Name (-LCMN)

Use the Listener Certified Session Name adapter command (-LCMN) for data targets to specify the name of the receiving certified session.

-LCMN session_name

Option

Description
<i>session_name</i>
This is the name of the receiving certified session.
When the TIBCO Rendezvous Adapter is used as a data target that is intended to publish point-to-point certified messages to a participating certified message listening endpoint (subscriber), the Listener Certified Session Name adapter command (-LCMN) specifies the certified session name of the listening endpoint.
Specifying the certified session name of the listening endpoint with the Listener Certified Session Name adapter command (-LCMN) allows the certified receiver to begin receiving certified messages immediately from the adapter target as the messages are sent. Without this information, the receiver must rely upon the registration discovery exchange preceding the publication of the first message. It is possible that the name of the certified session used by the adapter will not be available to the certified receiver for the initial message.
The following is an example of a target command line that could be used to publish certified messages to the rvcmlistener listening endpoint subject having the RVCMLISTEN certified session name from the RVCMSENDER target certified session:

-SBN rvcmlistener -CMN RVCMSENDER -LCMN RVCMLISTEN

Note: The Certified Session Name adapter command (-CMN) must also be specified.

Ledger File Name (-LFN)

The Ledger File Name adapter command (-LFN) specifies for data sources and data targets the ledger file name for the logging of the TIBCO Rendezvous certified delivery for a particular certified session.

-LFN [ledger_name]

Option

Description
<i>ledger_name</i>
Specify the ledger file name for certified delivery TIBCO Rendezvous logging.

For example, to specify the **TIBCOout.fil** ledger file name, the syntax would be:

```
-LFN TIBCOout.fil
```

The ledger file name must be unique for each certified session name. The Certified Session Name adapter command (**-CMN**) must also be specified.

If the Ledger File Name adapter command (**-LFN**) does not specify the name of the ledger file or if the Certified Session Name adapter command (**-CMN**) is used without the Ledger File Name adapter command (**-LFN**), a process-based ledger for certified message tracking is automatically maintained.

The TIBCO Rendezvous certified message delivery stores messages in a ledger file until confirmations from registered certified receivers have been received (certified delivery). The ledger file is the permanent, disk-based, storage mechanism that TIBCO Rendezvous uses to ensure data retention for certified message delivery.

Listen (-LSN)

Use the Listen adapter command (**-LSN**) for data sources to indicate the length of time (in seconds) that the TIBCO Rendezvous Adapter waits for a message to be received. Map execution is suspended until the message is received or until the specified time lapses. The default value is one hour (3600 seconds).

```
-LSN {0|S|wait_time_in_sec}
```

Option

Description

0

Do not wait at all. If there is no input available, the adapter does not wait for more messages.

S

This is an infinite wait time.

wait_time_in_sec

This is the number of seconds that the adapter waits for a message.

For example, to specify a wait time of two minutes, the syntax would be:

```
-LSN 120
```

The maximum period of time available is 128 hrs (460800 seconds). The Listen adapter command (**-LSN**) is ignored when dealing with certified messages.

Message (-MSG)

The Message adapter command (**-MSG**) is used on data sources to instruct the adapter to return a unique message ID for each message triggered back to the Launcher.

```
-MSG *
```

For data sources, on the Integration Flow Designer Sources tab of the execution settings, use the **-MSG * message** adapter command. The incoming message data is returned with the * wildcard character expanded to form the unique output name.

The Message adapter command (**-MSG**) is a mechanism that allows the adapter to generate a unique name that corresponds to each TIBCO Rendezvous message. If the Message command (**-MSG ***) is defined with a data source, use * to designate a unique file or resource name. The * can then be used in subsequent cards to link the resources.

Network (-NETWORK)

The Network adapter command (**-NETWORK**) is used for data sources and data targets to specify the network name or address that the TIBCO Rendezvous software uses for all communications involving the current session.

```
-NETWORK name
```

Option

Description

name

This is the name or address of the network to use.

For example, to specify the network address of a server, the syntax would be:

```
-NETWORK 101.55
```

The adapter uses a single session for all TIBCO Rendezvous non-certified messages and a separate session for each TIBCO Rendezvous PRO certified subject.

No Old Messages (-NOM)

The No Old Messages adapter command (**-NOM**) is valid only for data sources and allows unacknowledged messages sent from a certified sender to be deleted from the ledger file regardless of whether confirmation of receipt was obtained.

-NOM

The default behavior when the No Old Messages adapter command (**-NOM**) is not used requires the associated certified sender to retain any unacknowledged messages in the ledger until the listener has confirmed receipt of the message.

Note: The Certified Session Name adapter command (**-CMN**) also must be specified. The No Old Messages adapter command (**-NOM**) is valid only on TIBCO Rendezvous PRO messaging systems that support certified message delivery.

Opaque Message Format (-OPAQUE)

The Opaque Message Format adapter command (**-OPAQUE**) is used for data sources and data targets to define the message data type as opaque, not self-describing.

-OPAQUE

The Opaque Message Format adapter command (**-OPAQUE**) informs the adapter to receive the input data and format only the raw message data as a blob (opaque) data type (not a self-describing message type). For data targets, the entire data is sent as a binary buffer of variable length (RVMSG_OPAQUE).

Point-to-Point (-PP)

The Point-To-Point adapter command (**-PP**) is used only on data sources. If this command is specified, a unique inbox subject name is created and one message is published on a subject specified using the Subject Name adapter command (**-SBN**) with its reply subject set to the inbox name.

-PP

For example, to implement point-to-point connection between a sending map (**Map1**) and a receiving map (**Map2**), see the following procedure.

To implement point-to-point connection

1. Select the **my.pp.connection** subject name upon which information about the unique inbox name will be exchanged.
2. Specify the following **GET > Source > Command** settings for the **Card1** input card of **Map1**:

```
-SBN my.pp.connection -HDR
```

3. Map the data object representing the header reply from the **Card1** input card of **Map1** to the data object representing the header subject in an output card for the **Card2** example.
4. Specify the following **GET > Source > Command** setting for the **Card1** input card of **Map2**:

```
-SBN my.pp.connection -PP
```

5. Run **Map1**.
6. Run **Map2**.

The adapter of the **Card1** input card of **Map2** will create a unique inbox name called **_INBOX.something** with a call to the **tibrvTransport_CreateInbox()** function. Then it will publish a message on the **my.pp.connection** subject with the reply subject of the message set to the **_INBOX.something** unique inbox subject name. It will then start to listen for messages on the **_INBOX.something** subject name.

Queue Options (-QO)

Use the Queue Options adapter command (**-QO**) to assign listener options for members of a distributed queue.

```
-QO LstWt[, AccTm[, CplTm[, Tasks]]]
```

Option

Description

LstWt

This is the listener weight, which is a value that determines the listener that receives a task from the scheduler. The member with the highest listener weight (largest numerical value) takes precedence.

AccTm

This is the listener accept time.

If non-zero, this is the length of time (in milliseconds) the scheduler waits for a listener member to accept an assigned task before reassigning the task.

If zero, there is no limit on the acceptance time. Therefore, the scheduler does not set a timer and does not reassign the task.

CplTm

This is the listener complete time.

If non-zero, this is the length of time (in milliseconds) the scheduler waits for a listener member to complete an assigned task before reassigning the task.

If zero, there is no limit on the completion time. Therefore, the scheduler does not set a timer and does not reassign the tasks.

Tasks

This is the maximum number of tasks a listener can accept.

When the scheduler receives a task, it assigns the task to the available listener with the greatest listener weight (*LstWt*). If *LstWt* is not defined, the default listener weight is 1.

If the accept time (*AccTm*) lapses before the scheduler receives acceptance from the listener member, the scheduler reassigns the task to another listener member. Zero is a special value (the default) that specifies no limit on the acceptance time. In this case, the scheduler does not set a timer and does not reassign tasks when the task has not been accepted. *AccTm* is specified in milliseconds and must be less than the complete time (unless complete time is zero). If *AccTm* is not specified, the default listener accept time is 0.

If complete time (*CplTm*) elapses before the scheduler is notified that the listener member has completed its task, the scheduler reassigns the task to another listener member. Zero is a special value (the default) that specifies no limit on the completion time, in which case the scheduler does not set a timer and does not reassign tasks when the task has not been completed. *CplTm* is specified in milliseconds and must be greater than the *AccTm* (unless complete time is zero). If *CplTm* is not specified, the default listener complete time is 0. *AccTm* is required before you can specify *CplTm*.

When the number of accepted tasks reaches the maximum set in listener tasks (*Tasks*), the listener cannot accept additional tasks until it completes one or more of them. When the scheduler receives a task, it assigns the task to the listener with the greatest *LstWt* unless the pending tasks assigned to that listener exceed its task capacity. When the preferred listener has too many tasks, the scheduler assigns the new inbound task to the listener with the next greatest *LstWt*. The value must be 1 or greater. If *Tasks* is not specified, the default listener tasks value is 1. *AccTm* and *CplTm* are required before you can specify *Tasks*.

Note: The Queue Options adapter command (-*QO*) must be accompanied by the Subject Name adapter command (-*SBN*) and the Distributed Queue adapter command (-*DQ*).

The following example demonstrates how to implement a distributed queue named **dq.local** that listens for messages from the **rv.msg** subject using the following specifications:

- The scheduler weight is 5.
- This scheduler heartbeat is 2000 (2 seconds).
- The scheduler activation is 6000 (6 seconds).
- The listener weight is 50.
- The accept time is 5000 (5 seconds).
- The complete time is 10000 (10 seconds).

```
-SBN rv.msg -DQ dq.local, 5, 2000 6000 -QO 50, 5000, 10000
```

For assigning a correspondent name identifying data sources as members of a distributed queue and for setting scheduling parameters, see the [Distributed Queue \(-DQ\)](#) adapter command.

Quantity (-QTY)

Use the Quantity adapter command (-*QTY*) to specify the number of messages to retrieve on the subject name as specified using the Subject Name adapter command (-*SBN*). For the TIBCO Rendezvous Adapter, this value always equals 1.

-QTY {value|S}

Option

Description

S

For TIBCO Rendezvous, this returns one message on the subject name.

value

This is a positive integer representing the number of concurrent multiple messages to be retrieved. The TIBCO Rendezvous Adapter overwrites this to equal 1.

The Quantity adapter command (-*QTY*) defines the maximum number of messages across all bursts for a map.

The number of bursts in a single map invocation can be controlled using the combination of values set in the Map Designer **SourceRule > FetchAs > FetchUnit** setting and the Quantity adapter command (-*QTY*).

Note: When using the Quantity adapter command (-*QTY*) with the **S** option through either the Launcher or the Command Server, the Listen adapter command (-*LSN*) is also required with a value specified other than **S**.

Reply (-REPLY)

Use the Reply adapter command (-*REPLY*) to specify the subject name to receive the confirmation message receipt (the reply). The Reply adapter command (-*REPLY*) is used on data targets only.

-REPLY reply_name

Option

Description

reply_name

This is the subject name of the reply message.

For example, to specify the **tixrecv** subject name for the data target messages, the syntax would be:

```
-SBN output_RCV -REPLY tixrecv
```

If a reply name is specified in the header information, the header information takes precedence over the Reply adapter command (-*REPLY*). The adapter assumes that the listener for the reply is operational and has established delivery tracking with the receiver of the output map message.

To implement a reply in a map receiving a message, use an input card to receive a message from a subject that has published a message using the Reply adapter command (-REPLY). Next, map the data object representing the header reply from an input card to the data object representing the header subject in an output card. Format the output to send the appropriate reply message. The presence of the subject in the header forces the adapter to publish the reply to the subject.

Request Listen (-RQSTLSN)

Use the Request Listen adapter command (-RQSTLSN) to set the maximum listen time (in seconds) to wait for the reply when the message is sent using the Request Reply adapter command (-RQSTRPL). If it is not specified, the **tibrvTransport_SendRequest** or the **tibrvcvTransport_SendRequest** function call will block until the receipt of the replied message.

-RQSTLSN requestlsntime

Option

Description
<i>requestlsntime</i>

This is the maximum listen time (in seconds) to wait for the reply when the message is sent using the Request Listen adapter command (-RQSTLSN).

Request Reply (-RQSTRPL)

Use the Request Reply adapter command (-RQSTRPL) to request to receive the confirmation message receipt (the reply). The Request Reply adapter command (-RQSTRPL) is used on data targets only.

-RQSTRPL

For example, to send the message to the **output_RCV** subject name with the request for confirmation for the message receipt, the syntax would be:

-SBN output_RCV -RQSTRPL

The adapter assumes that the listener for the reply is operational and has established delivery tracking with the receiver of the output map message. When the Request Reply adapter command (-RQSTRPL) is specified, the adapter invokes either the TIBCO Rendezvous **tibrvTransport_SendRequest** function call (without certified messaging) or the **tibrvcvTransport_SendRequest** (with certified messaging) function call. This function call blocks until either the receipt of the replied message or until the call times out, whichever is first to occur. The timeout period for which to wait until the reply can be specified using the [Request Listen \(-RQSTLSN\)](#) adapter command.

When the subscribing application (recipient) receives the message, the TIBCO Rendezvous callback function is activated and the subscribing application extracts the inbox reply subject name from the received message. This inbox subject name is automatically generated by the **tibrvTransport_SendRequest** or the **tibrvcvTransport_SendRequest** function call. The subscribing application must then format the reply message and send it to the subject.

To implement a reply in a map receiving a message, use an input card to receive a message from a subject that has published a message using the **tibrvTransport_SendRequest** function call. Next, map the data object representing the header reply from an input card to the data object representing the header subject in an output card. The presence of the subject in the header forces the adapter to publish the reply to the subject.

Subject Name (-SBN)

Use the Subject Name adapter command (-SBN) for data sources and data targets to specify the message subject name.

-SBN subject_name

Option

Description
<i>subject_name</i>

This is the TIBCO Rendezvous source or target message subject name. Wildcards are permitted.

For specific wildcard use rules, see "[Wildcard Support](#)". The usage of wildcards with the Subject Name adapter command (-SBN) allows subscribers to listen to collections of related subjects. No other checking is performed to confirm TIBCO Rendezvous syntax compliance.

For example, to specify the **output_RCV** subject name for the data target messages, the syntax would be:

-SBN output_RCV

To specify a subject name using the wildcard right angle bracket (>) to match all subjects beginning with **recv** and containing any characters that follow the > symbol, use the syntax:

-SBN recv>

For data sources, this is the subject name of the incoming message. For data targets, this is the subject name of the message to which to subscribe as a listener for message receipt.

If the message header contains any information, the header information overrides the Subject Name adapter command (-SBN).

Note: The Point-To-Point Subject Name adapter command (-PPN) might not be used on the same output card as the Subject Name adapter command (-SBN).

Service (-SERVICE)

Use the Service adapter command (-SERVICE) for data sources or data targets to specify the service group or port that the TIBCO Rendezvous daemon will use for the current session.

```
-SERVICE service_name
```

Option

Description

service_name

This is the service group or port to be used for the current session.

For example, to specify the **7501** service port , the syntax would be:

```
-SERVICE 7501
```

The adapter uses a single session for all TIBCO Rendezvous non-certified messages and a separate session for each TIBCO Rendezvous PRO certified subject.

Trace (-T)

Use the Trace adapter command (-T) to produce a log file with the default name **m4rv.log** in the map execution directory. The Trace adapter command (-T) is used with data sources or data targets to enable adapter tracing. The trace file contains information about internal adapter states.

```
-T[E] [full_path]
```

Option

Description

E

Produce a trace file containing only the adapter errors that occurred during map execution.

full_path

This is the name of the trace file. If no name is specified, the default file name is **m4rv.log**.

For example, to specify the **tibrcv.log** trace file name in the map directory, the syntax would be:

```
-T tibrcv.log
```

Note: You can override the adapter command line trace options dynamically using the Management Console. See "Dynamic Adapter Tracing" in the *Launcher* documentation for detailed information.

Syntax summaries

The following are syntax summaries listed for the TIBCO Rendezvous Adapter commands for data sources and targets.

- [Data Sources](#)
- [Data Targets](#)
- [Example Files](#)

Data Sources

The following is the syntax of the TIBCO Rendezvous Adapter commands used for data sources:

```
-SBN subject_name [-CMN [session_name] [-CONFIRM] [-NOM] [-LFN [ledger_name]]]  
[-DAEMON full_path]  
[-DQ QName[, ScdWt[, Hrtbt][, Actvn]]]  
[-ERN subject_name]  
[-HDR]  
[-LSN {0|S|wait_time_in_sec}]  
[-MSG *]  
[-NETWORK name]  
[-OPAQUE]  
[-PP]  
[-QO LstWt[, AccTm[, CplTm][, Tasks]]]  
[-QTY {value|S}][  
[-SERVICE name]  
[-T[E] [full_path]]
```

Data Targets

The following is the syntax of the TIBCO Rendezvous Adapter commands used for data targets:

```
-SBN subject_name [-CMN session_name [-REPLY reply_name] [-LFN [ledger_name]]]
[-LCMN session_name]
[-DAEMON full_path]
[-ERN subject_name]
[-NETWORK name]
[-OPAQUE]
[-REPLY reply_name]
[-RQSTLSN request_lsn_time]
[-RQSTRPL]
[-SERVICE name]
[-T[E] [full_path]]
```

Example Files

The TIBCO Rendezvous Adapter example files are located in the following directory: *install_dir\examples\adapters\tibrv*.

- [Type Tree Example](#)
 - [Map Example](#)
-

Type Tree Example

The **tibco_rv.mtt** type tree is suitable for receiving and sending complex, opaque, or simple, self-described data. To send and receive opaque data, use the **OpaqueMessage** group. This group represents the structure of the data that is being passed to the adapter in the mapping process.

Map Example

The following TIBCO Rendezvous map is provided.

- **rvmsg.mms** - This is a map source file that contains the following six maps:
 - **request** is a map that publishes a message on the **rv.test.out** subject with the request for a reply. It waits three seconds for the reply.
 - **requestreply** is a map that listens on the **rv.test.out** subject. When it receives the message, it maps the reply subject name from the input data header field to the subject name in the output data header field and publishes the message on that subject.
 - **rvcmsg_in** is a map that listens for certified messages on the **rv.test.out** subject. When it receives the message, it maps the data to **out.txt**. Because the **receiver.ldg** ledger file name is specified, the **receiver** acts as the persistent certified correspondent.
 - **rvcmsg_send** is a map that publishes a certified message on the **rv.test.out** subject. The certified correspondents are **sender** and **receiver**. Because the **ledgerout.ldg** ledger file name is specified, **sender** acts as the persistent certified correspondent.
 - **rvmsg_in** is a map that listens for messages on the **rv.test.out** subject. When it receives the message, it maps the data to **out.txt**. Because the **receiver.ldg** ledger file name is specified, the **receiver** acts as the persistent correspondent.
 - **rvmsg_send** is a map that publishes a message on **rv.test.out** subject. The certified correspondents are **sender** and **receiver**. Because the **ledgerout.ldg** ledger file name is specified, **sender** acts as the persistent correspondent.
-

Troubleshooting

For information about error codes and messages returned by the adapter, see [Return Codes and Error Messages](#).

- [Trace Log](#)
 - [Advisory messages](#)
-

Trace Log

Use the Trace adapter command (-T) to create a trace file that reports adapter activity information, recording the events that occur while the adapter is retrieving and sending data. The trace command produces a log file with the default name **m4rv.log** file in the map execution directory.

The trace file contains information about connections to TIBCO Rendezvous, message sizes, error messages, and adapter error messages.

Advisory messages

The TIBCO Rendezvous software generates advisory messages to notify users of warnings and error conditions. The default advisory subject has the base name **_RV.>**. It is possible to map TIBCO Rendezvous advisory and error subjects by specifying them as the subject name on the command line of an input map card.

For the Command Server only, the TIBCO Rendezvous Adapter registers the subject **_RV.>** by default, for non-certified messages only and logs these message receipts to the **m4rv.log** file in the map directory when the Trace adapter command (-T) is specified.

Return Codes and Error Messages

Return codes and messages are returned when the particular activity completes. Return codes and messages might also be recorded as specified in the audit logs, trace files, execution summary files, and so on.

- [Messages](#)
-

Messages

The following is a listing of all the return codes and messages that can be returned as a result of using the TIBCO Rendezvous Adapter for sources or targets.

In addition to the following return codes, the TIBCO Rendezvous Adapter can return TIBCO Rendezvous error codes. Consult the TIBCO Rendezvous documentation for descriptions of these error codes.

Note: Adapter return codes with positive numbers are warning codes that indicate a successful operation. Adapter return codes with negative numbers are error codes that indicate a failed operation.

Return Code	Message
1	Could not close RV
2	Extraneous data in the complex message found
3	WARNING: Extra messages were fetched and not processed
4	WARNING: Timeout period has expired
-1	Command line is in error
-2	Could not open RV
-3	Could not connect to RV transport
-4	Unknown error encountered
-5	Could not publish message
-6	Could not retrieve message
-7	Failed to allocate memory
-8	Triggering failed
-9	Error in specified number of watches
-10	Function could not initialize: improper parameter
-13	Message not in the cache
-14	Exceeded the max thread count
-19	Timed out waiting for rvcm_Confirm
-20	Overflowed buffer

Archive (Zip) Adapter

This is an introduction to the Archive (Zip) Adapter. Use the adapter with a Command Server, Launcher, Software Development Kit, or map in a map rule.

- [Overview](#)
 - [System requirements](#)
 - [Command alias](#)
 - [Archive \(Zip\) Adapter commands](#)
 - [Syntax summary](#)
 - [Return codes and error messages](#)
-

Overview

Use the Archive (Zip) adapter to compress or extract files.

System requirements

The minimum system requirements and operating system requirements for the Archive (Zip) adapter are detailed in the release notes. It is assumed that a Command Server has already been installed on the computer where the adapter is to be installed for run-time purposes.

Command alias

Adapter commands can be specified by using a command string on the command line or by creating a command file that contains adapter commands. The execution command syntax is:

```
-IA[alias] card_num  
-OA[alias] card_num
```

where -IA is the Input Source Override execution command and -OA is the Output Target Override execution command, *alias* is the adapter alias, and *card_num* is the number of the input or output card. The following table shows the adapter alias and its execution command.

Adapter	Alias	As Input	As Output
Archive (Zip)	ZIP	-IAZIP <i>card_num</i>	-OAZIP <i>card_num</i>

Archive (Zip) Adapter commands

This is a description of the functions and use of the Archive (Zip) adapter commands and their options.

- [List of commands](#)

List of commands

The following table lists valid commands for the Archive (Zip) Adapter, the command syntax, and whether the command is supported (✓) for use with data sources, data targets, or both.

The Archive (Zip) Adapter commands do not require a hyphen before the command. The Archive (Zip) Adapter commands do not follow the command syntax of other adapters.

Command	Syntax	Source	Target
Archive File (FILE)	FILE,archive_name,files_to_zip		✓
Archive Data (MEMORY)	MEMORY,archive_name,filename_of_archived_data		✓
Archive Restore	archive_name,files_to_unzip	✓	

- [Archive File \(FILE\)](#)
- [Archive Data \(MEMORY\)](#)
- [Archive Restore](#)

Archive File (FILE)

Use the Archive File adapter command (FILE) to produce a compressed archive file in the map directory. Data passed from the output card is ignored.

FILE,archive_name,files_to_zip

Option

Description

archive_name

Specify the name of the resulting archive file.

,

The comma is a required separator between options.

files_to_zip

Specify the file to be archived. The *files_to_zip* must be a fully qualified path. If the file name is not a fully qualified path, the adapter returns a success status, but the file is not added to the archive.

Specify multiple files as a space-delimited list of file names and paths. You can also use standard DOS wildcards (*) and (?). Enclose long filenames in double quotation marks ("longfilename"). Do not include spaces before the first file name or after the last file name.

Archive Data (MEMORY)

Use the Archive Data adapter command (MEMORY) to produce a compressed archive file from the data supplied by the output card or PUT rule. The data is added to the archive.

MEMORY,archive_name,filename_of_archived_data

Option

Description
<code>archive_name</code>
Specify the name of the resulting archive file.
,
The comma is a required separator between options.
<code>filename_of_archived_data</code>
Specify the file name that will be used to represent the archived data. Enclose a long file name in double quotation marks ("longfilename").

Archive Restore

Use the Archive Restore command to retrieve files from a specified archive. Specify the following arguments with no preceding adapter command:

`archive_name,files_to_unzip`

Option

Description

archive_name
Specify the name of the archive that contains the files to be restored.

,
The comma is a required separator between options.

`files_to_unzip`

Specify the files to be restored from archive. The `files_to_unzip` must be a fully qualified path. If the file name is not a fully qualified path, the adapter returns a success status, but the file is not restored from the archive.

Specify multiple files as a space-delimited list of file names and paths. You can use standard DOS wildcards (*) and (?). Enclose long filenames in double quotation marks ("longfilename"). Do not include spaces before the first file name or after the last file name.

Note: The Archive Restore command restores multiple files only to memory and only as one continuous data stream that concatenates all the selected data files. Multiple files cannot be individually restored to disk.

Syntax summary

This is a description of the Archive (Zip) adapter syntax summary and how it is used.

- [Data sources](#)
- [Data targets](#)
- [Examples](#)

Data sources

The following is the syntax of the Archive (Zip) adapter commands used for data sources:

`archive_name,files_to_unzip`

Data targets

The following is the syntax of the Archive (Zip) Adapter commands used for data targets:

`FILE,archive_name,files_to_zip
MEMORY,archive_name,filename_of_archived_data`

Examples

For the **Source** setting in an input card, select Archive (Zip). In the GET->Source->Command field, enter the Restore adapter command:

`SavRecord,data.txt`

The **Archive Restore** command extracts and restores a file named **data.txt** from an archive file named **SavRecord**.

This example extracts multiple files to memory as a continuous data stream:

`SavRecord,data.txt *.dat abc\def\notes.txt "documents and settings\Administrator\listing.txt"`

From an archive file named **SavRecord**, the **Archive Restore** command extracts and restores:

- A file named **data.txt**
- All files with the ***.dat** extension
- The file **abc\def\notes.txt**

- The file documents and settings\Administrator\listing.txt

To override Archive (Zip) adapter settings for the first output card and create an archive file named **SavRecord** containing the file **data.txt**, the Archive file adapter command string is:

```
-OAZIP1 'FILE,SavRecord,data.txt'
```

Return codes and error messages

Return codes and messages are returned when the particular activity completes. Return codes and messages might also be recorded as specified in the audit logs, trace files, execution summary files, and so on.

- [Messages](#)

Messages

The following is a listing of all the codes and messages that can be returned as a result of using the Archive (Zip) adapter for sources or targets.

Adapter return codes with positive numbers are warning codes that indicate a successful operation. Adapter return codes with negative numbers are error codes that indicate a failed operation.

Return Code	Message
0	OK
-1	Invalid command line: missing zip file name after 'file' keyword.
-1	Invalid command line: missing zip file name after 'memory' keyword.
-1	Invalid command line: missing zip file name after 'echo' keyword.
-1	Invalid command line: missing zip member name (after zip file name).
-1	Invalid command line: expected 'file', 'memory', or 'echo' as first argument.
-1	Invalid command line: missing zip file name.
-600	Internal Error: Resource Manager Error
-?	ZIP_ADD: DynaZip error occurred (????).
-?	ZIP_MEMTOFILE: DynaZip error occurred (????).
-?	UNZIP_COUNTALLZIPMEMBERS: DynaZip error occurred (????).
-?	UNZIP_COUNTNAMEDZIPMEMBERS: DynaZip error occurred (????).
-?	UNZIP_GETNEXTNAMEDZIPINFO: DynaZip error occurred (????).
-?	UNZIP_FILETOMEM: DynaZip error occurred (????).
-?	-? is a DynaZip API error return (non-zero) made negative if > 0 and where ???? is a DynaZip API supplied error message.

Resource Registry overview

The Resource Registry application can be used to define resource name aliases for source and target resources that are specified in map cards.

A resource name is an alias that has different values defined for multiple deployment environments. A resource value can be either a command card setting for a map, an adapter command override setting for a map or system component, or an argument to environment functions within a map rule (**RUN**, **PUT**, **GET**, **DBLOOKUP**, **DBQUERY**, and **EXIT**). In addition, other resources that can be aliased include:

- backup files
- audit files and locations
- trace files and locations
- work space locations
- external parser location (defined in the Type Designer)
- map server location (defined in the Integration Flow Designer)

Using the Resource Registry, a resource name is associated with one or more logical virtual servers (deployment environments). Each logical virtual server might have a different resource value for a given resource name.

For example, suppose **Payroll** is a resource name. **Payroll** is associated with a **Test**, **Development**, and **Production** logical server. For **Test**, **Payroll** might be an SQL Server query. For **Development**, **Payroll** might be an Oracle query. For **Production**, **Payroll** could be a DB2® query.

This feature minimizes the maintenance of systems and maps when moving them to different environments.

With the Resource Registry you define the various resource values once, and those values get resolved at run-time (map execution) by the map server (Command Server, Launcher, or Transformation Extender Programming Interface), thereby eliminating the need to modify maps during design-time.

Without the Resource Registry, you must manually edit input and output card settings for each platform the system and/or maps are moved to in order to reflect the appropriate parameters (resource value) for that environment.

For the purpose of this documentation, the generic term **map server** is used when referencing the Command Server, Launcher, or Transformation Extender Programming Interface.

See the Command Server documentation for information on items to take into consideration if you use resource names in maps running on a OS/390® Batch or CICS® version of the Command Server.

Tip:

A Resource Registry example is provided with the Design Studio at *install_dir/examples/general/rsrcreg*.

- [**Resource Registry files**](#)

The Resource Registry files include the resource name file, the resource configuration file, and the master key file.

- [**Resource Registry interface overview**](#)

The Resource Registry provides a graphical user interface to create and manage resource aliases.

- [**Resource Registry map deployment overview**](#)

Resource configuration files are used by a map server to determine available resources and their values when executing a map. Configuration files are specified for a map server through an application option, initialization function, or the config.yaml file.

- [**Creating a Resource Registry resource name alias**](#)

Prior to using the Resource Registry, you must create a resource name alias.

- [**Access to the Resource Registry**](#)

The IBM Transformation Extender installation program adds an entry for the Resource Registry to the Design Studio programs menu.

Resource Registry files

The Resource Registry files include the resource name file, the resource configuration file, and the master key file.

Resource name file (.mrn)

Contains a named set of virtual servers and a named set of resources.

Resource configuration file (.mrc)

Specifies the associated resource name file and which virtual servers are active for a map server.

Master key file (.mkf)

Contains one or more keys that are used to encrypt resource values in the .mrn files.

The resource name, resource configuration, and master key files are in XML format.

Resource Registry interface overview

The Resource Registry provides a graphical user interface to create and manage resource aliases.

- [**Resource Registry navigator overview**](#)

The Resource Registry navigator graphically presents all of the opened resource name and configuration files, and the resource values that they contain.

- [**Resource Registry menu commands**](#)

Actions are performed in the Resource Registry using menu commands.

Resource Registry navigator overview

The Resource Registry navigator graphically presents all of the opened resource name and configuration files, and the resource values that they contain.

Resource Registry menu commands

Actions are performed in the Resource Registry using menu commands.

When working with resource aliases, there are several ways you can activate commands:

- Select functionality using the menu bar.
- Right-click an entry in the Resource Registry window to display its context menu.
- Press **Insert** to add resources and map systems.
- Press **Delete** to delete resources and map systems.

Resource Registry commands are available as listed above; however, most procedural information in this documentation uses the menu access as a default method. Use the activation method most convenient for you.

Resource Registry map deployment overview

Resource configuration files are used by a map server to determine available resources and their values when executing a map. Configuration files are specified for a map server through an application option, initialization function, or the config.yaml file.

See "[Specifying Configuration Files for a Map Server](#)" for more information.

Creating a Resource Registry resource name alias

Prior to using the Resource Registry, you must create a resource name alias.

To create a resource name alias, you must first define the resource and virtual server names, and then define the value for each resource name and virtual server combination.

To define a resource name alias

1. Create a resource name (**.mrn**) file.
2. Define the resource name(s).
3. Define the virtual servers.
4. Define the resource values for each resource name and virtual server combination.
5. Save the **.mrn** file.
6. Create a resource configuration file (**.mrc**).
7. Associate the **.mrn** file and active server with map server.
8. Save the **.mrc** file.
9. Specify the name and location of the **.mrc** file for the Command Server or Launcher through an application option.

Access to the Resource Registry

The IBM Transformation Extender installation program adds an entry for the Resource Registry to the Design Studio programs menu.

For UNIX platforms, the Resource Registry can be accessed from:

`install_dir\resourceregistry.sh`

Resource name files

With the Resource Registry, you define the resource and then associate that resource to a deployment environment (virtual server).

For example, a resource could be defined as an Oracle query on a **Development** server (Windows) and an SQL Server query on a **Test** server (HP-UX).

Each named resource has a specified value for that resource for each virtual server. If a resource value contains sensitive data (such as passwords), you can encrypt the value.

The resource name and virtual server name must be unique and are case-sensitive.

The value of the **Command** setting (adapter-specific commands) in input and output cards and in a **GET** or **PUT** function within a map rule cannot exceed 2000 bytes. The expansion of the resource value is included in this limitation.

See the Map Designer documentation for more information about settings and map rules.

- [Creating Resource Registry name files](#)

You need to define a name file for the Resource Registry.

You need to define a name file for the Resource Registry.

The resource name (**.mrn**) file contains specifications for the resource such as:

- Resource name
- Resource value
- Virtual server that the resource is associated with
- Encryption setting for the resource value, encryption key ID, and master key file

You can use the same resource name across all platforms. The **.mrn** file is platform-independent.

You can define an unlimited number of sets of resources and virtual servers in one **.mrn** file.

See the Command Server documentation for information on items to take into consideration if you use resource names in maps running on a z/OS® Batch or CICS® version of the Command Server.

To create a resource name file:

1. In the navigator, select Resource Files.
2. From the **File** menu, choose **New**.
The New dialog appears.
3. Specify the location of the resource name file by browsing your file structure.
4. In the **File name** field, enter the name of the resource name file.
5. Click **New**.
The resource name file displays in the navigator.

- **[Defining Resource Registry names](#)**

Resource names are defined for the currently active resource name file in the Resource Registry.

- **[Defining resource names with environment variables](#)**

You can avoid customizing resource name and configuration files for specific deployments by defining resources as operating system environment variables. At run time, the Resource Registry **\$GETENV()** function resolves resource aliases that are defined as environment variables. The **\$GETENV()** function can be used with or without Resource Registry encryption.

- **[Defining Resource Registry virtual servers](#)**

Virtual servers are created to represent the deployment environment for the Resource Registry.

- **[Defining Resource Registry values](#)**

After you have defined the resource and virtual server names for the Resource Registry, the value for each resource name and virtual server combination is defined. The resource value can be encrypted in the .mrn file as a security precaution.

Defining Resource Registry names

Resource names are defined for the currently active resource name file in the Resource Registry.

To create a resource name

1. In the navigator, right-click Resources and select Add.
The **Add** New Resource dialog appears.
2. In the **Name** field, enter the name of the resource.
3. Click OK.
The resource displays in the navigator.

Defining resource names with environment variables

You can avoid customizing resource name and configuration files for specific deployments by defining resources as operating system environment variables. At run time, the Resource Registry **\$GETENV()** function resolves resource aliases that are defined as environment variables. The **\$GETENV()** function can be used with or without Resource Registry encryption.

1. Use an operating system environment variable to define the resource. For example, on a Microsoft Windows system, use Control Panel > System > Advanced system settings > Advanced > Environment Variables. The environment variable might be case-sensitive, depending on the operating system.
2. In the Resource Registry resource name (.mrn) file, specify the Resource Value as **\$GETENV(env_var)** where **env_var** is the environment variable name. The string must not contain any spaces. The **\$GETENV()** function is not case sensitive.
If the specified environment variable is not defined, the **\$GETENV()** function resolves to an empty (zero-byte) string.

An informational log file message like the following indicates environment variable usage. For example:

```
DTXLN2154I - Fri Feb 10 07:32:21 2017 - tid: 12068 inst: -1 - Resource Registry Info : Alias [username] uses environment variable [USERNAME] (mresname.c:1243)
```

Defining Resource Registry virtual servers

Virtual servers are created to represent the deployment environment for the Resource Registry.

To create a server resource

1. In the navigator, select Servers.
2. From the **Edit** menu, choose **Add**.
The **Add** New Server dialog appears.
3. In the **Name** field, enter the name of the server resource.
It is good practice to keep the same naming conventions for the server resources in the Resource Registry as used for the server definitions in the Integration Flow Designer. See the Integration Flow Designer documentation for more information.
4. Click OK.
The server resource displays in the navigator.

Defining Resource Registry values

After you have defined the resource and virtual server names for the Resource Registry, the value for each resource name and virtual server combination is defined. The resource value can be encrypted in the .mrn file as a security precaution.

To define the resource value:

1. In the navigator, select the resource name.
2. From the **Edit** menu, choose **Edit**.
The Edit Resource dialog appears.
3. In the **Resource Value** field, enter the value of the resource for a specified server.
Note: The resource value is only validated during map execution, not during the design phase.
4. In the **Encryption** field, select ON to protect the resource name file or accept OFF as the default.
5. Click OK.

Implementing Resource Registry resource names

To use a Resource Registry resource name, enclose the resource name with the percent (%) symbol.

For example, to use a different root file path on different virtual servers without changing the system definition, the resource name **RootPath** is specified in an output card. At execution time, the resource value for %RootPath% is resolved.

- [**Resource Registry alias overview**](#)

If aliases are used with the Resource Registry, a single % sign in a command is ignored. If a command requires more than one % sign, an alias must exist that contains the % sign.

- [**Resource Registry alias usage overview**](#)

When resource aliases are used with the Resource Registry, aliases are resolved for the resources when each map is initialized.

- [**Resource Registry alias expansion in external interface functions**](#)

During Resource Registry map execution, aliases are resolved for external interface functions.

- [**Resource Registry aliases in audit and debug files**](#)

Resource Registry aliases are also used in audit and debug files.

Resource Registry alias overview

If aliases are used with the Resource Registry, a single % sign in a command is ignored. If a command requires more than one % sign, an alias must exist that contains the % sign.

For example, if a command requires %TID%, then an alias should exist that has a value of %TID%.

Aliases in commands are interpreted in three ways depending on the following:

- Aliases are used and the alias has a value
- Aliases are used but the alias name does not exist
- Aliases are not used, which means that:
 - No configuration file was specified
 - The configuration file does not exist
 - There is an invalid configuration file (or resource files)

To demonstrate alias meanings, the following table shows the values of the command cat_%verb%_dog in three different scenarios:

Action	Result
Aliases are used and verb has a value of fights	cat_fights_dog
Aliases are used but verb is not defined	cat_dog
Aliases are not used	cat_%verb%_dog

Resource Registry alias usage overview

When resource aliases are used with the Resource Registry, aliases are resolved for the resources when each map is initialized.

When the Command Server or Launcher is started, or for users of the Transformation Extender Programming Interface or Java API, if the resource configuration file does not exist or is not specified, resource aliases are not used and any aliases in any commands remain unaliased (keeping the % signs in the command).

When resource aliases are used, aliases are resolved for the following resources upon each map initialization:

Each input card's **FilePath** or **Command**

2000 bytes

Each output card's **FilePath** or **Command**

2000 bytes

Each card's backup path

255 bytes

Audit file path and name

259 bytes

Trace file path and name
259 bytes
Work file path
259 bytes
Map server location
259 bytes

Resource Registry alias expansion in external interface functions

During Resource Registry map execution, aliases are resolved for external interface functions.

During map execution, aliases can be resolved for the following external interface functions and can expand the string up to 256 bytes greater than the original length.

Function	What's Resolved	Size
EXIT	library name function name	256 bytes extra 256 bytes extra
DDEQUERY	application name topic text	256 bytes extra 256 bytes extra 256 bytes extra
RUN	map name	256 bytes extra
DBQUERY	SQL_statement mdq_filename database_name parameters	256 bytes extra 256 bytes extra 256 bytes extra 256 bytes extra
DBLOOKUP	SQL_statement mdq_filename database_name parameters	256 bytes extra 256 bytes extra 256 bytes extra 256 bytes extra
GET	adapter type adapter command	256 bytes extra 256 bytes extra
PUT	adapter type adapter command	256 bytes extra 256 bytes extra
external parser (document verification)	Location property associated with the Document type property of a type tree object	256 bytes extra

Resource Registry aliases in audit and debug files

Resource Registry aliases are also used in audit and debug files.

When you select the **Encrypt** option for Resource Registry resource aliases, the aliases appear masked (with asterisks) anywhere they are displayed in audit logs and debug files.

The masked resource alias is represented by five asterisks (*****) followed by a space, followed by the resource alias in parenthesis. The masked resource aliases could appear in the following:

- Source and target command lines/filenames
- Backup files
- Audit directories/files
- Trace directories/files
- Work file directories

Resource aliases used in **GET**, **PUT** and **RUN** rules do not appear in environment trace or log files.

- [Resource Registry trace file usage overview](#)

Use the `/runtime/Launcher/log` logging options in the config.yaml configuration file to set the Resource Registry trace options. The trace file logs encrypted aliases and displays them as masked values.

- [Resource Registry encrypted alias usage with masked command lines](#)

Resource Registry aliases can use masked command lines in some system components.

Resource Registry trace file usage overview

Use the `/runtime/Launcher/log` logging options in the config.yaml configuration file to set the Resource Registry trace options. The trace file logs encrypted aliases and displays them as masked values.

Resource Registry encrypted alias usage with masked command lines

Resource Registry aliases can use masked command lines in some system components.

Masked command lines are displayed in the following areas:

- Audit log
 - Execution summary
 - Map settings
 - Data (card) settings
- Launcher log (**Launcher.txt**)
 - Map errors
 - Map warnings
 - System configuration information
- Compound****.log
- GPF/abort logs

Resource Registry encryption overview

You can encrypt resource values that contain sensitive data, such as user credentials and database passwords. The Resource Registry uses a secure encryption key that it stores separately from the resource values. This means that you can share a resource name file but withhold the key file to protect confidential information.

The Resource Registry uses the Advanced Encryption Standard (AES) algorithm to encrypt and decrypt resource alias values by using a secure encryption key.

Encrypting a resource value:

1. Generates an encryption key.
2. Assigns an ID to the key.
3. Encrypts the key itself, creating the master encryption key.
4. Generates a master key (.mkf) file and stores the master encryption key in it.
5. Stores the key ID and .mkf file name and path in the resource name (.mrn) file.

During map execution, IBM Transformation Extender uses the master encryption key that is identified in the .mrn file to decrypt the resource value.

You can specify a custom passphrase when you generate a new .mkf file. The passphrase lets you regenerate the master encryption key if the .mkf file is damaged or lost. When you don't supply a custom passphrase, the Resource Registry randomly generates one. A randomly generated passphrase is more secure. You cannot regenerate the master encryption key with a randomly generated passphrase.

- [Resource Registry name files and encryption keys](#)

Each Resource Registry master encryption key has a numeric key ID that uniquely identifies it. A resource name (.mrn) file that has encrypted values specifies the key ID and file name of its master encryption key.

- [Resource Registry master key file and encrypted value display](#)

The presence of the master key file for an encrypted resource determines whether the resource value displays as plain text or as an encrypted string.

- [Encrypting and resetting Resource Registry values](#)

You can encrypt a resource value, remove resource encryption, or reset encrypted values. You can convert a legacy resource name (.mrn) file to use the Advanced Encryption Standard. With the correct passphrase, you can regenerate a lost or damaged master encryption key.

Resource Registry name files and encryption keys

Each Resource Registry master encryption key has a numeric key ID that uniquely identifies it. A resource name (.mrn) file that has encrypted values specifies the key ID and file name of its master encryption key.

The default master key file name is itx.mkf, located in the .mrn file directory.

For example, the following .mrn file is encrypted by the key with ID 6734, which is stored in the itx.mkf file:

```
<MRN version="1.0" key_id="6734" key_file="itx.mkf">
...
<Resource>
  <Name>db_password</Name>
  <Value Server="Production" encrypt="ON"
id="_home_markdown_jenkins_workspace_Transform_in_SSVD8_11.0.1_com.ibm.websphere.dtx.rsrcreg.doc_references_r_adaptref_master_
key_file_5623" iv="240b35b4358e185a071548a0b0ac7d22">
9ad3d263a52819cf944cb4b20106784e0747ac006e7e8d5af4053cd06ed64469</Value>
  <Value Server="Development" encrypt="OFF">dev_password</Value>
</Resource>
...
</MRN>
```

In the master key file, the entry for the master encryption key with ID 6734 has a format that is similar to the following example:

```
<MasterKeys>
  <entry>
    <cipher>AES-256-CBC</cipher>
    <key>0be4d263a52819cf944cb4b20106784e0747ac006e7e8d5af4053cd06ed64460</key>
    <iv>351c464c5469f296b182659a1c1ad8e3</iv>
    <id>6734</id>
  </entry>
</MasterKeys>
```

A master key file can contain multiple master encryption keys. You manually edit a master key file to append keys to it or remove keys from it.

Resource Registry master key file and encrypted value display

The presence of the master key file for an encrypted resource determines whether the resource value displays as plain text or as an encrypted string.

When the .mkf file is present on the computer, the Resource Registry displays encrypted values as plain text ("in the clear"). To allow access to encrypted values in an .mrn file, you can either:

- Provide the master key (.mkf) file that decrypts the resource values.
- Provide the passphrase that regenerates the .mkf file.

Alternatively, you can withhold the .mkf file and passphrase to protect sensitive resource values. When the .mkf is not present or does not contain the correct master encryption key, the Resource Registry displays encrypted values as encrypted strings. Without the .mkf file, you can use the Resource Registry to view the read-only encrypted values or to clear the values and enter new ones. Optionally, you can encrypt the new values with a new key in a new .mkf file.

Encrypted resource values always display as encrypted strings when you open the .mrn file in a text editor.

On UNIX® systems, only the user who generates the .mkf file has permission to read and write to it. Other users cannot read or write to the .mkf file.

Encrypting and resetting Resource Registry values

You can encrypt a resource value, remove resource encryption, or reset encrypted values. You can convert a legacy resource name (.mrn) file to use the Advanced Encryption Standard. With the correct passphrase, you can regenerate a lost or damaged master encryption key.

- [Encrypting a Resource Registry value and generating a master key file](#)

Encrypting a resource value in a resource name (.mrn) file for the first time creates a master encryption key and the master key (.mkf) file where the key is stored. When you subsequently encrypt other resource values in that .mrn file, the Resource Registry uses the same master encryption key and .mkf.

- [Converting a Resource Registry name file to AES encryption](#)

You can convert a legacy resource name (.mrn) file that contains encrypted resources to the Advanced Encryption Standard (AES) encryption format by using a new or existing master encryption key. If you do not want to convert to AES format, you can continue to use your existing .mrn files without converting them.

- [Resetting encrypted Resource Registry values](#)

When the master key (.mkf) file is present with Resource Registry, you control encryption at the individual resource level. When the .mkf file is not present and you reset the encryption of a single resource value, the Resource Registry resets the encryption of all of the values in the .mrn file. The Resource Registry clears the encrypted strings and sets the encryption flags to OFF. It also removes the master key file name and key ID from the .mrn file.

- [Regenerating a Resource Registry master key file](#)

You can regenerate a lost or damaged Resource Registry master encryption key with its custom password.

Encrypting a Resource Registry value and generating a master key file

Encrypting a resource value in a resource name (.mrn) file for the first time creates a master encryption key and the master key (.mkf) file where the key is stored. When you subsequently encrypt other resource values in that .mrn file, the Resource Registry uses the same master encryption key and .mkf.

1. In the Resource Registry, edit the resource that you want to encrypt or the virtual server that includes the resource.
2. In the Edit window, set the Encryption field to ON and click OK.
3. Click Yes to enable the Advanced Encryption Standard format.
4. Select Change master key file or regenerate master key.
5. In the Encryption Master Key window:
 - a. Select Generate a new master key.
 - b. Accept the default master key file name itx.mkf, or specify your own master key file name and path.
The path to the master key file must be relative to the .mrn file location.
 - c. Choose whether to use a custom passphrase:
 - If you do not select Use custom passphrase, the Resource Registry generates the key with a random passphrase. A randomly generated passphrase is more secure, but you cannot use it to regenerate a master encryption key.
 - To create an encryption key that you can regenerate, select Use custom passphrase. Enter a passphrase up to 32 bytes in length in the Passphrase field.
 - d. Click OK.

The Resource Registry encrypts the resource value and creates the .mkf file.

6. Save the changes to the .mrn file.

Converting a Resource Registry name file to AES encryption

You can convert a legacy resource name (.mrn) file that contains encrypted resources to the Advanced Encryption Standard (AES) encryption format by using a new or existing master encryption key. If you do not want to convert to AES format, you can continue to use your existing .mrn files without converting them.

1. Open the legacy .mrn file in the Resource Registry.
2. Open the encrypted resource or server and click OK to display the Encryption prompt.
If you click Cancel, the .mrn file retains the legacy encryption format.
3. Click Yes to enable AES encryption format.
 - Select Use an existing master key to browse to an existing master key file (.mkf) and choose a key ID.
 - Select Generate new master key and follow the procedure to generate a new master encryption key and .mkf file.
4. Click OK.

Related tasks

- [Encrypting a Resource Registry value and generating a master key file](#)
-

Resetting encrypted Resource Registry values

When the master key (.mkf) file is present with Resource Registry, you control encryption at the individual resource level. When the .mkf file is not present and you reset the encryption of a single resource value, the Resource Registry resets the encryption of all of the values in the .mrn file. The Resource Registry clears the encrypted strings and sets the encryption flags to OFF. It also removes the master key file name and key ID from the .mrn file.

1. Use the Resource Registry to open an .mrn file that contains encrypted values on a computer where the .mkf file is not present.
 2. Open an encrypted resource or a server that contains encrypted resources.
 3. Click Reset encrypted values to empty plain text and click OK.
 4. Save the resource or server file.
-

Regenerating a Resource Registry master key file

You can regenerate a lost or damaged Resource Registry master encryption key with its custom password.

1. Use the Resource Registry to open an encrypted resource or its server.
 2. Click Change master key file or regenerate master key.
 3. In the Encryption Master Key window:
 - a. Click Generate new master key.
 - b. Click Use custom passphrase.
 - c. Enter the passphrase of the missing or damaged master key file.
 4. Click OK to create the master key file and close the resource or server.
-

Resource configuration files overview

Resource configuration files (**.mrc**) specify the associated resource name file and which virtual servers are active for a map server.

- [**Creating resource configuration files**](#)
After defining your resources and their associated virtual servers, you need to create a resource configuration file.
- [**Creating a Launcher resource configuration file**](#)
After defining your resources and their associated virtual servers, you need to create a resource configuration file.
- [**Creating a Command Server resource configuration file**](#)
After defining your resources and their associated virtual servers, you need to create a resource configuration file.
- [**Transformation Extender Programming Interface and Java API resource configuration files**](#)
After defining your resources and their associated virtual servers, you need to create a resource configuration file.
- [**Saving resource configuration files**](#)
After defining your resources and their associated virtual servers, you need to create and save resource configuration files.
- [**Resource configuration files for map servers**](#)
After defining your map server, you need to create a resource configuration file.

Creating resource configuration files

After defining your resources and their associated virtual servers, you need to create a resource configuration file.

After you have defined your resources and their associated virtual servers, the next step is to create a configuration file. This file is used by the Command Server, Launcher, Transformation Extender Programming Interface, Java API, and Map Designer in determining which virtual servers are active for that map server when executing maps.

The resource configuration file is the key to tying everything together. This file defines the active virtual server (in a resource name file), for the map server and the available resources for that virtual server.

The .mrc file contains specifications for the map server such as:

- Active virtual server(s) for a map server
 - Associated.mrn file
- Resource name files are associated with the Launcher at the system level.

The specified active resource configuration file is used at runtime for dynamic resource resolution. Because an .mrn file and its file path are specified, the .mrc file is platform-dependent.

To create a resource configuration file

1. In the navigator, select Configuration Files.
2. From the **File** menu, choose **New**.
The Save As dialog appears.
3. Specify the location of the configuration name file by browsing your file structure.

4. In the **File name** field, enter the name of the configuration file.

Note: A default filename, resource.mrc, is specified in the config.yaml file. If you use a different filename and want to specify the .mrc file for a map server through the .ini file, you must manually modify the .ini file with the correct filename. See "[Initialization File](#)" for more information.

5. Click Save As.

The resource configuration file appears in the navigator.

Creating a Launcher resource configuration file

After defining your resources and their associated virtual servers, you need to create a resource configuration file.

A resource name file (**.mrn**) and active virtual server are associated with the Launcher at the system level. Launcher system files (**.msl**) are created using the Integration Flow Designer.

At runtime, if an **.mrn** file is not specified for Launcher, the **.mrn** file specified for **Global** is used by default. If **.mrn** files are specified for both Launcher and **Global**, the **.mrn** file specified for Launcher takes precedence.

Adding a system for the Launcher

1. In the navigator, select Launcher.
2. From the **Edit** menu, choose **Add**.

The Add New System dialog displays.

3. Click **Browse** to select the system file to associate with the Launcher.
4. Click OK.

The associated system displays in the navigator.

Activating a server for the Launcher

1. In the navigator, select the system.
2. From the **Edit** menu, choose **Edit**.

The Edit Resources dialog displays.

3. Click Add.
4. The Select dialog displays.

5. Select the resource name file (**.mrn**) to associate with the map server.
6. In the **Active Virtual Server** field, select the server to activate for the map server.

The specified **.mrn** file and active virtual server are associated with the map server.

Creating a Command Server resource configuration file

After defining your resources and their associated virtual servers, you need to create a resource configuration file.

A resource name file is used by the Command Server in determining which virtual servers are active when executing maps.

At runtime, if the **.mrn** file is not specified for Command Server, the file specified for **Global** is used. If **.mrn** files are specified for both Command Server and **Global**, the Command Server file takes precedence.

To activate a server for the Command Server

1. In the navigator, select Command Server.
2. From the **Edit** menu, choose **Edit**.

The Edit Resources dialog displays.

3. Click Add.
4. The Select dialog displays.

5. Select the resource name file (**.mrn**) to associate with the Command Server.
6. In the **Actual Virtual Server** field, select the server to activate for the Command Server.

The specified **.mrn** file and active virtual server are associated with the Command Server.

Transformation Extender Programming Interface and Java API resource configuration files

After defining your resources and their associated virtual servers, you need to create a resource configuration file.

A resource name file (**.mrn**) is used by the Transformation Extender Programming Interface and Java API in determining which virtual servers are active for a map server when executing maps.

To associate a resource name file and activate a server for the Transformation Extender Programming Interface or Java API, follow the instructions outlined in section "[Command Server](#)". However, select the **Global** object instead of the Command Server object when associating .mrn files and active virtual servers. The **Global** object represents the Map Designer, Transformation Extender Programming Interface, and Java API map servers.

Saving resource configuration files

After defining your resources and their associated virtual servers, you need to create and save resource configuration files.

After you have created the configuration file and associated .mrn file(s) and active server(s) with the map server, you need to save the .mrc file.

To save the configuration file

1. In the navigator, select the resource configuration file.
2. From the **File** menu, choose **Save**.

The resource configuration file is saved.

Resource configuration files for map servers

After defining your map server, you need to create a resource configuration file.

A configuration file is specified for a map server using one of the following methods:

- Menu option in the Map Designer or Launcher Administration
- Transformation Extender Programming Interface
- Java API
- config.yaml initialization file
- [Map Designer resource configuration file](#)
You need to create a resource configuration file for the Map Designer.
- [Launcher resource configuration file](#)
You need to create a resource configuration file for the Launcher.
- [Transformation Extender Programming Interface resource configuration file](#)
You need to create a resource configuration file for the Transformation Extender Programming Interface (TX PI).
- [Java API resource configuration file](#)
You need to create a resource configuration file for the Java API.
- [Initialization resource configuration file](#)
There are various scenarios in which you need to modify the initialization file (config.yaml).

Map Designer resource configuration file

You need to create a resource configuration file for the Map Designer.

An .mrc file can be specified for maps as a run option. When a map runs, the .mrc file defers to the .mrn file, and the virtual server used is derived from the **Global** setting.

To access run options in the Map Designer, from the Design Studio, select **Window** > **Preferences** > **Transformation Extender** > **Map** > **Run Options**.

See the Map Designer documentation for more information.

Launcher resource configuration file

You need to create a resource configuration file for the Launcher.

The associated .mrc file for the Launcher is specified on the General tab in the Launcher Administration interface.

See the Launcher documentation for more information.

Transformation Extender Programming Interface resource configuration file

You need to create a resource configuration file for the Transformation Extender Programming Interface (TX PI).

The associated .mrc file is specified using the **mpiInitAPI (lpszConfigFile)** initialization function or the **mpiInitAPIEx (lpszResourceConfigFile)** initialization function when invoking maps using the Transformation Extender Programming Interface.

See the Transformation Extender Programming Interface documentation for more information.

Java API resource configuration file

You need to create a resource configuration file for the Java API.

The Java API uses the MMap Java™ class and the initializeAPI(java.lang.String resourceConfigFile) method to specify the Resource Registry .mrc configuration file.

See the Java API documentation for more information.

Initialization resource configuration file

There are various scenarios in which you need to modify the initialization file (config.yaml).

If an .mrc file has not been specified for the Command Server or Launcher through an application option, then the map server looks in the config.yaml file for the specified .mrc file. A default filename, resource.mrc, is specified in the config.yaml file. If you create an .mrc file using a different name, and do not specify the filename through an application option, then you need to manually modify the configuration file.

In the config.yaml file, the .mrc file is specified in the **/runtime/Command Server/resourcefile** path for the Command Server and in the **/runtime/launcher/ResourceCfgFile** path for the Launcher. To determine where to specify the .mrc file, search for the string **ResourceCfgFile** and modify the current default .mrc filename.

See the Launcher documentation for more information.

Execution maps

This documentation includes information about the various ways to use execution commands and their options when executing maps. Execution commands are used to specify how to execute a map.

- [Command Server overview](#)
- [Using Command Server commands](#)

Command Server overview

Use execution commands to override the map settings or card settings in a compiled map when a map is run using one of the following methods:

- from a command line prompt
- from the Map Designer
- from the Command Server user interface
- using the `RUN()` function in a map rule
- using the API

Use the map settings and input and output card settings of the Map Designer or the execution settings in the Integration Flow Designer to define map execution. These map execution settings are included in the compiled map file. Many settings compiled into the map can be overridden (or partially overridden) using executions commands and options.

Some execution commands are limited because they can only be used with a Command Server and cannot be used from within a `RUN()` function or using the Platform API. Also, some commands are not available on all platforms. See ["Command Availability"](#) for more information.

- [Coordinating execution commands with compiled settings](#)
- [Compiled map files](#)
- [Command files](#)

Coordinating execution commands with compiled settings

The following table lists valid execution commands and the **MapSetting** or input/output card settings that they override.

- [Map settings](#)
- [Card settings](#)

Map settings

Execution Command	Map Setting
MapAudit (-A)	MapAudit
Close Window (-B)	-
Century (-D)	Century
Fail on Warnings (-F)	Warnings
Ignore Validation Options (-G)	

```
Validation > RestrictionError, SizeError, PresentationError
List (-I)
-
Initialization File Override (-N)
-
WorkSpace Page Size (-P)
    WorkSpace > PageSize
Refresh Status (-R)
-
MapTrace (-T)
    MapTrace
Stop Validation (-V)
    Validation > OnValidationError
WorkSpace (-W)
    WorkSpace
Retry (-Y)
    Retry
Ignore Warnings (-Z)
    Warnings
```

Card settings

```
Execution Command
Card Setting
Input Overrides (-I)
WorkFiles (W)
    SourceRule > FetchAs > WorkArea
Retry (R)
    SourceRule > GET > Retry
Rollback (B)
    SourceRule > GET > Transaction > onFailure
Delete (X)
    SourceRule > GET > Transaction > onSuccess
Output Overrides (-O)
Retry (R)
    TargetRule > PUT > Target > Retry
Append (+)
    TargetRule > PUT > Transaction > Target > onSuccess
Delete (X)
    TargetRule > PUT > Transaction > onSuccess
Rollback (B)
    TargetRule > PUT > Transaction > onFailure
```

Compiled map files

You can create compiled map (.mmc) files using the Map Designer, Integration Flow Designer, or Command Server commands.

- [Map Designer](#)
 - [Integration Flow Designer](#)
-

Map Designer

Using the Map Designer, specify how you want each map to run. These specifications, which are compiled into the map file, include map settings and card settings, as well as execution command overrides for `RUN()` functions which execute a map from a map rule. See the *Map Designer* documentation for more information about map and card settings and the *Functions and Expressions* documentation for information about the `RUN()` function.

Integration Flow Designer

Using the Integration Flow Designer, you can create systems of maps and define execution settings, map settings, and input/output settings. These settings are recorded into the compiled map file that results from building the map within the Integration Flow Designer.

If the system has an execution mode of Command, a command file can be generated that can be executed with a Command Server. Edit this command file using the appropriate execution commands and options.

The following is an example of a command file generated by the Integration Flow Designer for a system named **Inbound EDI**:

```
;;
;;
; Date:   Wed Jun 16 18:15:15 2001 (Version 6.0(126))
; File:  D:\MapsAndTrees\ANSIACK\ProcessEDI.msd
```

```

; System: Inbound EDI
;
; Component: ReceiveEDI
; Map File: D:\MapsAndTrees\ANSIACK\Audit997.mms
; Executable Map: ReceiveEDI
;
D:\MapsAndTrees\ANSIACK\ReceiveEDI.mmc
    -GR
;
; Component: ACK997
; Map File: D:\MapsAndTrees\ANSIACK\Audit997.mms
; Executable Map: ACK997
;
D:\MapsAndTrees\ANSIACK\ACK997.mmc
    -OF1BR10:2 'OutboundACK.EDI'
;
; End System: Inbound EDI
;
```

Notice that the execution settings defined in an Integration Flow Designer command file correspond with the execution commands and options. See the *Integration Flow Designer* documentation for more information.

Command files

You can create command files that contain listings of maps to execute with execution commands that specify how you want each map to run. A command file is a text file that can contain multiple maps to run, as well as multiple execution commands and options for each map; however, a command file cannot run another command file.

Using Command Server commands

When you run a map, you can:

- Override sources and targets for the cards used in the map
- Create trace and audit log files
- Control a variety of characteristics of the run-time environment

The execution commands entered when you run a map do not permanently affect the compiled map file.

The command options that you specify override the settings compiled into the map file. If you do not use a specific execution command, the corresponding map and card settings compile into the map.

- [General rules for execution commands](#)
- [From a command line](#)
- [From the Platform API](#)
- [From a RUN\(\) function in a map rule](#)

General rules for execution commands

The general rules that apply when specifying execution commands are:

- Each execution command must begin with a hyphen.
- At least one space is required between commands (for example, `-T -v`).
- There are no spaces within an option except for the commands that allow you to override sources and targets (for example, `-I` and `-O`), which require a space before the source or target name.
- If a source or target name contains a space, enclose the entire name in single quotes.
- Execution commands and options are not case-sensitive.
- If spaces or single quotes can occur in echoed data specified using the `-IE` execution command, the size option should be used (`-IExs`).
- Multiple options can be specified for a single map.
- Command files can contain multiple lines. Line breaks can be used to replace of the required space that must exist between commands. The line breaks are converted to spaces before to processing.

From a command line

You can execute a particular map or a command file containing a list of maps to execute from the command line of the system hosting your Command Server. The syntax is:

```
DTXCMDSV {map [exec_command...] | @command_file}...
```

Ensure your current directory is where the Command Server is located, or specify the relative path or full path, as applicable, when entering the command.

Option

Description

DTXCMDSV

Specifies the name of the Command Server execution command for both UNIX and Windows platforms.

map Specifies the name of the compiled map (.mmc) file to run. If applicable, include the full path or a relative path to the directory that contains the map file.

exec_command Specifies any execution command and its associated options that you want to use to override the settings compiled into the map file when the specified map is run.

command_file Specifies the name of a command file that contains a list of maps to execute. If applicable, include the full path or a relative path to the directory that contains the command file.

- [Command line help](#)

Command line help

You can access Help from the command line by typing:

```
DTXCMDSV -?
```

From the Platform API

To run the server using a Platform API, as part of an argument to the **RunMap()** function, specify:

```
map [exec_command...] |@command_file
```

Option	Description
<i>map</i>	Specify the name of the compiled map (.mmc) file to run. If applicable, include the full path or a relative path to the directory that contains the map file.
<i>exec_command</i>	Specify any execution command and its associated options you want to override the settings compiled into the map file when the specified map is run.
<i>command_file</i>	Specify the name of a command file that contains a list of maps to execute. If applicable, include the full path or a relative path to the directory that contains the command file.

From a RUN() function in a map rule

To execute a map from a map rule, use the **RUN()** function as follows:

```
RUN ( "map" , "exec_command..." ] )
```

Option	Description
<i>map</i>	Specify the name of the compiled map (.mmc) file to run. If applicable, include the full path or a relative path to the directory that contains the map file.
<i>exec_command</i>	Specify any execution command and its associated options you want to override the settings compiled into the map file when the specified map is run.

Execution commands

This documentation contains information about the execution commands and associated options, including the syntax of each command and its particular usage.

- [Command availability](#)
- [MapAudit \(-A\)](#)
- [Close Window \(-B\)](#)
- [Century \(-D\)](#)
- [Fail on Warnings \(-F\)](#)
- [Ignore Validation Options \(-G\)](#)
- [Input Source Override - Application \(-IA\)](#)
- [Input Source Override - Database \(-ID\)](#)
- [Input Source Override - Echo \(-IE\)](#)
- [Input Source Override - File \(-IF\)](#)
- [Input Source Override - Message \(-IM\)](#)
- [Input Source Override - Pointer \(-IP\)](#)
- [List \(-L\)](#)
- [Initialization File Override \(-N\)](#)
- [Output Target Override - Application \(-OA\)](#)
- [Output Target Override - Database \(-OD\)](#)
- [Output Target Override - Echo \(-OE\)](#)
- [Output Target Override - File \(-OF\)](#)
- [Output Target Override - Message \(-OM\)](#)
- [Output Target Override - Pointer \(-OP\)](#)

- [WorkSpace PageSize \(-P\)](#)
- [Refresh Status \(-R\)](#)
- [MapTrace \(-T\)](#)
- [Stop Validation \(-V\)](#)
- [WorkSpace \(-W\)](#)
- [Retry \(-Y\)](#)
- [Ignore Warnings \(-Z\)](#)

Command availability

Some commands are limited because they can only be used with a Command Server and cannot be used from within a `RUN()` function or using the Platform API. Also, there are some commands that are not available on all platforms. The Integration Flow Designer, the Map Designer, and the Command Server have a user interface for these commands where they apply.

The following table lists the execution commands and where they can be executed (Y), as well as those commands that have limited platform availability.

Execution Command	Command Server	RUN Function	Platform API	Platform Availability
MapAudit (-A)	Y	Y	Y	
Close Window (-B)	Y			Windows only
Century (-D)	Y	Y	Y	
Fail on Warnings (-E)	Y	Y	Y	
Ignore Validation (-G)	Y	Y	Y	
Application (-IA)	Y	Y	Y	
Database (-ID)	Y	Y	Y	
Echo (-IE)	Y	Y	Y	
File (-IF)	Y	Y	Y	
Message (-IM)	Y	Y	Y	
List Sources/Targets (-L)	Y			Non-Windows only
Initialization File Override (-N)	Y			
Application (-OA)	Y	Y	Y	
Database (-OD)	Y	Y	Y	
Echo (-OE)	Y*	Y	Y	
File (-OF)	Y	Y	Y	
Message (-OM)	Y	Y	Y	
WorkSpace PageSize (-P)	Y	Y	Y	
Refresh Status (-R)	Y			
MapTrace (-T)	Y	Y	Y	
Stop Validation (-V)	Y	Y	Y	
WorkSpace (-W)	Y	Y	Y	
Retry (-Y)	Y	Y	Y	
Ignore Warnings (-Z)	Y	Y	Y	

MapAudit (-A)

Use the MapAudit execution command (-A) to control the creation of audit log information. Using this command, audit information is recorded which can include data audit information and execution audit information.

When used *with* other specified options, the **MapAudit** settings specified with the **MapAudit** command (-A) override *all* **MapAudit** settings compiled into the map. When used *without* other specified options, no audit log is produced.

```
-A[D[R|W]{1|2}][E[R|W]][B[R|W]][C[R|W]][P[R|W]]
[M[S] | [U] [+!+][={file|dir}]]
```

For information about the audit log files, see the *Map Designer* documentation.

Option

Description

D

Include data audit information in the audit log. The recorded information is based on the **MapAudit** settings configured in the Map Designer for a particular map. The sub-options for D include: R, W, and 1 or 2

R

Generate audit files each time the map is run only if the outcome of the map's execution is an error condition. Used with the burst audit data information (D), execution summary audit information (E), burst execution information (B), card (data) settings (C), and map settings (P) options.

W

Generate audit files each time the map is run only if the outcome of the map's execution is a warning or an error condition. Used with the burst audit data information (D), execution summary audit information (E), burst execution information (B), card (data) settings (C), and map settings (P) options.

1|2

1 is the audit code for fail size (**WrongSize**).

2 is the audit code for fail minimum size or fail maximum size (**TooLongTooShort**).

Note If using the 1 or 2 option as well as the R or W option with the D sub-option, the 1 or 2 must follow the R or W. For example:

-AEDW2

This example represents SummaryAudit Execution, SummaryAudit Data, OnWarning, and TooLongTooShort.

B

The burst execution information includes:

- the burst return value
- the time it took for the burst to execute
- the number of bytes, the adapter return value, and the content return value for each card

Example:

```
<ExecutionLog burstreturn="0" ElapsedSec="0.6846">
  <inputstatus card="1" bytes="4904" adapterreturn="0" contentreturn="0"/>
  <inputstatus card="2" bytes="270" adapterreturn="0" contentreturn="0"/>
  <outputstatus card="1" bytes="18695" adapterreturn="0" contentreturn="0"/>
</ExecutionLog>
```

E

The execution summary audit information includes:

- return codes resulting from the map execution
- execution time (in seconds)
- command line used to execute the map
- number of input objects found and output objects built (Command Server only)
- details about the map sources and destinations
- work file names and sizes

C

The card (data) settings audit information includes:

- the data settings for sources and target cards

P

The map settings audit information includes:

- the map settings

M

The memory setting can be used only when the map is run by the `RUN()` function or an API. If memory is set, audit information is stored in memory and appended to the last set of data returned during map execution.

When the map is run by a Command Server or Launcher, this setting is ignored and the audit log is created in a file in the map directory, using the `<mapname>.log` naming convention.

If the audit in memory fails or there is insufficient available memory, a `Disk write error or Not enough memory to execute map` message can be returned.

S

Only use with the memory option (M). This option prefixes the audit data with a text number indicating the size (in bytes) of the audited data followed by a space. This option causes the resulting audit log entry to be returned as an ASCII number followed by a space, then followed by the specified number of bytes of data. This helps to distinguish it from multiple character strings in the returned data when at least one output and the audit log are echoed.

For example, if used in a `RUN()` function:

```
RUN ("mymap.mmc","-OE3S -ADEMS")
```

If the data resulting from output card #3 was This and That, the string returned by this `RUN()` function is:

```
13 This and That1031 BEGIN RUN Function for Windows....
```

U

Generate audit files with unique names each time the map is run. The name of the audit log for top-level maps is:

```
mer_mapname_processkey_mapcounter_hostname.bak
```

where `processkey` is a combination of the process id from the operating system and the time the process started, `mapcounter` is a value representing each unique instance of a map within the same process, and `hostname` is the host name of the computer where Transformation Extender is running.

Run maps use:

```
run_mapname_processkey_mapcounter_hostname.bak
```

Unique names are globally unique; each process has a unique value.

Unique log file names are typically used in a multi-threaded environment so that more than one instance of the same map can run at the same time. In a command environment, unique log file names are typically used when more than one command specifies the same map to run and you do not want one log file to overwrite the other. Or, use this option when multiple servers run the same map. This option prevents one log file from overwriting another log file.

This option cannot be used with the append option (+).

+

Each time the map is run, append specified information to the audit log. If no audit log exists at the start of map execution, an audit log is created. Turn off the + option. Discontinue appending specified information to the audit log. This option cannot be used with the U option for unique names.

!+

Turn off the **+ option**. Discontinue appending specified information to the audit log. This option cannot be used with the **U** option for unique names.
=file/dir
Specify a file name for appending audit information or specify a full directory path and file name in which you want the audit log file created. By default, the audit log file is created in the same directory as the compiled map (.mmc) file, using the <mapname>.bak naming convention.

- [Examples using MapAudit \(-A\)](#)

Examples using MapAudit (-A)

```
-AD
    BurstAudit Data
-AD1
    BurstAudit Data, WrongSize
-AD2
    BurstAudit Data, TooLongTooShort
-AB
    BurstAudit Execution
-AE
    SummaryAudit Execution
-AC
    SettingsAudit Data
-AP
    SettingsAudit Map
-ABD
    BurstAudit Execution and BurstAudit Data
-AEB
    SummaryAudit Execution and BurstAudit Execution
-ABDE
    BurstAudit Execution, BurstAudit Data, and SummaryAudit Execution
-AEDW2
    SummaryAudit Execution, SummaryAudit Data, OnWarning, and TooLongTooShort
-AER
    SummaryAudit Execution and OnError
-ADW
    BurstAudit Data and OnWarningOrError
AERCWPW
    SummaryAudit Execution and OnError and SettingsAudit Data and OnWarningOrError and SettingsAudit Map and OnWarningOrError
-ADR2BW=c:\install_dir\mymap.log
    SummaryAudit Execution, and SummaryAudit Data, OnError, TooLongTooShort, and BurstAudit Data, OnWarningOrError written to c:\install_dir\mymap.log
-ADEU=/install_dir
    AuditData and AuditExecution using the unique filename mechanism and writing the log file to the install directory
-ADER+=mymap.log
    AuditData and AuditExecution. If the map execution failed, the data audit information (AuditData) and execution summary and audit information (AuditExecution) content is appended to the mymap.log file that is in the map directory.
-AERM
    AuditExecutionSummary in memory, only if map OnExecutionFailure
```

Close Window (-B)

When executing a map using a Command Server for Windows or AS/400 OS/400®, use the Close Window execution command (-B) to exit the Command Server application after the map has completed.

-B [I]

Option	Description
I	When using a Command Server for Windows, specify to exit the Command Server application only when the map completes successfully.

Century (-D)

Use the Century execution command (-D) to specify the start of a 100-year range used to supply a missing century portion for a Date & Time. This command is useful for the transition to the year 2000 so that dates with a YYMMDD format can be interpreted properly for the correct century.

-D [ccyy|0]

Option

Description

ccyy	A missing century for a Date & Time item is derived based on the ccyy value.
0	Turn off the Century execution command.

For example, if you specify **-D1960**, a YYMMDD-formatted date between 60mmdd and 99mmdd is interpreted as having a century value of 19. Dates between 00mmdd and 59mmdd are interpreted as having a century value of 20. In this example, data with a date of 580101 is assigned a century of 20; however, data with a date of 600101 is assigned a century of 19.

Fail on Warnings (-F)

Use the Fail on Warnings (-F) execution command to specify that the map should fail when any of the selected warning codes are returned. When used *with* one or more warning codes, a map-failed condition is returned for those warning codes. When used *without* other options specified, all warning codes result in a map-failed condition. Using this command impacts **OnSuccess** settings, **OnFailure** settings, and so on.

-F[!][warning_code[:warning_code...]]

Option

Description

!

Use to specify warning codes that remain in their default states, that is, as warnings, when they are detected, and as a result, do not cause the mapping process to return a map failed message.

When used with **-F** only, for example, **-F!**, a map-failed condition is returned if any of the warning codes is detected.

warning_code

Specify the warning codes that the mapping process needs to detect, and, depending on how the other Fail on Warnings command options are set, use to determine if it should return a map failed message. Valid warning code selections include: 14, 18, 21, 26, 27, 28, 29, and 34.

- [Examples using Fail on Warnings \(-F\)](#)

Examples using Fail on Warnings (-F)

-F

A map-failed message is returned when any of the warning codes is encountered.

-F!

A map-failed message is returned when any of the warning codes is encountered.

-F18

A map-failed message is returned when a Page Usage Count Error (18) is encountered.

-F!18

Excluding the Page Usage Count Error (18), a map-failed message is returned for all of the other warning codes.

-F18:27:28

A map-failed message is returned when any of the specified warning codes is encountered (18, 27, or 28 in this case).

-F34

A map-failed message is returned when a Too Few Pages Requested, More Allocated Error (34) is encountered.

If the **-F** command is used and the map fails due to the cause associated with the specified warning, the execution audit includes an Error map status with the appropriate warning code preceded by a minus sign and the map-failed message.

For example:

```
<Summary MapStatus="Error" Return="-28">
  <Elapsed TimeInSec>0.0898</Elapsed TimeInSec>
  <Message>Input type contains errors.</Message>
  <CommandLine>'install_dir\example\ansiack\receiveEDI.mmc'
  </CommandLine>
  <ObjectsFound>0</ObjectsFound>
  <ObjectsBuilt>0</ObjectsBuilt>
</Summary>
```

When using this command, you cannot specify the same warning code using the **-Z** command. If you do, the setting of the latter command in the command line is used. For example, if your command line included **-F18:26 -Z18:27:28**, a return code of 18 (Page Use Count error) would be ignored because the Ignore Warnings command (**-Z**) follows the Fail on Warnings command (**-F**).

A map-failed message is returned when any of the warning codes is encountered for both the **-F** and **-F!** commands.

Ignore Validation Options (-G)

Use the Ignore Validation Options execution command (-G) to specify the item properties, if any that should be ignored during validation. When used *with* other specified options, only those item properties are ignored. When used *without* other specified options, no item properties are ignored.

-G[R][P][S][T][C]

Option

Description

R

Ignore restrictions for all items appearing in the input data.

Restrictions of syntax items are not ignored because they are used in determining the value of a syntax object.

P

Ignore the presentation settings of items.

For example, if a number is defined as an integer but it appears in the data as a decimal number, it validates anyway.

If the Ignore command is used and an item has an invalid presentation, it is mapped as if it were NONE. However, if the input is converted to text using the TEXT function, it is mapped.

S

Ignore the minimum and maximum size of items that appear in delimited data.

For example, if an item is defined as having a minimum of three bytes, but the data contains only two bytes, it validates anyway.

T

Ignore restrictions for all items appearing in the input data with no warnings.

C

Prevents a component rule from being evaluated when either:

- The group does not include a component that has a group identifier attribute.
- The component rule occurs on a component that is after the component that has the group identifier attribute.

Use this option to run a map without validating component rules, for example, when your data does not conform exactly to a particular type tree.

Input Source Override - Application (-IA)

Use the Input Source Override - Application execution command (-IA) to override specifications in the compiled map file for a specific input card for a single execution of a map. Only those adapter settings specified are used to override the settings compiled into the map.

For example, if the WorkArea option (w or !w) is not specified, the **WorkArea** setting compiled into the map is used.

```
-IA[alias]card_num[M['filename']]  
[W|!W][Rcount:interval][B][EN][EA][EF][ES][EW]  
[AF][AI][PM][PB][PC][F'n'][U][!U]  
[K['filename']]|[+'filename']|[U'filepath']|[U]|  
[1'filename']|[+filename']|[U1'filepath']|[U1]  
[library],[function],[application_command_line]|  
{src-adptr-cmd|..}
```

Option

Description

alias

This is the adapter alias for the specific application adapter. For example, **FTP** is the adapter alias for the FTP adapters. If the *alias* command option is not used, you must use *library*, *function*, and *application_command_line*. If *alias* is used, you must use *src-adptr-cmd* or *".*. See the appropriate adapter documentation in the *Resource Adapters* documentation.

card_num

This is the card number of the input to override.

M'filename'

This is the metadata (XML schema or DTD) file location of the input card to override.

Specify *filename* as an absolute path along with the file name and enclosed in single quotes, for the metadata when parsing input XML. An example is:

```
M'k:\my_project_folder\myschema.xml'
```

Use this command option to override the Metadata location setting for the root type specified on an input card in the Map Designer GUI.

W

After a map runs for the first time, the work area created for the input card is not deleted. For subsequent executions of the same map, the data for this input card is not validated and the work area information for the card is retrieved from the existing work area.

!W

The work area for input is created when map execution begins.

Rcount:interval

Specify **Retry** settings. If the source is unavailable, the adapter will attempt to access the source as many times as specified by the *count* setting at the interval specified by the *interval* setting.

count number of attempts to access the application

interval number of seconds to wait between attempts

Note: If this option is not specified, the **Retry** settings compiled into the map are used. To eliminate adapter retry, specify **RO:0**.

B

If the map, burst, or card does not successfully complete, roll back any changes that were made to this data source.

Note: If this option is not specified, the **OnFailure** setting compiled into the map is used.

EN

Use this command option to never call the external parser for document verification.

EA

Use this command option to always call the external parser for document verification.

EF

Use this command option to call the external parser for document verification only if validation fails.

ES

Use this command option to call the external parser for document verification only if validation is successful.

EW

Use this command option to call the external parser for document verification only for the well-formed document.

AF

Use this command option to fail on adapter warnings.

AI

Use this command option to ignore adapter warnings.

PM

Use this command option to apply the OnSuccess and OnFailure settings based on the success or failure of the map execution.

PB Use this command option to apply the OnSuccess and OnFailure settings based on the success or failure of each burst.

PC Use this command option to apply the OnSuccess and OnFailure settings based on the success or failure of the input card processing

Fn FetchUnit defines the number of units of data to retrieve. The default value for FetchUnit is S (unspecified all).

U Use this command option to apply the burst setting.

IU Use this command option to not burst (integral mode).

K When the map, burst, or card runs, no backup file will be created.

K'filename' When the map, burst, or card runs, a backup file will always be created to *filename*.
Note: If '*filename*' is not an absolute path, the map directory will be appended as a prefix to it.

K+'filename' When the map, burst, or card runs, a backup file will always be appended to *filename*.
Note: If '*filename*' is not an absolute path, the map directory will be appended as a prefix to it.

KU'filepath' When the map, burst, or card runs, a backup file will always be created using a unique backup name in the directory *filepath*.
Note: If '*filepath*' is not an absolute path, the map directory will be appended as a prefix to it.

KU When the map, burst, or card runs, a backup file will always be created using a unique backup name in the map directory.

K1'filename' If the map, burst, or card does not successfully complete, a backup file will be created to *filename*.
Note: If '*filename*' is not an absolute path, the map directory will be appended as a prefix to it.

K1+'filename' If the map, burst, or card does not successfully complete, a backup file will be appended to *filename*.
Note: If '*filename*' is not an absolute path, the map directory will be appended as a prefix to it.

KU1'filepath' If the map, burst, or card does not successfully complete, a backup file will be created using a unique backup name in the directory *filepath*.
Note: If '*filepath*' is not an absolute path, the map directory will be appended as a prefix to it.

KU1 If the map, burst, or card does not successfully complete, a backup file will be created using a unique backup name in the map directory.

library This command option specifies the user library name for the adapters when using an application adapter as a source. The user library name should be followed by the function name and application command line, separated by commas. This option, along with the function name and application command line, must be used when *alias* is not specified.

function This command option specifies the function name for the adapters when using an application adapter as a source and should be prefaced by the user library name and followed by the application command line, separated by commas.

application_command_line This command option specifies the application command line for the adapters when using an application adapter as a source and should be prefaced by the user library name and the function name, separated by commas.

src-adptr-cmd This command option provides the adapter-specific commands to connect to the data source and retrieve the input data. Either this option or ":" must be used if the *alias* option is specified. See the adapter-specific information in the *Resource Adapters* documentation.

Use **Command** in the compiled map file. Either this option or **src-adptr-cmd** must be used if the *alias* option is specified.

Input Source Override - Database (-ID)

Use the Input Source Override - Database execution command (-ID) to override specifications in the compiled map file for a specific input card for a single execution of a map. Only those adapter settings specified are used to override the settings compiled into the map.

For example, if the WorkArea option (W or !W) is not specified, the **WorkArea** setting compiled into the map is used.

```
-IDcard_num[M['filename']]
[W|!W][Rcount:interval][B][EN][EA][EF][ES][EW]
[AF][AI][PM][PB][PC][F'n'][U][!U]
[K]['filename']|[+'filename']|[U'filepath']|[U]
[1'filename']|[1+'filename']|[U1'filepath']|[U1]
{src-adptr-cmd}|.
```

Option

Description

card_num

This is the card number of the input to override.

M'filename'

This is the metadata (XML schema or DTD) file location of the input card to override.

Specify *filename* as an absolute path along with the file name and enclosed in single quotes, for the metadata when parsing input XML. An example is:

```
M'k:\my_project_folder\myschema.xml'
```

Use this command option to override the Metadata location setting for the root type specified on an input card in the Map Designer GUI.

W

After a map runs for the first time, the work area that is created for the input card is not deleted. Then, on subsequent executions of the same map, the data for this input card is not validated and the work area information for the card is retrieved from the existing work area.

IW

Input's work area is created when map execution begins.

Rcount:interval

Specify **Retry** settings. If the source is unavailable, the adapter attempts to access the source as many times as specified by the *count* setting at the interval specified by the *interval* setting.

count number of attempts to access the database

interval number of seconds to wait between attempts.

If this option is not specified, the **Retry** settings compiled into the map are used. To eliminate adapter retry, specify **R0:0**.

B

If the map, burst, or card does not complete successfully, roll back any changes that were made to this data source. If this option is not specified, the **OnFailure** setting compiled into the map is used.

EN

Use this command option to never call the external parser for document verification.

EA

Use this command option to always call the external parser for document verification.

EF

Use this command option to call the external parser for document verification only if validation fails.

ES

Use this command option to call the external parser for document verification only if validation is successful.

EW

Use this command option to call the external parser for document verification only for the well-formed document.

AF

Use this command option to fail on adapter warnings.

AI

Use this command option to ignore adapter warnings.

PM

Use this command option to apply the OnSuccess and OnFailure settings based on the success or failure of the map execution.

PB

Use this command option to apply the OnSuccess and OnFailure settings based on the success or failure of each burst.

PC

Use this command option to apply the OnSuccess and OnFailure settings based on the success or failure of the input card processing

Fn

FetchUnit defines the number of units of data to retrieve. The default value for FetchUnit is S (unspecified all).

U

Use this command option to apply the burst setting.

!U

Use this command option to not burst (integral mode).

K

When the map, burst, or card runs, no backup file will be created.

K'filename'

When the map, burst, or card runs, a backup file will always be created to *filename*. If *filename*' is not an absolute path, the map directory will be appended as a prefix to it.

K+'filename'

When the map, burst, or card runs, a backup file will always be appended to *filename*. If *filename*' is not an absolute path, the map directory will be appended as a prefix to it.

KU'filepath'

When the map, burst, or card runs, a backup file will always be created using a unique backup name in the directory *filepath*. If '*filepath*' is not an absolute path, the map directory will be appended as a prefix to it.

KU

When the map, burst, or card runs, a backup file will always be created using a unique backup name in the map directory.

K1'filename'

If the map, burst, or card does not successfully complete, a backup file will be created to *filename*. If '*filename*' is not an absolute path, the map directory will be appended as a prefix to it.

K1+'filename'

If the map, burst, or card does not successfully complete, a backup file will be appended to *filename*. If '*filename*' is not an absolute path, the map directory will be appended as a prefix to it.

KU1'filepath'

If the map, burst, or card does not successfully complete, a backup file will be created using a unique backup name in the directory *filepath*. If '*filepath*' is not an absolute path, the map directory will be appended as a prefix to it.

KU1

If the map, burst, or card does not successfully complete, a backup file will be created using a unique backup name in the map directory.

src-adptr-cmd

Specify the settings for the source using the adapter-specific commands. See the specific database adapter documentation in the *Resource Adapters* documentation.

Use **Command** in the compiled map file.

Input Source Override - Echo (-IE)

Use the Input Source Override - Echo execution command (-IE) to override specifications in the compiled map file for a specific input card for a single execution of a map.

-IEcard_num[M['filename']]

[Ssize] [W|!W] [EN] [EA] [EF] [ES] [EW] [AF] [AI] [PM] [PB] [PC] [F' n'] [U] [|U] source

Option

Description

card_num

This is the card number of the input to override.

M'filename'

This is the metadata (XML schema or DTD) file location of the input card to override.

Specify *filename* as an absolute path along with the file name and enclosed in single quotes, for the metadata when parsing input XML. An example is:

M'k:\my_project_folder\myschema.xml'

Use this command option to override the Metadata location setting for the root type specified on an input card in the Map Designer GUI.

Ssize

This is the size of the data specified by *source* in bytes.

W

After a map runs for the first time, the WorkArea created for the input card is not deleted. On subsequent executions of the same map, the data for this input card is not validated and the WorkArea information for the card is retrieved from the existing WorkArea.

!W

The WorkArea for input is created when map execution begins.

EN

Use this command option to never call the external parser for document verification.

EA

Use this command option to always call the external parser for document verification.

EF

Use this command option to call the external parser for document verification only if validation fails.

ES

Use this command option to call the external parser for document verification only if validation is successful.

EW

Use this command option to call the external parser for document verification only for the well-formed document.

AF

Use this command option to fail on adapter warnings.

AI

Use this command option to ignore adapter warnings.

PM

Use this command option to apply the OnSuccess and OnFailure settings based on the success or failure of the map execution.

PB

Use this command option to apply the OnSuccess and OnFailure settings based on the success or failure of each burst.

PC

Use this command option to apply the OnSuccess and OnFailure settings based on the success or failure of the input card processing

Fn

FetchUnit defines the number of units of data to retrieve. The default value for FetchUnit is S (unspecified all).

U

Use this command option to apply the burst setting.

!U

Use this command option to not burst (integral mode).

source

This is the data to be used for the input.

For example, to pass the value ABC123XYZ to a map as the data for input card #2, type the following command:

-IE2 ABC123XYZ

If the data to be echoed contains spaces or hyphens, it must be surrounded by single quotes as shown in the following example:

-IE2 `ABC 123 XYZ`

If the echoed data contains single quotes or the null character (hex 00), use the **S** option to specify the number of bytes of echoed data. Changing the example above to use to the **S** option, the single quotes are omitted:

-IE1S11 ABC 123 XYZ

The examples above demonstrate using the Input Source Override - Echo execution command (-IE) in a map rule.

If you are using the Input Source Override - Echo execution command (-IE) in the Command Server command line and the data contains spaces or hyphens, in addition to surrounding the data to be echoed with single quotes, you must then add double-quotes outside the single-quotes as shown in the following example:

-IE2 " `ABC 123 XYZ` "

Also, if you are using the Input Source Override - Echo execution command (-IE) in the Command Server command line and the data contains single quotes or the null character (hex 00), in addition to using the **S** option to specify the number of bytes of echoed data, and omitting the single quotes, if the data contained them, you must also surround the data with double-quotes as shown in the following example:

-IE1S11 "ABC 123 XYZ"

One use for echoed input data is in conjunction with the **RUN** function to pass data from one map to another. For an example of using an echoed data source as an input argument using the **RUN** function, see the **RUN** function in the *Functions and Expressions* documentation. Another example of using an echoed data source is provided in the *Platform API* documentation.

Input Source Override - File (-IF)

Use the Input Source Override - File execution command (**-IF**) to override specifications in the compiled map file for a specific input card for a single execution of a map. Only the specified adapter settings are used to override the settings compiled into the map.

For example, if the WorkArea option (**w** or **!w**) is not specified, the **WorkArea** setting compiled into the map is used.

```
-IFcard_num[M['filename']]  
[W|IW][X|X0][Rcount:interval][B][EN][EA][EF][ES][EW]  
[AF][AI][PM][PB][PC][F'n'][U][!U]  
[K['filename']]|[+'filename']|[U'filepath']|[U]|  
[1'filename']|[1+'filename']|[U1'filepath']|[U1]|  
{source|..}
```

Option

Option	Description
<i>card_num</i>	This is the card number of the input to override.
<i>M'filename'</i>	This is the metadata (XML schema or DTD) file location of the input card to override. Specify <i>filename</i> as an absolute path along with the file name and enclosed in single quotes, for the metadata when parsing input XML. An example is: M'k:\my_project_folder\myschema.xml'
<i>W</i>	Use this command option to override the Metadata location setting for the root type specified on an input card in the Map Designer GUI.
<i>IW</i>	After a map runs the first time, the WorkArea created for the input card is not deleted. For subsequent executions of the same map, the data for this input card is not validated and the WorkArea information for the card is retrieved from the existing WorkArea.
<i>X</i>	The WorkArea of the input is created when map execution begins.
<i>X0</i>	If the map, burst, or card completes successfully, the input data file is deleted.
<i>Rcount:interval</i>	Specify Retry settings. If the source is unavailable, the adapter attempts to access the source as many times as specified by the <i>count</i> setting at the interval specified by the <i>interval</i> setting. <i>count</i> number of attempts to access the data file <i>interval</i> number of seconds to wait between attempts . If this option is not specified, the Retry settings compiled into the map are used. To eliminate adapter retry, specify RO:0 .
<i>B</i>	If the map, burst, or card does not complete successfully, roll back any changes made to this data source. If this option is not specified, the OnFailure setting compiled into the map is used.
<i>EN</i>	Use this command option to never call the external parser for document verification.
<i>EA</i>	Use this command option to always call the external parser for document verification.
<i>EF</i>	Use this command option to call the external parser for document verification only if validation fails.
<i>ES</i>	Use this command option to call the external parser for document verification only if validation is successful.
<i>EW</i>	Use this command option to call the external parser for document verification only for the well-formed document.
<i>AF</i>	Use this command option to fail on adapter warnings.
<i>AI</i>	Use this command option to ignore adapter warnings.
<i>PM</i>	Use this command option to apply the OnSuccess and OnFailure settings based on the success or failure of the map execution.
<i>PB</i>	Use this command option to apply the OnSuccess and OnFailure settings based on the success or failure of each burst.
<i>PC</i>	Use this command option to apply the OnSuccess and OnFailure settings based on the success or failure of the input card processing
<i>Fn</i>	FetchUnit defines the number of units of data to retrieve. The default value for FetchUnit is S (unspecified all).
<i>U</i>	Use this command option to apply the burst setting.
<i>IU</i>	Use this command option to not burst (integral mode).
<i>K</i>	When the map, burst, or card runs, no backup file will be created.
<i>K'filename'</i>	When the map, burst, or card runs, a backup file will always be created to <i>filename</i> . If ' <i>filename</i> ' is not an absolute path, the map directory will be appended as a prefix to it.
<i>K+'filename'</i>	

When the map, burst, or card runs, a backup file will always be appended to *filename*. If '*filename*' is not an absolute path, the map directory will be appended as a prefix to it.

KU'filepath'

When the map, burst, or card runs, a backup file will always be created using a unique backup name in the directory *filepath*. If '*filepath*' is not an absolute path, the map directory will be appended as a prefix to it.

KU

When the map, burst, or card runs, a backup file will always be created using a unique backup name in the map directory.

K1'filename'

If the map, burst, or card does not successfully complete, a backup file will be created to *filename*. If '*filename*' is not an absolute path, the map directory will be appended as a prefix to it.

K1+'filename'

If the map, burst, or card does not successfully complete, a backup file will be appended to *filename*. If '*filename*' is not an absolute path, the map directory will be appended as a prefix to it.

KU1'filepath'

If the map, burst, or card does not successfully complete, a backup file will be created using a unique backup name in the directory *filepath*. If '*filepath*' is not an absolute path, the map directory will be appended as a prefix to it.

KU1

If the map, burst, or card does not successfully complete, a backup file will be created using a unique backup name in the map directory.

source

Specify the name of the input file. If the input file is not located in the same directory as the map file, the full path must be specified using the platform-specific syntax.

Use the **FilePath** in the compiled map file.

Input Source Override - Message (-IM)

Use the Input Source Override - Message execution command (-IM) to override specifications in the compiled map file for a specific input card for a single execution of a map. Only the specified adapter settings are used to override the settings compiled into the map.

For example, if the WorkArea option (w or !w) is not specified, the **WorkArea** setting compiled into the map is used.

```
-IM[alias]card_num[M['filename']]  
[W|!W][X|X0][Rcount:interval][B][EN][EA][EF][ES][EW]  
[AF][AI][PM][PB][PC][F'n'][U][!U]  
[K[filename]]|[+'filename'][U'filepath'][U]  
[1'filename']|[1+'filename'][U1'filepath'][U1]  
{src-adptr-cmd| .}
```

Option

Description

alias

This is the adapter alias for the specific messaging adapter. For example, AQ is the adapter alias for the Oracle AQ adapters. See the adapter-specific information in the *Resource Adapters* documentation. For operating system and version-specific information, see the [system requirements](#) and the [release notes](#).

card_num

This is the card number of the input to override.

M'filename'

This is the metadata (XML schema or DTD) file location of the input card to override.

Specify *filename* as an absolute path along with the file name and enclosed in single quotes, for the metadata when parsing input XML. An example is:

```
M'k:\my_project_folder\myschema.xml'
```

Use this command option to override the Metadata location setting for the root type specified on an input card in the Map Designer GUI.

W

After a map runs the first time, the WorkArea created for the input card is not deleted. For subsequent executions of the same map, the data for this input card is not validated and the WorkArea information for the card is retrieved from the existing WorkArea.

!W

The WorkArea of input is created when map execution begins.

X

If the map, burst, or card completes successfully, the message is deleted from its source.

X0

If the map, burst, or card completes successfully, the message is deleted from its source, only if it has no content. If neither delete option (x or X0) is specified, the **OnSuccess** setting compiled into the map is used.

Rcount:interval

Specify **Retry** settings. If the source is unavailable, the adapter attempts to access the source as many times as specified by the *count* setting at the interval specified by the *interval* setting.
count number of attempts to access the message queue.

interval number of seconds to wait between attempts. If this option is not specified, the **Retry** settings compiled into the map are used. To eliminate adapter retry, specify **R0:0**.

B

If the map, burst, or card does not complete successfully, roll back any changes made to this data source. If this option is not specified, the **OnFailure** setting compiled into the map is used.

EN

Use this command option to never call the external parser for document verification.

EA

Use this command option to always call the external parser for document verification.

EF

Use this command option to call the external parser for document verification only if validation fails.

ES Use this command option to call the external parser for document verification only if validation is successful.
EW Use this command option to call the external parser for document verification only for the well-formed document.
AF Use this command option to fail on adapter warnings.
AI Use this command option to ignore adapter warnings.
PM Use this command option to apply the OnSuccess and OnFailure settings based on the success or failure of the map execution.
PB Use this command option to apply the OnSuccess and OnFailure settings based on the success or failure of each burst.
PC Use this command option to apply the OnSuccess and OnFailure settings based on the success or failure of the input card processing
Fn FetchUnit defines the number of units of data to retrieve. The default value for FetchUnit is S (unspecified all).
U Use this command option to apply the burst setting.
!U Use this command option to not burst (integral mode).
K When the map, burst, or card runs, no backup file will be created.
K'filename' When the map, burst, or card runs, a backup file will always be created to *filename*. If '*filename*' is not an absolute path, the map directory will be appended as a prefix to it.
K+'filename' When the map, burst, or card runs, a backup file will always be appended to *filename*. If '*filename*' is not an absolute path, the map directory will be appended as a prefix to it.
KU'filepath' When the map, burst, or card runs, a backup file will always be created using a unique backup name in the directory *filepath*. If '*filepath*' is not an absolute path, the map directory will be appended as a prefix to it.
KU When the map, burst, or card runs, a backup file will always be created using a unique backup name in the map directory.
K1'filename' If the map, burst, or card does not successfully complete, a backup file will be created to *filename*. If '*filename*' is not an absolute path, the map directory will be appended as a prefix to it.
K1+'filename' If the map, burst, or card does not successfully complete, a backup file will be appended to *filename*. If '*filename*' is not an absolute path, the map directory will be appended as a prefix to it.
KU1'filepath' If the map, burst, or card does not successfully complete, a backup file will be created using a unique backup name in the directory *filepath*. If '*filepath*' is not an absolute path, the map directory will be appended as a prefix to it.
KU1 If the map, burst, or card does not successfully complete, a backup file will be created using a unique backup name in the map directory.
src-adptr-cmd Specify settings for the source using the adapter-specific commands. See the adapter-specific information in the *Resource Adapters* documentation.

Use **Command** in the compiled map file.

-IMALE2F2AFUPBXOR10:5BK'ale.bak'<ale_commands>

In this example, Input 2 is ALE **-IMALE2**. Use 2 as a fetch unit, specified as **F2**. **F** is the **Fetch** unit command option and 2 is the actual fetch unit. Use fail on adapter warnings **AF**, burst **U**, adapter scope is burst **PB**, delete if no content **X0**, retry 10 times every 5 seconds **R10:5**, **B** is the Rollback if failure command option. When **B** is specified, at map run time, if the map, burst, or card does not complete successfully, any changes made to this data source are rolled back and backup is written to the file ale.bak.

B

Note: While the **-IA/-OA** can be used in place of **-IM/-OM** to include the message or application <type>, **-IM/-OM** CANNOT be used to replace **-IA/-OA** when specifying a library and function name for user adapters.

Input Source Override - Pointer (-IP)

The Input Source Override - Pointer execution command (**-IP**) is for platform API users only, and allows data to be passed in via a pointer rather than a buffer.

-IP*card_num\$xxxx yyyy*

Option

Description

card_num

This is the card number of the input to override.

xxxx

This specifies the size of the data that is passed in via pointer.

yyyy

This specifies the address of the data that is passed in via pointer.

List (-L)

When using a Command Server on a platform other than Windows, use the List execution command (-L) to display a list of the data sources and targets that are compiled into a map file.

This command does not execute the map. It displays the list of sources and destinations.

For example, if you are using the Command Server on a UNIX platform, you can display the sources and destinations for the **sinkmap** example map by using the following command:

```
dtxcmdsv sinkmap.mmc -L
```

The result would be similar to the following:

```
Map file: /examples/general/map/sinkmap/sinkmap.mmc -L
Card information for map /examples/general/map/sinkmap/
sinkmap.mmc:
T column is data type: F=File, A=Application, D=Database
  Name          T Data
  ----          -
Input # 1 all_employees      F allemps.txt
Input # 2 employee_to_lookup F employee.txt
Output # 1 week              F
Output # 2 ttlhours          F
Output # 3 emp_hours         F emphours.txt
```

This information can be useful when determining the current source or target names, as well as to confirm the exact source, target types, or locations.

Initialization File Override (-N)

To override the default initialization (config.yaml) file for the Command Server, use the -N command option as the first command to the Command Server, as indicated in the following syntax:

```
-N=<INI_filename>
```

where <

<INI_filename> is the name of the renamed initialization file. If the path is not absolute, it will be resolved to the path relative to the current path. The current path is the path from where the command server was invoked. Also, if spaces exist in this path/name, enclose the <INI_filename> value in single quotes. When running a map using this override, the entire syntax should be written in the following way:

```
[<INI_filename>] [<mapname> <commands> ...]
```

The following example uses the myinit.ini initialization file to run the sdq.mmc map with an execution audit on a Windows operating system:

```
dtxcmdsv64 -n=c:\dtx_n.n\config.yaml sdq.mmc -ae
```

Output Target Override - Application (-OA)

Use the Output Target Override - Application execution command (-OA) to override specifications in the compiled map file for a specific output card for a single execution of a map.

```
-OA[alias]card_num[M['filename']]
[Rcount:interval][B][EN][EA][EF][BS][EW][AF][AI][U][!U]
[K['filename']]|['filename']|[U'filepath']|[U]
[1['filename']]|[1+'filename']|[U1'filepath']|[U1]
[library],[function],[application_command_line]|
{target-adptr-cmd} .
```

Option

Description

alias

This is the adapter alias for the specific application adapter. For example, **FTP** is the adapter alias for the FTP adapters. If the **alias** option is not used, you must use **library**, **function**, and **application_command_line**. If **alias** is used, you must use **target-adptr-cmd** or **"."**.

card_num

This is the card number of the output to override.

M'filename'

This is the metadata (XML schema or DTD) file location of the output card to override.

Specify **filename** as an absolute path along with the file name and enclosed in single quotes, for the metadata when parsing output XML. An example is:

```
M'k:\my_project_folder\myschema.xml'
```

Use this command option to override the Metadata location setting for the root type specified on an output card in the Map Designer GUI.

Rcount:interval

Specify **Retry** settings. If the target is unavailable, the adapter attempts to access the target as many times as specified by the **count** setting at the interval specified by the **interval** setting.

count number of attempts to access the application

interval number of seconds to wait between attempts. If this option is not specified, **Retry** settings compiled into the map are used. To eliminate adapter retry, specify **RO:0**.

B	If the map, burst, or card does not complete successfully, roll back any changes made to this data target. If this option is not specified, the OnFailure setting compiled into the map is used.
EN	Use this command option to never call the external parser for document verification.
EA	Use this command option to always call the external parser for document verification.
EF	Use this command option to call the external parser for document verification only if validation fails.
ES	Use this command option to call the external parser for document verification only if validation is successful.
EW	Use this command option to call the external parser for document verification only for the well-formed document.
AF	Use this command option to fail on adapter warnings.
AI	Use this command option to ignore adapter warnings.
U	Use this command option to apply the burst setting.
!U	Use this command option to not burst (integral mode).
K	When the map, burst, or card runs, no backup file will be created.
K'filename'	When the map, burst, or card runs, a backup file will always be created to <i>filename</i> . If ' <i>filename</i> ' is not an absolute path, the map directory will be appended as a prefix to it.
K+filename'	When the map, burst, or card runs, a backup file will always be appended to <i>filename</i> . If ' <i>filename</i> ' is not an absolute path, the map directory will be appended as a prefix to it.
KU'filepath'	When the map, burst, or card runs, a backup file will always be created using a unique backup name in the directory <i>filepath</i> . If ' <i>filepath</i> ' is not an absolute path, the map directory will be appended as a prefix to it.
KU	When the map, burst, or card runs, a backup file will always be created using a unique backup name in the map directory.
K1'filename'	If the map, burst, or card does not successfully complete, a backup file will be created to <i>filename</i> . If ' <i>filename</i> ' is not an absolute path, the map directory will be appended as a prefix to it.
K1+filename'	If the map, burst, or card does not successfully complete, a backup file will be appended to <i>filename</i> . If ' <i>filename</i> ' is not an absolute path, the map directory will be appended as a prefix to it.
KU1'filepath'	If the map, burst, or card does not successfully complete, a backup file will be created using a unique backup name in the directory <i>filepath</i> . If ' <i>filepath</i> ' is not an absolute path, the map directory will be appended as a prefix to it.
KU1	If the map, burst, or card does not successfully complete, a backup file will be created using a unique backup name in the map directory.
library	This command option specifies the user library name for the adapters when using an application adapter as a target and should be followed by the function name and application command line, separated by commas. This option, along with the function name and application command line, must be used when <i>alias</i> is not specified.
function	This command option specifies the function name for the adapters when using an application adapter as a target and should be prefaced by the user library name and followed by the application command line, separated by commas.
application_command_line	This command option specifies the application command line for the adapters when using an application adapter as a target and should be prefaced by the user library name and the function name, separated by commas.
target-adptr-cmd	This command option provides the adapter-specific commands that connect to the data target and process the output data. Either this option or ":" must be used if the <i>alias</i> option is specified. See the adapter-specific documentation in the <i>Resource Adapters</i> documentation.
	Use Command in the compiled map file. Either this option or <i>target-adptr-cmd</i> must be used if the <i>alias</i> option is specified.

Output Target Override - Database (-OD)

Use the Output Target Override - Database execution command (-OD) to override specifications in the compiled map file for a specific output card for a single execution of a map.

```
-ODcard_num[M['filename']]
 [Rcount:interval] [B] [EN] [EA] [EF] [ES] [EW] [AF] [AI] [U] [+U]
 [K['filename']] [+filename] [U'filepath'] [+U] [+1filename] [+1+filename] [+U1filepath'] [+U1]
 {target-adptr-cmd} .
```

Option

Description

card_num

This is the card number of the output to override.

M'filename'
This is the metadata (XML schema or DTD) file location of the output card to override.
Specify *filename* as an absolute path along with the file name and enclosed in single quotes, for the metadata when parsing output XML. An example is:

```
M'k:\my_project_folder\myschema.xml'
```

Use this command option to override the Metadata location setting for the root type specified on an output card in the Map Designer GUI.

Rcount:interval
Specify **Retry** settings. If the target is unavailable, the adapter attempts to access the target as many times as specified by the *count* setting at the interval specified by the *interval* setting.
count number of attempts to access the database
interval number of seconds to wait between attempts. If this option is not specified, the **Retry** settings compiled into the map are used. To eliminate adapter retry, specify **RO:0**.

B
If the map, burst, or card does not complete successfully, roll back any changes made to this data target. If this option is not specified, the **OnFailure** setting compiled into the map is used.

EN
Use this command option to never call the external parser for document verification.

EA
Use this command option to always call the external parser for document verification.

EF
Use this command option to call the external parser for document verification only if validation fails.

ES
Use this command option to call the external parser for document verification only if validation is successful.

EW
Use this command option to call the external parser for document verification only for the well-formed document.

AF
Use this command option to fail on adapter warnings.

AI
Use this command option to ignore adapter warnings.

U
Use this command option to apply the burst setting.

!U
Use this command option to not burst (integral mode).

K
When the map, burst, or card runs, no backup file will be created.

K'filename'
When the map, burst, or card runs, a backup file will always be created to *filename*. If '*filename*' is not an absolute path, the map directory will be appended as a prefix to it.

K+'filename'
When the map, burst, or card runs, a backup file will always be appended to *filename*. If '*filename*' is not an absolute path, the map directory will be appended as a prefix to it.

KU'filepath'
When the map, burst, or card runs, a backup file will always be created using a unique backup name in the directory *filepath*. If '*filepath*' is not an absolute path, the map directory will be appended as a prefix to it.

KU
When the map, burst, or card runs, a backup file will always be created using a unique backup name in the map directory.

K1'filename'
If the map, burst, or card does not successfully complete, a backup file will be created to *filename*. If '*filename*' is not an absolute path, the map directory will be appended as a prefix to it.

K1+'filename'
If the map, burst, or card does not successfully complete, a backup file will be appended to *filename*. If '*filename*' is not an absolute path, the map directory will be appended as a prefix to it.

KU1'filepath'
If the map, burst, or card does not successfully complete, a backup file will be created using a unique backup name in the directory *filepath*. If '*filepath*' is not an absolute path, the map directory will be appended as a prefix to it.

KU1
If the map, burst, or card does not successfully complete, a backup file will be created using a unique backup name in the map directory.

target-adptr-cmd
Specify the settings for the target using the adapter-specific commands. See the appropriate adapter information in the *Resource Adapters* documentation.

Use **Command** in the compiled map file.

Output Target Override - Echo (-OE)

Use the Output Target Override - Echo execution command (-OE) to override specifications in the compiled map file for a specific output card for a single execution of a map.

Use this command when you want the data resulting from one or more output cards passed back to a calling application or a map, rather than being directed to a file, database, application, or message. If this command is used to override more than one output, the results of the outputs are concatenated without separators.

For example, use this command in conjunction with the **RUN()** function to pass data from one map to another:

```
-OEcard_num[M['filename']]
[S|X][EN][EA][EF][ES][EW][AF][AI][U][!U]
```

Option	Description
<i>card_num</i>	This is the card number of the output to override.
M'filename'	This is the metadata (XML schema or DTD) file location of the output card to override. Specify <i>filename</i> as an absolute path along with the file name and enclosed in single quotes, for the metadata when parsing output XML. An example is: M'k:\my_project_folder\myschema.xml'
S	Use this command option to override the Metadata location setting for the root type specified on an output card in the Map Designer GUI.
	Return the size (in bytes) of the output data. This option causes the resulting output data to be returned as an ASCII number representing the size of the data, followed by a space, followed by the data itself. For example, if used in a RUN() function: RUN ("mymap.mmc", "-AE -OE2S -OE4S")
	If the data resulting from output card #2 is This and That and the output from output card #4 is 19439.27, the string returned by this example RUN() function is 13 This and That 8 19439.27.
	If the size option is not used, the data returned by this example RUN() function is This and That 19439.27.
X	Exclude this card from echo. The output is stored in memory and not echoed back to the specified map or application. You might use this option to optimize performance when a map has an output used strictly as a <i>scratch</i> card, that is, it is built only to be used by a later output.
EN	Use this command option to never call the external parser for document verification.
EA	Use this command option to always call the external parser for document verification.
EF	Use this command option to call the external parser for document verification only if validation fails.
ES	Use this command option to call the external parser for document verification only if validation is successful.
EW	Use this command option to call the external parser for document verification only for the well-formed document.
AF	Use this command option to fail on adapter warnings.
AI	Use this command option to ignore adapter warnings.
U	Use this command option to apply the burst setting.
!U	Use this command option to not burst (integral mode).

Output Target Override - File (-OF)

Use the Output Target Override - File execution command (-OF) to override data destination specifications for a specific output card in a compiled map file for a single execution of a map.

```
-OFcard_num[+!+][M['filename']]  
[X|X0][Rcount:interval][B][EN][EA][EF][ES][EW][AF][AI][U][!U]  
[K['filename']]|[+'filename']|[U'filepath']|[U]|  
[1'filename']|[1+'filename']|[U1'filepath']|[U1]|  
{target}..
```

Option	Description
<i>card_num</i>	This is the card number of the output to override.
+	If the map, burst, or card completes successfully, the output data is appended to an existing data file. If the data file does not exist, it is created.
!+	The data file is created at the start of map execution. If the file exists, it is overwritten. If more than one burst occurs, data is appended to the same data file.
M'filename'	This is the metadata (XML schema or DTD) file location of the output card to override. Specify <i>filename</i> as an absolute path along with the file name and enclosed in single quotes, for the metadata when parsing output XML. An example is: M'k:\my_project_folder\myschema.xml'
R <i>count:interval</i>	Use this command option to override the Metadata location setting for the root type specified on an output card in the Map Designer GUI.
	Specify Retry settings. If the target is unavailable, the adapter attempts to access the target as many times as specified by the <i>count</i> setting at the interval specified by the <i>interval</i> setting. <i>count</i> number of attempts to access the file <i>interval</i> number of seconds to wait between attempt. If this option is not specified, the Retry settings compiled into the map are used. To eliminate adapter retry, specify RO:0 .

	If the map, burst, or card does not complete successfully, roll back any changes made to this data target. If this option is not specified, the OnFailure setting compiled into the map is used.
EN	Use this command option to never call the external parser for document verification.
EA	Use this command option to always call the external parser for document verification.
EF	Use this command option to call the external parser for document verification only if validation fails.
ES	Use this command option to call the external parser for document verification only if validation is successful.
EW	Use this command option to call the external parser for document verification only for the well-formed document.
AF	Use this command option to fail on adapter warnings.
AI	Use this command option to ignore adapter warnings.
U	Use this command option to apply the burst setting.
!U	Use this command option to not burst (integral mode).
K	When the map, burst, or card runs, no backup file will be created.
K'filename'	When the map, burst, or card runs, a backup file will always be created to <i>filename</i> . If ' <i>filename</i> ' is not an absolute path, the map directory will be appended as a prefix to it.
K+'filename'	When the map, burst, or card runs, a backup file will always be appended to <i>filename</i> . If ' <i>filename</i> ' is not an absolute path, the map directory will be appended as a prefix to it.
KU'filepath'	When the map, burst, or card runs, a backup file will always be created using a unique backup name in the directory <i>filepath</i> . If ' <i>filepath</i> ' is not an absolute path, the map directory will be appended as a prefix to it.
KU	When the map, burst, or card runs, a backup file will always be created using a unique backup name in the map directory.
K1'filename'	If the map, burst, or card does not successfully complete, a backup file will be created to <i>filename</i> . If ' <i>filename</i> ' is not an absolute path, the map directory will be appended as a prefix to it.
K1+'filename'	If the map, burst, or card does not successfully complete, a backup file will be appended to <i>filename</i> . If ' <i>filename</i> ' is not an absolute path, the map directory will be appended as a prefix to it.
KU1'filepath'	If the map, burst, or card does not successfully complete, a backup file will be created using a unique backup name in the directory <i>filepath</i> . If ' <i>filepath</i> ' is not an absolute path, the map directory will be appended as a prefix to it.
KU1	If the map, burst, or card does not successfully complete, a backup file will be created using a unique backup name in the map directory.
target	Specify the name of the output file. If the output file is not located in the same directory as the map file, the full path must be specified using the platform-specific syntax.
	Use the FilePath in the compiled map file.

Output Target Override - Message (-OM)

Use the Output Target Override - Message execution command (-OM) to override data destination specifications for a specific output card in a compiled map file for a single execution of a map.

```
-OM[alias]card_num[M['filename']]
[X|X0][Rcount:interval][B][EN][EA][EF][ES][EW][AF][AI][U][!U]
[K['filename']]|[+'filename']|[U'filepath']|[U]
[1'filename']|[1+'filename']|[U1'filepath']|[U1]
{target-adptr-cmd} .
```

Option

Option	Description
alias	This is the adapter alias for the specific messaging adapter. For example, <code>MSMQ</code> is the adapter alias for the Microsoft MSMQ adapters. See the adapter-specific information in the <i>Resource Adapters</i> documentation.
card_num	This is the card number of the output to override.
M'filename'	This is the metadata (XML schema or DTD) file location of the output card to override. Specify <i>filename</i> as an absolute path along with the file name and enclosed in single quotes, for the metadata when parsing output XML. An example is: <code>M'k:\my_project_folder\myschema.xml'</code>
X	Use this command option to override the Metadata location setting for the root type specified on an output card in the Map Designer GUI.
X0	If the map, burst, or card completes successfully, the message is deleted from its target.

If the map, burst, or card completes successfully, the message is deleted from its target only if it has no content. If neither delete option (x or x0) is specified, the **OnSuccess** setting compiled into the map is used.

Rcount:interval

Specify **Retry** settings. If the target is unavailable, the adapter attempts to access the target as many times as specified by the *count* setting at the interval specified by the *interval* setting.
count number of attempts to access the message queue.

interval number of seconds to wait between attempts. If this option is not specified, the **Retry** settings compiled into the map are used. To eliminate adapter retry, specify **R0:0**.

B

If the map, burst, or card does not complete successfully, roll back any changes made to this data target. If this option is not specified, the **OnFailure** setting compiled into the map is used.

EN

Use this command option to never call the external parser for document verification.

EA

Use this command option to always call the external parser for document verification.

EF

Use this command option to call the external parser for document verification only if validation fails.

ES

Use this command option to call the external parser for document verification only if validation is successful.

EW

Use this command option to call the external parser for document verification only for the well-formed document.

AF

Use this command option to fail on adapter warnings.

AI

Use this command option to ignore adapter warnings.

U

Use this command option to apply the burst setting.

IU

Use this command option to not burst (integral mode).

K

When the map, burst, or card runs, no backup file will be created.

K'filename'

When the map, burst, or card runs, a backup file will always be created to *filename*. If '*filename*' is not an absolute path, the map directory will be appended as a prefix to it.

K+'filename'

When the map, burst, or card runs, a backup file will always be appended to *filename*. If '*filename*' is not an absolute path, the map directory will be appended as a prefix to it.

KU'filepath'

When the map, burst, or card runs, a backup file will always be created using a unique backup name in the directory *filepath*. If '*filepath*' is not an absolute path, the map directory will be appended as a prefix to it.

KU

When the map, burst, or card runs, a backup file will always be created using a unique backup name in the map directory.

K1'filename'

If the map, burst, or card does not successfully complete, a backup file will be created to *filename*. If '*filename*' is not an absolute path, the map directory will be appended as a prefix to it.

K1+'filename'

If the map, burst, or card does not successfully complete, a backup file will be appended to *filename*. If '*filename*' is not an absolute path, the map directory will be appended as a prefix to it.

KU1'filepath'

If the map, burst, or card does not successfully complete, a backup file will be created using a unique backup name in the directory *filepath*. If '*filepath*' is not an absolute path, the map directory will be appended as a prefix to it.

KU1

If the map, burst, or card does not successfully complete, a backup file will be created using a unique backup name in the map directory.

trgt-adptr-cmd

This option provides the adapter-specific commands that connect to the data source and retrieve the input data. See the adapter-specific information in the *Resource Adapters* documentation.

Use **Command** in the compiled map file.

Output Target Override - Pointer (-OP)

The Output Target Override - Pointer execution command (-OP) is for platform API users only, and allows data to be returned via a pointer rather than a buffer.

-OPcard_nums

Option

Description

card_num

This is the card number of the output to override.

S

The data is returned in the form **xxxx yyyy** where xxxx is the size of the data being returned and yyyy represents the address of the returned data.

WorkSpace PageSize (-P)

When a file-based WorkSpace is used, the amount of memory accessed at a given time depends on the number of pages and the size of pages you configure for a map to run. Use the WorkSpace PageSize execution command (-P) to specify the size of and number of pages to be used.

If this execution command is used without a corresponding WorkSpace execution command (-w), the WorkSpace settings in the compiled map are used.

Generally, for large data files, you might want to increase the number of pages by specifying a large count. For maps that include a large number of type references, you might want to increase the page size.

-Psize: count

Option

Description

size

Size in kilobytes of a single page of memory used during map execution. Page size is measured in kilobytes and is specified as a multiple of 2 between 2 and 1024.

count

Number of pages used during map execution. The count is specified as an integer greater than 0.

Refresh Status (-R)

The Refresh Status execution command (-R) is used to specify the frequency of map status update when displayed in the Command Server window when using the Windows operating system or in the console when using all other operating systems. To reflect execution time, as well as the number of input objects found and output objects built, specify the interval at which the map status is refreshed.

-Rupdate_rate

Option

Description

update_rate

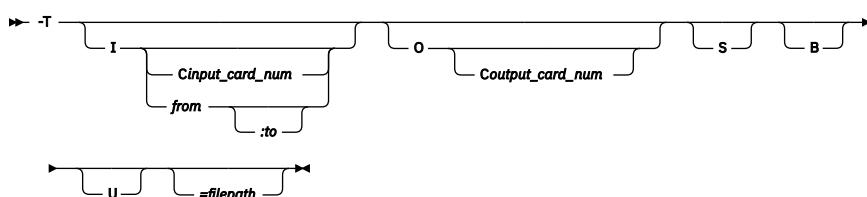
This is the number of times per second the map status is refreshed. The default refresh interval is approximately twice per second. If you specify 0 as the refresh interval, the map status is not displayed.

MapTrace (-T)

The MapTrace execution command (-T) creates a trace file that you can use to diagnose invalid data or incorrect type definitions. For more information about the trace file, see the *Map Designer* documentation.

Use this command to trace a particular input or output card, a range of input data objects, or to produce a trace summary. When the **-T** command is used without other options, no trace log is produced.

The MapTrace (-T) execution command overrides all trace settings in the compiled map. You cannot override individual map settings. When you run the **-T** command, only the trace options that you specify on the command are in effect.



Option

Description

I

Produces detailed trace information for all input cards.

Cinput_card_num

Produces detailed trace information for a single input card. For example, the command **-TIC3** produces detailed trace information for input card 3.

from[:to]

Produces detailed trace information for a range of input objects. For example, to produce trace information for objects 1000 to 1320, specify **-TI1000:1320**. The upper limit of the object range is optional. To trace from input object 1000 to the end of the file, you can specify **-TI1000**.

O

Produces detailed trace information for all output cards.

Coutput_card_num

Produces detailed trace information for a single output card. For example, the command **-TOC3** produces detailed trace information for output card 3.

S

Produces summary level trace information for all input and output cards providing a message for each card. For example, the message for a particular card might be Input 1 was valid, but 34 bytes of unknown data found or Output 2 built successfully.

B

Creates the trace file in binary format.

U

Creates the trace file with a unique name, in the format *Mer_mapname_processKey_mapCounter_hostname.mtr*.

=filepath

Specifies a file name, file path, or file name and path for the trace file. By default, the trace file is created as *mapname.mtr* in the directory of the compiled map (.mmc) file.

If you specify **=filepath**, it must be the last option on the **-T** command.

- [Examples of the MapTrace \(-T\) command](#)

Examples of the MapTrace (-T) command

The commands in the following table show combinations of the **-T** command options:

```
-T [I [C | [from[:to]]] [O [C]] [S] [B] [U] [=filepath]
```

The table is not an exhaustive list of all possible option combinations.

The command options are not positional, with these exceptions:

- If specified, the *filepath* option must be the last option on the command.
- The *C* and *from:to* variables must follow the *I* option. The *C* must follow the *O* option.

For example, in the **-TIC30C4B** command, 3 is the number of the input card and 4 is the number of the output card. However, **-TBOC4IC3** is an identical command.

Table 1. Example MapTrace commands

MapTrace command	Trace target	Trace file format	Trace file name	Trace file location
-T	Do not trace	text	<i>mapname.mtr</i>	map directory
-TI	Trace all inputs	text	<i>mapname.mtr</i>	map directory
-TI2:10	Trace input objects 2 through 10	text	<i>mapname.mtr</i>	map directory
-TIC7	Trace input card 7	text	<i>mapname.mtr</i>	map directory
-TIB	Trace all input objects	binary	<i>mapname.mtr</i>	map directory
-TIBU	Trace all inputs	binary	unique	map directory
-TIBU=C:\TXTRACES	Trace all inputs	binary	unique	C:\TXTRACES directory
-TI2:10BU=C:\TXTRACES	Trace input objects 2 through 10	binary	unique	C:\TXTRACES directory
-TO	Trace all output cards	text	<i>mapname.mtr</i>	map directory
-TOC3BU=C:\TXTRACES\Card3Trc.mtr	Trace output card 3	binary	Card3Trace.mtr	C:\TXTRACES directory
-TIC30C4	Trace input object 3 and output card 4	text	<i>mapname.mtr</i>	map directory
-TI2:22OC1	Trace input objects two through 22 and output card 1	text	<i>mapname.mtr</i>	map directory
-TS	Trace all input and output cards and provide a message for each card	text	<i>mapname.mtr</i>	map directory

Stop Validation (-V)

The Stop Validation execution command (**-v**) specifies to stop validation at the card where the first error is detected. When this command is used, validation occurs only through the input card where the first error is encountered. Map execution is stopped after validating that input card, even if there are multiple input cards.

Under certain circumstances, multiple errors are detected during validation. You can stop by using **-V** at the first error detected or continue (with no **-V** validating data until all errors are found).

If you use the **-V** command, validation only occurs through the input card where the first error occurred. It stops executing the map after validating that input card, even if there are multiple input cards. If the **-V** command is not used, the Validation OnValidationError setting is used.

In all situations, if an error is found in the input, no output objects are built.

WorkSpace (-W)

When you run a map, the Command Server uses its own paging scheme to access information about the data as efficiently as possible. Portions of both workspace data and actual data are paged as needed. The workspace can be either a combination of files and memory or memory-based. When file-based workspace is used, the amount of memory accessed at a given time depends on the number of pages and the count of pages configured for map execution.

When the WorkSpace execution command (**-W**) is used without other specified options, a file-based WorkSpace is used and the work files are named using the map name; they are placed in the same directory as the compiled map file. If the **-W** command is used without a corresponding WorkSpace PageSize execution command (**-P**), the **WorkSpace** settings in the compiled map are used.

You can optimize the speed of map processing using the workspace configuration setting. When processing a large amount of input data in a map, use the file-based workspace with default paging. When small amounts of data are processed, smaller page sizes and memory-based workspaces are recommended. When using burst processing, adjust the workspace configuration for the size of burst data.

```
-W [M] | { [D] [U] [=dir] }
```

Option	Description
D	A file-based WorkSpace is used. The work files are deleted after a map is executed. All work files are deleted, except those being saved for reuse as specified on the input cards for each map.

U

A file-based WorkSpace is used. A unique name is generated for each work file created and the files are automatically deleted after the map is executed. Unique work file names are useful if you are running more than one instance of the same map at the same time. Each time a map is run, a work file is created with a unique name, which eliminates any conflict between work files.

Unique workfiles are automatically deleted upon completion. The format of the unique workfile name is:

`Mer_mapname_processID_time_xxxxxxx_hostname[I|O].cardnumber`

mapname

The executable map name.

processID

The process id from the operating system.

time

The time when the process started, in number of seconds since midnight on January 1, 1970.

xxxxxx

The mapping instance for the process, as an incremental hex number starting at 1 (thru 0xFFFFFFF).

hostname

The host name of the computer where Transformation Extender is running.

[I|O]

I is an input and **O** is an output

cardnumber

The input or output card number.

M

A memory-based WorkSpace is used. This might enhance performance for specific maps depending on work file requirements and the amount of available memory by keeping the work file information in memory and eliminating the I/O necessary with file-based work files. If there is insufficient memory available to build the work file, the map returns a *Disk write error* message.

When this option is used, none of the other work file options can be used. For example, **D** - deleting work files, **U** - unique work file names, and **=dir** - specifying the directory.

=dir

Specify the directory path in which you want the work files to be created. By default, work files are created in the same directory as the compiled map (.mmc) file.

Retry (-Y)

The Retry execution command (-y) is used to specify the retry rate for non-data files accessed during map execution. This includes the compiled map file, as well as, work files, the trace file, and the audit log. If one of these files cannot be accessed at execution time, an attempt is made to access the file as specified.

-Ycount:interval

Option

Description

count

This is the number attempts to be made to access the file.

interval

This is the number of seconds to wait between each attempt.

Ignore Warnings (-Z)

The execution Ignore Warnings execution command (-z) specifies for a map that selected warning codes should be ignored and a 0 return code should be issued. When -z is used with one or more warning codes, a 0 return code is issued for those warning codes. When used without other options specified, all warning codes result in a 0 return code. Using this command impacts **OnSuccess** settings, **OnFailure** settings, and so on.

-Z[!][warning_code[:warning_code...]]

Option

Description

!

Use to specify those warning codes that, when encountered, remain in their default state; they will remain as warnings and a 0 return code is not issued. The default warning code is followed by the warning code message.

When used with -z only (-z!), a 0 return code followed by the warning code message is returned if any one of the warning codes is encountered.

warning_code

Specify the warning codes to be acted upon when encountered.

Valid warning code selections include: 14, 18, 21, 26, 27, 28, 29, and 34.

- [Examples using Ignore Warnings \(-Z\)](#)

Examples using Ignore Warnings (-Z)

-Z

This command causes a 0 return code with the warning message to be returned when any of the warning codes is encountered.

-Z!

This command causes a 0 return code with the warning message to be returned when any of the warning codes is encountered.

-Z18
This command causes a 0 return code with the warning message to be returned when a Page Usage Count Error (18) is encountered.

-Z!18
This command excludes the Page Usage Count Error (18), a 0 return code with the warning message is returned for all of the other warning codes.

-Z18:27:28
This command causes a 0 return code with the warning message to be returned when any of the specified warning codes is encountered (18, 27, or 28 in this case).

-Z34
This command causes a 0 return code with the warning message to be returned when a Too Few Pages Requested, More Allocated Error (34) is encountered.

If the Ignore Warnings command (-Z) is enabled and a map encounters the condition for any warning code as specified, the execution section of the audit log includes a Valid map status with a 0 return code and the associated warning message. For example:

```
<Summary MapStatus="Valid" Return="0">
  <ElapsedTimeInSec>0.1026</ElapsedTimeInSec>
  <Message>Input type contains errors.</Message>
  <CommandLine>'install_dir\examples\ansiack\receiveEDI.mmc'
  </CommandLine>
  <ObjectsFound>0</ObjectsFound>
  <ObjectsBuilt>0</ObjectsBuilt>
</Summary>
```

When using the -Z command, you cannot specify the same warning code using the Fail on Warnings command (-F).

Both the -Z and -Z! commands cause a 0 return code with the warning message to be returned when any of the warning codes is encountered.

Examples

This documentation presents a variety of examples involving the execution of maps using various execution commands in a **RUN()** function and from the command line in UNIX and Windows environments.

- [Running a map located in a different directory](#)
- [Creating work files in an alternate directory](#)
- [Executing multiple maps from a single command](#)
- [Overriding a database target](#)
- [Overriding the schema location at run time](#)
- [Executing a command file](#)
- [Using a run function and echoed outputs in memory](#)
- [Audit Log in memory](#)

Running a map located in a different directory

From a UNIX command line, enter:

```
/install_dir/dtxcmdsv /x12/edi.mmc -TIO -IF2!WXR5:1 dtin -OF1B out.txt
```

Option

Description

/X12/edi.mmc

This specifies the full path and name of the map to be executed.

-TIO

A trace file is created or all input and output objects are placed in the same directory as the map and named using the **<mapname>.mtr** file naming convention.

-IF2!WXR5:1 dtin

This changes the source of input card #2 to be a file named **dtin**. The other adapter settings for this card include:

Do not re-use workfiles.

If the map successfully executes, delete the **dtin** file.

If the file is not available, attempt to access it five more times at one-second intervals.

-OF1B out.txt

This changes the destination for the data from output card #1 to be a file named **out.txt**. The **B** option causes any changes to **out.txt** to be backed out (rolled back), if the map does not successfully execute. All other adapter settings are as compiled into the map file.

The file locations in this example are relative to the location of the compiled map file.

Creating work files in an alternate directory

From a Windows command line, enter:

```
install_dir\dtxcmdsv k:\maps\invoices.mmc -AE -T -WD=c:\tmp
```

Option

Description

k:\maps\invoices.mmc

-AE	This specifies the name of the map to be executed.
-T	This produces an execution audit log using the AuditLocation settings for MapAudit compiled into the map file.
-WD=c:\tmp	This specifies that no trace information is to be produced.
	This creates the work files in the c:\tmp directory and deletes the work files after the map file is executed.

Executing multiple maps from a single command

From a Windows command line, enter:

```
dtxcmdsv mymap.mmc -R2 -D1949 flatfile.mmc -F -Y10:2
```

Option	Description
mymap.mmc	This specifies the name of the first map to be executed, mymap.mmc .
-R2	This causes the server display to be updated two times per second.
-D1949	This command option specifies that dates without a century value should be converted. Years between 00 and 49 are interpreted as 20xx and years between 50 and 99 are interpreted as 19xx.
flatfile.mmc	This command option specifies the name of another map to be executed, flatfile.mmc . When flatfile.mmc is executed, it uses the default options, not the options that precede it on the command line.
-F	Fail on the return of any warning code.
-Y10:2	If any non-data file resource is unavailable, such as the map file, audit log file, work file, and so on, up to ten attempts should be made at two-second intervals to access the unavailable resource.

Overriding a database target

Suppose you want to run the map **multitran.hp**, overriding some of the settings for the fourth output card, which is a database target.

Use the following command:

```
/install_dir\dtxcmdsv multitran.hp -G -OD4R5:3B "``-DELETE -TRACE ERROR``"
```

Option	Description
multitran.hp	This specifies the name of the map to be executed, multitran.hp .
-G	All item properties, including size, presentation, and restrictions, are fully validated.
-OD4R5:3B "``...``"	This changes settings for the database target for the data of output card #4 as follows: R5:3 <ul style="list-style-type: none"> • If the database resource is unavailable, attempt to make the connection five more times at three-second intervals
B	<ul style="list-style-type: none"> • If the map does not complete successfully, roll back any changes made to the database table
"``-DELETE -TRACE ERROR``"	These database-specific adapter commands described in the <i>Resources Adapters</i> documentation indicate that all rows in the target database table should be deleted before processing the output data and a trace file should be generated that only contains error information.

Overriding the schema location at run time

When you have a map that is designed to parse input XML data, but want to use a different schema location than what was set up in the Metadata location setting for the root type on the input card in the Map Designer GUI, run the map from the command line and override the map settings. Use the Input Source Override - File execution command (-IF) and specify the card number and the metadata location (M) command line option with the schema file location, enclosed in single quotes.

From a Windows command line, enter:

```
install_dir\dtxcmdsv mymap.mmc -IF1M'k:\billing_project\myschema.xsd' myfile.txt
```

Option	Description
--------	-------------

```

mymap.mmc
  This specifies the name of the compiled map file to be run.
-IF1
  This will change the source file specification on input card 1 in the mymap.mmc compiled map file to myfile.txt.
M'k:\billing_project\myschema.xsd'
  This specifies the new metadata location for input card 1.
myfile.txt
  This is the source file, specified on the -IF execution command, that will override the file specified on input card 1.

```

Executing a command file

As an alternative to typing execution commands on the command line, you can run maps using maps and execution commands in a command file. To use a command file named **cmd_file.txt**, from a UNIX command line, enter:

```
dtxcmdsv @cmd_file.txt
```

A command file can contain spaces and new lines to separate arguments and maps to be run. The following is an example of a command file to be executed on a UNIX platform:

```

/x12/edi.mmc -TIO -IF2!WXR5:1 dtin -OF1B out.txt
/maps/invoices.mmc -AE -T -WD=/tmp
mymap.mmc -R2 -D1949
flatfile.mmc -F -Y10:2
/maps/daily/multitran.hp -G -OD4R5:3B ` -DELETE -TRACE ERROR'

```

Using this command file, the maps are executed sequentially. These command files can be manually created or can be generated by the Integration Flow Designer for systems with an execution mode of Command.

Using a run function and echoed outputs in memory

When using the Output Target Override - Echo execution command (-OE) with the X option, the output is stored in memory and is not echoed back to the calling map or application. A typical scenario in which this is useful and can serve to optimize performance is when a map has an output that is only used as a *scratch* card. That is, it is built to be used only by a subsequent output.

For example, if used in a **RUN()** function:

```
RUN ("some_map.mmc", "-WM -OE1 -OE3X -OE4")
```

In this example, a memory-based WorkSpace is used and the data for output cards 1 and 4 are echoed back to the rule in which the **RUN** function is located. All other map settings and input and output card settings for executing this map, such as **MapAudit**, **Century**, and so on, would use the settings compiled into **some_map.mmc**.

This results in the execution of **some_map.mmc**. The data produced for output card 1 and 4 is concatenated and *echoed* back to the rule containing the **RUN** function. The data for output card 3 is not *echoed* back to the rule because of the use of the X option. Given that output card 1 builds 103910^ABC, output card 3 builds justsomedata, and output card 4 builds 01/23/98, the **RUN** function returns 103910^ABC01/23/98.

The most common use for this feature of echoed outputs is when the data for an output card is needed only during the execution of the map, perhaps for use on a later output card, and can be discarded after the map has completed execution.

For another example of using an echoed data destination, see the *Platform API* documentation.

Audit Log in memory

Similar to how the data built for an output card can be echoed back to a calling map or application, the audit log, including Data and Execution information, can be placed in memory using the MapAudit execution command (-A) with the M option and echoed back to its calling map or application with the E option.

If the audit in memory fails, either of these messages might be returned from the Command Server: Disk write error or Not enough memory to execute map.

The -AEM audit option can be useful when using a **RUN** function or the Platform API to place the audit log in memory during map execution and then to return it, along with any outputs being echoed back. Placing the audit log in memory avoids the file I/O necessary if the audit log was written to disk.

For example, if used in a **RUN()** function, enter:

```
RUN ("some_map.mmc", "-AEM -WM -OE1")
```

This results in the execution of **some_map.mmc** and returns the concatenation data created for output card 1 and the audit log data.

When the audit log data is returned in this function implementation, it still adheres to the format described by the **audit.mtt** type tree included in the examples directory.

When the audit log is written to memory and is echoed back to the calling map or application, it is appended to any echoed outputs. Therefore, if output cards 2 and 4 were echoed, along with the audit log, the data for output card 2 is followed by the data for output card 4, which is followed by the contents of the audit log.

INI commands for debugging maps

This documentation presents the **config.yaml** file setting you can use for the debugging of adapter connections for maps run using non event-driven execution methods not run through events using the Launcher. This includes using a command-driven execution method such as the Command Server, a command parameter file, batch file, shell script, graphical user interface (GUI), scheduling program, and JCL on the mainframe, or an application-embedded execution method using components of the Development Kit in which maps can be embedded in specific third-party applications for web-based integration scenarios.

For information about debugging maps run using event-driven execution methods, see the *Launcher* documentation, *Map Designer* documentation and individual adapter documentation.

The logging feature is used to help you debug adapter connection issues. To use this feature, configure the debugging paths under /runtime/Connections Manager/log paths in the config.yaml file. .

- [Logging](#)

Logging

The logging feature (non-Launcher usage) enables you to specify the type of information you want to capture about adapter connections in log files. You do this by configuring the following severity classes and debug components. They are listed in the order of impact of these severity classes and debug components referred to as logging categories, from lowest to highest, along with their equivalent entry in the **/runtime/Connections Manager/log** logging entry. The logging categories in the **config.yaml** file will be configured to `false` to turn that category off or `true` to turn that logging category on.

Logging Category

Logging Category in the Connections Manager

Trace	/runtime/Connections Manager/log/trace
Debug	/runtime/Connections Manager/log/debug
Info	/runtime/Connections Manager/log/info
Warn	/runtime/Connections Manager/log/warning
Error	/runtime/Connections Manager/log/error
Fatal	/runtime/Connections Manager/log/fatal

- [Logging categories](#)
- [Debug](#)

Logging categories

The logging categories, non-Launcher usage, provide a variety of debugging information. They are used to determine which severity levels and components to debug.

```
/runtime/Connections Manager/log/X=  
Provide the specific category, such  
as trace, debug, info, warning, error or fatal,  
for X and false to turn that logging category off, or true (to turn that category  
on) for n.  
Example: trace: true
```

Provide the specific category, such as Trace, Debug, Info, Warning Or Fatal, for X and false to turn that logging category off, or true (to turn that category on) for n.

Example: LogTrace=true

Logging Category	Value	Description
LogTrace	false	Disabled
	true	Enables the logging of sequence of events Example Results: Before Resource Pending <map> After Resource Pending <map>
LogDebug	false	Disabled
	true	Enables the logging of information to diagnose adapter connection behavior Example Results: New audit file name <name> Adapter wildcard.<wildcard>
LogInfo	false	Disabled
	true	Enables the logging of general information about the state of adapter connections Example Results: :Map started InitPendingHigh reached
.LogWarning	false	Enabled

Logging Category	Value	Description
	true	Enables the logging of issues that are not severe, but might be important to know Example Results: Resource Manager table too small - using default Map warning <map>
.LogError	false	Enabled
	true	Enables the logging of critical adapter connection errors Example Results: Map failure <map> Socket failure <socket_type>
LogFatal	false	Enabled
	true	Enables the logging of a fatal error Example Results: Corrupted MMC file during initialization <mmc_name> Adapter listener Initialization failure <adapter>

Debug

Use the **DebugName** and **DebugAppend** commands configured in the **config.yaml** file under /runtime/Connections Manager to specify how you want to create the output log file.

- [DebugName](#)
- [DebugAppend](#)

DebugName

Use this setting to define the complete file name of the debug file. Remember that this logging category is only for non-Launcher usage and cannot be configured dynamically in the Management Console. When the map is started, it will use the settings that were configured in the **config.yaml** file.

The default file name is **dtxconn.log**.

The syntax is the following:

DebugName: [file_name]

Option

Description

file_name

The file name of the debug file for the Connections Manager

DebugAppend

Use this setting to append the debug information to an already existing debug log file. Remember that this logging category is only for non-Launcher usage and cannot be configured dynamically in the Management Console. When the map is started, it will use the settings that were configured in the **config.yaml** file.

The default file name is **dtxconn.log**.

The syntax is the following:

DebugAppend: {false | true}

Option

Description

false

Disables the append specification so that the debug information is written to a new debug log file.

true

Enables the append specification so that the debug information overwrites the contents of an existing debug log file.
This is the default.

Expressions and evaluations

You can use functions and expressions to create component rules in the Type Designer and to create map rules in the Map Designer.

The **Examples** directory, located in the product installation directory, contains type trees and maps that can help you learn how to accomplish your mapping tasks. These type trees and maps contain component and map rules that show the use of many functions and expressions.

- [Expressions](#)
 - [Literals](#)
 - [Data object names](#)
 - [Evaluating expressions](#)
 - [Operators](#)
 - [Using functions in expressions](#)
 - [Map names in expressions](#)
 - [Expressions that evaluate to NONE](#)
 - [Evaluated expressions assigned to output items](#)
 - [Automatic item format conversions](#)
-

Expressions

An expression is a statement about data objects. Expressions are used in component rules in the Type Designer and in map rules in the Map Designer.

The following are examples of expressions:

```
Account Balance = Credits - Debits
TrunkRoom < 15
PRESENT (Store#:StoreInfo)
```

As you enter an expression, you can add spaces before and after component names and operators to make the expression more readable. Additional spaces have no effect on an expression.

Expressions are a combination of literals, object names, operators, functions, and map names.

- [Component rule expressions evaluate to true or false](#)
 - [Map rule expressions evaluate to data](#)
-

Component rule expressions evaluate to true or false

Component rules are expressions that evaluate to "true" or "false". For example, the result of each of the following component rules is "true" or "false".

```
WHEN (PRESENT (Qualifier), PRESENT (Address))
COUNT (Exchange) < COUNT (Purchase)
Invoice#:Item = Invoice#:Item[1]
```

Map rule expressions evaluate to data

Map rules are expressions that evaluate to data.

For example, the result of each of the following map rules is data:

```
= "Florida"
= Price:Input
= COUNT ( Name:Roster )
= RecordMap ( FixedRecord )
```

Literals

A literal is a constant value. A literal may be a number or a text string.

The rules for using numeric literals are:

- A numeric literal is in integer or decimal format and can be signed.
- A comma separator should not be included in a numeric literal.
- A numeric literal cannot be greater than 254 digits.

The rules for using text literals are:

- A text literal is enclosed in double quotes.
- A double-quote included in a text literal is released by another double quote.

Literals are encoded in UTF-16 (Unicode).

The following are some examples of literals:

```
"This is a text literal"
"ABC Company"
"Some ""quoted data"" to show use of a double double-quote."
1.23
-9
1045
```

Carriage return/line feeds cannot be within a quoted string. To use a carriage return/line feed the string must be broken into two strings with <CR><LF> between them. For example:

```
="username=?????\" + "<CR><LF>" + "&password=?????"
```

will place the output username and password information on separate lines.

<CR><LF> must be within quotation marks.

Data object names

A data object is referenced in an expression by its name. An object name can be as simple as a card name or a local type name. It can also be complex depending on the object you want to reference.

- [Object names in map rules](#)
- [Object names in component rules](#)
- [Card name](#)
- [Local type name](#)
- [Partition list](#)
- [Component path](#)
- [Indexed object names](#)
- [Component paths separated by a colon](#)
- [Component paths separated by IN](#)
- [Comment object name](#)
- [Shorthand notation](#)

Object names in map rules

In a map rule, data object names always refer to the card name that contains the data object name. So, in a map rule, a data object name ends with a card name. In the following example image, the object name for the component **Contact** of the card **ContactFile** is **Contact:ContactFile**.

The general definition of an object name in a map rule can be divided into three parts as described in the following table. If one of these parts is used, it must be sequenced according to the order in the table.

Parts of an Object Name

Required

Component name

No

Set of component-paths, each ending with either a colon or the reserved word **IN**

No

Card name

Yes

When **IN** is used, there must be a space before and after it, such as **Dependent**

IN Claim.

Object names in component rules

In a component rule, data object names always refer to components in the same component list. In a component rule, a data object name ends with a component name. For example, if **LineItem** is in a component list, the object name for the **Qty Info** component of **LineItem** is **Qty Info:LineItem**.

In the preceding **Qty Info:LineItem** example, the colon would be interpreted as "is a component of". If the example was **Qty Info LineItem**, the space between **Info** and **LineItem** would be interpreted as "is a subtype of".

The general definition of a component rule object name can be divided into the two parts described in the following table. If one of these parts is used, it must be sequenced according to the order in the table.

Parts of an Object Name	Required
Set of one or more component-paths , infix separated with either a colon or the reserved word, IN	At least one is required
The reserved word, COMPONENT	No

IN and **COMPONENT** are reserved words. Reserved words are not case sensitive.

Card name

A map rule can reference the object of an entire card. A card name is a simple type name from 1 to 32 bytes in length in UTF-8 encoding. For example, if the input card name is **Invoice**, the name for the card object would be **Invoice**.

Local type name

An object name can be a local type name. A local type name is specified as one or more simple type names separated by spaces, such as:

City Field

Local type names are typically used to refer to a component. If used in a component rule, for example, the component referenced in a rule is specified as a local type name. In a map rule, local type names refer to components contained in a card object.

Local type names can also be used to refer to the partitioned type of a component. When a type is partitioned, you can refer to all of its partitions by simply referring to the partitioned type.

For this example, the type **Record** is partitioned into **Header**, **Detail**, and **Trailer**. The object name that references **Header**, **Detail**, and **Trailer Record** is: **Record**

Here is another example: the type **Transcript** has **Header Record**, **Detail Record**, and **Trailer Record** as components contained within it. The object name that references all **Header**, **Detail**, and **Trailer Records** within **Transcript** would be **Record IN Transcript**.

Partition list

A partition list is a set of partition names separated by the symbols <>. A partition name is the simple name that represents the partitions found under the parent, or partitioned, type. For example, the partition Header of the type Record would have this component path: **Header<>Record**

Component path

A component path is specified as an optional partition list, followed by a local type name and then followed by an optional index. Here are some examples of component paths:

```
Grand<>Ball Room[5]
Ball Room[LAST]
Really<>Grand<>Ball Room
Ball Room
```

Indexed object names

An object name can refer to a particular occurrence of a data object. The index of the occurrence appears in square brackets immediately after its name. For example, the object name of the third **Note** would be:

Note[3]

The index between the square brackets can be an integer or the reserved word, "LAST" (a special index value that refers to the last data object of a particular series). The index cannot be another object name. To use a variable index, use the **CHOOSE** function.

LAST is interpreted in the context in which it is used.

- [Using LAST in a map rule to reference an input](#)
- [Using LAST in a component rule](#)
- [Using LAST in a map rule to reference an output](#)

Using LAST in a map rule to reference an input

In a map rule, when referencing an input, LAST refers to the very last occurrence of that input. For example, the last **Note** of the input **Report** would have the name:

Note[LAST] :Report

In this context, all of the input has been validated before map rules are evaluated.

Using LAST in a component rule

In a component rule, LAST refers to the last occurrence found. For example, in the rule:

PO# Field:Order = PO# Field:Order[LAST]

LAST refers to previous occurrence of **Order** because when this rule is evaluated, the last known **Order** is the last one found.

Using LAST in a map rule to reference an output

In a map rule, when referencing an output, LAST refers to the last built occurrence of that object. For example, using the same expression:

PO# Field:Order = PO# Field:Order[LAST]

LAST refers to the previous occurrence of **Order** because, when this rule is evaluated, the last known **Order** is the last one built.

LAST is a reserved word and is not case sensitive.

Component paths separated by a colon

A component in the component list of another object is always referenced by a colon (:). The local type name of the nested component precedes the colon. The name of the object that contains the component follows the colon. For example, the **Line Item** component of **PO** would have this name:

Line Item:PO

The **Last Name** component of **Client**, which is in turn a component of **Account**, would have this name:

Last Name:Client:Account

Component paths separated by IN

To reference all occurrences of an object contained in another object, use the keyword **IN**.

For this example, **ShipSchedules** appears as a component of different components within **OrderAck**.

The object name for all occurrences of **ShipSchedules** within **ORD_ADR_OUT_WD Record** is:

ShipSchedules IN ORD_ADR_OUT_WD Record:OrderAck

When **IN** is used, the contained object name must refer to the same type. In the example, the **ShipSchedules** component of **ORD_ADR_OUT_WD Record** and the **ShipSchedules** component of **OrderLine Structure** must be of the same type.

In the Map Designer, you can refer to all occurrences of an object contained in a card object. The object name for all occurrences of **ShipSchedules** within **OrderAck** is:

ShipSchedules IN OrderAck

- [Referring to all occurrences with IN COMPONENT](#)

Referring to all occurrences with IN COMPONENT

In the Type Designer, you can refer to all occurrences of an object contained in a component list. To refer to all components in a component list, up to a given component, the word **COMPONENT** is used.

For example, the rule on the component **Conclusion Topic** needs to count all of the valid **Topics** up to, and including, **Conclusion Topic**.

The **IN COMPONENT** expression can be used:

#Topic Field:\$ = COUNT (Topic IN COMPONENT)

The above component rule works only if **Topic** is partitioned.

Comment object name

A comment name can be either an in-comment name or an @-comment name.

An in-comment name references all comment objects within a specified object. An in-comment name is specified as a component-path, a space, the reserved word **IN**, followed by another space, followed by the component path of the component on which the floating component type is defined. For example, given the data structure shown below, to reference all **INPUT Parameters** for a particular **Input**, regardless of whether they followed **Method OpenFile** or **Method GetEntry**, the object name would be **INPUT Parameters IN Input**.

An @-comment name references all comment objects that follow immediately after a specified object. An @-comment name is specified as the component path of the comment, followed by the reserved symbol "@" , followed by the component path of the component that follows the comment. For example, to reference only the **INPUT Parameters** that follow immediately after the **Method OpenFile** (and before **Method GetEntry**), the object name would be **INPUT Parameters @ Method OpenFile:Input**.

Shorthand notation

In a rule, a dollar sign (\$) can be used to refer to the object to the left of the rule cell. In the Type Designer, the \$ sign refers to the object you are qualifying. In the Map Designer, the \$ refers to the object to which the rule evaluates.

In certain cases, you can also use a period between colons to shorten the reference to a data object name. The **Use Ellipses** command can be used to automatically shorten data object names you drag into rule expressions.

- [Using the dollar sign \(\\$\)](#)
- [Using ellipses](#)

Using the dollar sign (\$)

The dollar sign can be used to refer to the object to the left of a rule cell.

For example, the component rule below applies to the component **RunRule**.

The **RunRule** in the rule can be replaced by **\$**. You can remember it this way: a dollar sign always means "whatever component appears to the left of the rule".

Here is an example of a Map Designer rule that uses the dollar sign:

```
Order:Invoice = MapMyOrder ( Order:PO , Index ( $ ) )
```

In this example, each time this rule is evaluated, the index of the **Order:Invoice** being built is the object referred to by the **\$**.

Using ellipses

A single period can be used, like ellipses, in place of object names, as long as that object name can be interpreted as a unique object.

For example, if you have the following object name:

```
City Field:BillTo Customer:Account:File
```

you can replace it with

```
City Field:..:File
```

If file also contains another **City Field**, such as

```
City Field:ShipTo Customer:Account:File
```

and you use the **Use Ellipses** option, the system returns

```
City Field:BillTo Customer:..:File
```

Evaluating expressions

Expressions are evaluated based on the context in which data object names, literals, operators, function names and map names appear in the expression, and the data on which it operates. A component rule always evaluates once for each occurrence of a component. A map rule can evaluate many times.

When a map rule evaluates, the Map Designer selects an evaluation set of values and then evaluates the rule. Then it selects another evaluation set of values, if there is one, and evaluates the rule again. This continues until all the different evaluation sets have been used.

The same number of evaluations is always produced, but output objects can be ordered differently, depending on how you define your map.

For a selected evaluation set, a particular instance of a rule is evaluated using parentheses, operator precedence rules, and left-to-right precedence rules to get one result.

- [Card order can influence the order of evaluation sets](#)
- [Functions influence the number of evaluation sets](#)
- [Object names influence the number of evaluation sets](#)

Card order can influence the order of evaluation sets

When an evaluation set is selected, card object names are ordered by their position in the map: card 1 before card 2, and so on. The card order for a map can affect the order of output results.

For example, suppose you want to evaluate the following map rule.

```
NumberSet ( s ) = B:Card2 + A:Card1
```

In this example, an evaluation set for this rule consists of one value for B and one value for A. One evaluation set produces one **NumberSet**.

As an example, here there are three As and three Bs in the data:

A Values	B Values
1	3
2	2
3	1

There are nine evaluation sets. The order of the cards might affect the order of the evaluation sets.

If Card 1 is the first card, you get the following results:

Evaluation #	A	B	Result
1	1	3	4
2	2	3	5
3	3	3	6

Evaluation #	A	B	Result
4	1	2	3
5	2	2	4
6	3	2	5
7	1	1	2
8	2	1	3
9	3	1	4

If Card 2 is the first card, you get the following results:

Evaluation #	B	A	Result
1	3	1	4
2	2	1	3
3	1	1	2
4	3	2	5
5	2	2	4
6	1	2	3
7	3	3	6
8	2	3	5
9	1	3	4

If both A and B are contained in the same card object, evaluation sets are selected based on the order A and B appear in the rule. Leftmost objects are selected first. If you change the rule to this:

```
NumberSet ( s ) = B:Card2 + A:Card2
```

the same results are produced as when card 2 is the first card.

Functions influence the number of evaluation sets

For this example, you have the following rule:

```
NumberSet ( s ) = A:Card1 + SUM ( B:Card1 )
```

Using the same values for A and B, there are only three evaluation sets as shown.

NumberSet[1] = 7

A Values	B Values
1	3
2	2
3	1

NumberSet[2] = 8

A Values	B Values
1	3
2	2
3	1

NumberSet[3] = 9

A Values	B Values
1	3
2	2
3	1

If your rule looks like:

```
NumberSet ( s ) = SUM ( A:Card1 ) + SUM ( B:Card1 )
```

there is one evaluation set: NumberSet[1] = 12

A Values	B Values
1	3
2	2
3	1

If you have the following rule:

```
NumberSet ( s ) = A:Card1 + EXTRACT ( B:Card2 , B:Card2 != 2 )
```

the order of the operands does not affect the order of evaluation set selected. This is because an A is selected, then the **EXTRACT** is evaluated, then evaluation sets are determined for that A and a given **EXTRACT** result. When all **EXTRACT** results are used, the next A is selected, and so on.

Given the same data, this time you get the following results:

Evaluation #	A	B Extract Values	Result
1	1	3	4
2	1	1	2
3	2	3	5
4	2	1	3
5	3	3	6
6	3	1	4

As another example, the following map rule applies to the output Status(s):

```
= IF ( Quantity:Order Record:Order > 10 &
OR ( Store:Order Record:Order = "A" ) ,
      "Accept" , "Reject" )
```

The first argument of the **IF** function is an expression. One evaluation set of this expression requires:

- the **Order**
- one **Order Record**
- one **Quantity** of the selected **Order Record**
- all the **Stores** of the selected **Order Record**

The **IF** function evaluates once for each **Quantity** of each **Order Record**. Other than the first argument of the **IF** function, nothing else in the map rule affects the number of times the map rule will be evaluated. So, if there are 1000 **Quantity** data objects contained in **Order**, this map rule will evaluate 1000 times.

Object names influence the number of evaluation sets

In some expressions, the same data object name might appear more than once. The same data object name always refers to the same data object. The same data object name influences the combinations of evaluation sets that are selected. For example:

```
BackOrder(s) = Qty Ordered:Record:Order - Qty Received:Record:Order
```

When a **BackOrder** evaluation set is selected, the **Record** that contains **Qty Ordered** is always the same **Record** that contains **Qty Received**. And an **Order** is always the same one for one evaluation set. When the same object name is used more than once in the same expression, both references bind to the same object.

If there is always one **Qty Ordered** per **Record** and one **Qty Received** per **Record**, the following would be sample data:

Record #	Qty Ordered	Qty Received
1	5	4
2	4	4
3	6	2

This would produce three **Back Orders**, one for each **Record**.

You can coordinate nested data objects easily by the way an object name is used.

- [Using object names to coordinate evaluation sets](#)
- [Using IN to decouple object name coordination](#)
- [Using IN to decouple different partitions in the same rule](#)

Using object names to coordinate evaluation sets

In the following example, common object names are used to coordinate **Qty** and **UnitPrice** so they do not get mixed up across different **Records**:

```
Extension = SUM ( Qty:Detail Record:Order * UnitPrice:Detail Record:Order )
```

This rule has only one evaluation set because the **SUM** consumes all the objects referenced in the rule.

Using IN to decouple object name coordination

Suppose you have a situation in which you do not want to coordinate common objects. You might, for example, want to count different objects contained in the same object, as in:

```
Summary = COUNTabs ( Header:Order:Message ) + COUNTabs ( Line Item:Order:Message )
```

If there are multiple **Orders** in your data, the **Summary** would not be evaluated correctly. This rule would only count the **Headers** of the first **Order** and add it to the count of all Line Items of the first **Order**. To count all the **Headers** and all the **Line Items** in all the **Orders**, use IN:

```
Summary = COUNTabs ( Header IN Message ) + COUNTabs ( Line Item IN Message )
```

Using IN to decouple different partitions in the same rule

If data objects in an expression are different partitions of the same type, the expression might evaluate to "none".

For example:

```
MyMap ( Good<>Design , Best<>Design )
```

Good and **Best** are partitions of the same type, **Design**. A given **Design** could never be both a **Good<>Design** and a **Best<>Design**; therefore, the second argument of **MyMap**, **Best<>Design**, evaluates to "none".

To reference both the **Good** and **Best** partitions, use IN:

```
MyMap ( Good IN Design , Best IN Design )
```

Operators

Operators are used for arithmetic functions and text functions on items. If an operator requires two operands, the operator symbol appears between them. For example, in the expression **a + b**, the operator symbol "+" appears between the operands **a** and **b**.

Operators are reserved symbols that cannot be used in type names.

- [Arithmetic operators](#)
 - [Text operators](#)
 - [Logical operators](#)
 - [Comparison operators](#)
 - [Order of operator evaluation](#)
 - [Operands](#)
-

Arithmetic operators

Operator	Description
+	addition
-	subtraction
*	multiplication
/	division

Text operators

Operator	Description
+	concatenation

Logical operators

Logical operators are used in expressions that evaluate to "true" or "false".

Operator	Description
^	exclusive or
&	and
	or
!	not
=	equals

Comparison operators

Operator	Description
=	equal to
>	greater than
<	less than
>=	greater than or equal to
<=	less than or equal to
!=	not equal to

NOT=

alternative for not equal to

The results of comparison operators are based on the native collating sequence of the machine used to run the map.

Order of operator evaluation

Use parentheses to group operations. Expressions within parentheses are evaluated before performing other operations. For example:

`(UnitPrice - Discount) * Tax`

The expression **UnitPrice - Discount** is evaluated before the rest of the expression.

Some operators are evaluated before others according to standard rules of precedence. Operators are evaluated in the following order:

Precedence	Operators	Description
first	()	Operations within parentheses
second	+ and -	unary plus and minus The sign indicates a positive or negative expression, for example: - quantity*rate
third	=, <, >, <=, >=, !=	comparative operators
fourth	^	logical operator "exclusive or"
fifth	&	logical operator "and"
sixth		logical operator "or"
seventh	* and /	multiplication and division
eighth	+ and -	addition and subtraction

Operators of equal precedence are evaluated left to right.

Operands

Operands are the arguments for an operator. For example, in the expression **a + b**, the **+** is the operator symbol, and **a** and **b** are the operands.

An operand can be any of the following types:

- a literal
- a data object name
- a function
- another operator

Using functions in expressions

Functions perform a particular action on its input arguments. A function is written like this:

`FUNCTION (argument1, argument2, ... argumentn)`

The arguments of the function appear inside the parentheses, separated by commas. A function might require a fixed number of arguments or might allow a varying number of arguments. You can use functions anywhere in an expression.

- [Function arguments](#)
- [Input arguments](#)
- [Nested input arguments](#)
- [Output arguments](#)
- [Function arguments and evaluation](#)

Function arguments

Input arguments themselves can be expressions. Some functions limit the type of expression that can be used as an input argument. The syntax specification tells you the number of data objects that can be used for one function evaluation.

There is *always* one output argument for a function. The specification of the output argument tells you: 1) the type of the result produced by a function and 2) the number of objects that can be produced when the function evaluates once.

You need to know the type of the output argument because functions can be used as other arguments. The result of a function can also be used to directly produce an output data object contained in an output destination.

When the Map Designer analyzes the interfaces of expressions in rules during the build process, it ensures that the output argument of a function matches the input argument it is used for. The analysis also checks the type of the output argument when that function is used to directly produce an output data object.

The output argument of a function specifies the result that the function produces for one evaluation. For example, one evaluation of the ABS function produces one positive number. One function evaluation might produce multiple outputs. If it does, the expression that contains that function might produce multiple evaluation sets.

Input arguments

The input arguments for a function specify the information the function uses to return a result. Some functions require an exact number of input arguments. For other functions, the number of input arguments can vary. Optional arguments for a function are shown in brackets [].

To correctly use a function, you must use the correct arguments in the correct order.

Each argument can, itself, be an expression. In general, an input argument might be any of the following:

- an object name
- a literal
- a function-name and its arguments
- an operator and its arguments
- an enumerated series of literals, such as {a, b, c}
- a map name and its arguments

For example, some valid expressions that use the **ABS** function are:

```
ABS ( Quantity:LineItem )
ABS ( UNIQUE ( Quantity:LineItem ) )
ABS ( Quantity:LineItem - 300 )
```

Each function has its own expression syntax for its input arguments. For example, the **ABS** function cannot use a map as an argument. See [Syntax of a Function](#) for notation used to specify valid expression syntax.

Nested input arguments

When a function, operator, or map is used as an input argument, each of these has arguments of its own. Arguments of other arguments are said to be *nested*. For example, in the use of the **ABS** function, you can use a function as an argument:

```
ABS ( UNIQUE ( Quantity:LineItem ) )
```

The output of the **UNIQUE** function becomes the input argument to the **ABS** function.

Output arguments

The output argument of a function indicates the type of object the function produces and the number of objects a function can produce. An output argument is specified as one of the following object types or values:

- A single data object of a type
- A series of data objects of the same type
- "True" or "False"

If the output of a function is a series, that function can be used only in a map rule in the Map Designer. It cannot be used in a component rule in the Type Designer.

For example, you cannot use the **CLONE**, **EXTRACT**, **REJECT**, **SORTDOWN**, **SORTUP** or **UNIQUE** functions in the Type Designer.

Function arguments and evaluation

When a function is evaluated, the number of argument objects used for one evaluation depends on the function. In the specification of the syntax for a function, the number of input objects that can be used for each argument for one function evaluation is expressed as "single" or "series".

Some functions use a single object as the value of an argument for one evaluation. For example, the **ABS** function uses a single object as the value of its argument for one evaluation.

```
ABS ( Quantity:LineItem )
```

When a single object is used for one evaluation, the function itself can evaluate many times if there is more than one data object that fits the argument definition. For example, if there were ten **Quantity:LineItem** data objects, the **ABS** function could evaluate ten times.

Some functions use an entire series of objects as the value of an argument for one evaluation. For example, the **COUNT** function uses an entire series of data objects as the value of its argument for one evaluation.

```
COUNT ( LineItem:PO )
```

When a series of data objects is used for one evaluation, the function evaluates just once when there is more than one data object that fits the argument definition. For example, if there are ten **LineItem:PO** data objects, the **COUNT** function would evaluate only once.

Some functions use an entire series as input and produce a series as output. For example, the evaluation set for the **EXTRACT** function produces a series from an input series. When a function produces a series, each output can be selected for different evaluation sets of the expression that contains that function.

Consider the following expression:

```
Line Item(s) = EXTRACT ( Line Item:Order, Qty:Line Item:Order > 1000 )
```

Each **Line Item** that fits the specified criteria produces a **Line Item** of **Order**. In this example, the map rule is evaluated many times, once for each **Line Item** produced by the **EXTRACT** function.

When functions are part of other expressions, the number of evaluation sets for that function depends on the object names used in the entire expression. For example:

```
Debit ( s ) = ABS ( Debit:Account:Input )
```

There might be many evaluation sets for the above expression. The ABS function might be evaluated many times, once for each **Debit** of each **Account** of **Input**. However, if the ABS function is part of a more complex expression, common objects of the expression might determine the number of ABS evaluations.

Map names in expressions

Map names in expressions that are used in map rules represent calls to functional maps.

The syntax of a functional map call in an expression is the same as the syntax of a function:

```
FunctionalMapName ( argument1 , argument2 , ... argumentn )
```

At run time, when a rule calls a functional map, the functional map writes the output, in the format defined by its output card type, directly to the output of the executable map. The functional map also places the information about the type or types that it builds in the output work file of the executable map, if necessary. So the type that the functional map builds is the type of the output card of the functional map, which must match the output type of the rule on which the functional map call is placed.

When the output type of the functional map is a group, map build analysis verifies that the type of the output card of the functional map matches the output type of the rule on which the call is placed. When the output type of the functional map is an item, map build analysis checks that the item classes match, but it does not require that the type of the output card of the functional map be the exact same type as the output type of the rule on which the functional map call is placed. Because of this way that map build analysis operates, functional maps can behave more like subroutines. You can use the same functional map to build various output objects that all have the same characteristics, as long as you do not try to reference the output objects of the calling rules in later rules.

A functional map does not behave like a function because when the functional map completes processing, it has already written its output to the output of the calling rule. That calling rule cannot access the data that the functional map built. This behavior is different from the behavior of a function call, which returns the value that the function produced to the point of invocation. The calling rule can manipulate the value that a function returned. The calling rule cannot manipulate the value that a functional map produced.

When the output of a functional map is an item, you can use the **ASFUNCTION** general function to return the output to the invoking map rule instead of to the output card.

Since the type that the functional map builds is the type of the output card of the functional map, later references to the output type of the rule, in subsequent rules, cannot work properly unless the type of the output card of the functional map exactly matches the output type of the rule. If the rule is on some type, for example, A, and the output type of the functional map is some different type, for example, B, the object that is reflected in the work file for the rule is B. If a subsequent rule tries to reference the instance of type A, that reference returns **NONE**.

The following examples show two ways, one that is valid and one that is not valid, to specify map rules that use map names in expressions. For both examples, the names of the functional maps are prefixed by **F_**.

Valid example:

```
=IF ( A > 10 , F_BigMap ( X ) , F_LittleMap ( Y ) )
```

This example is valid because the rule does a comparison on an item, and then calls one of two functional maps, which sends the results directly to the output of the executable map. If the value of A is greater than **10**, the rule calls the **F_BigMap** functional map; otherwise, the rule calls the **F_LittleMap** functional map.

Not valid example:

```
=IF ( F_Map ( A ) > 10 , F_BigMap ( X ) , F_LittleMap ( Y ) )
```

This example is not valid because the rule attempts to use the output of the **F_Map** functional map in a comparison. The rule calls the **F_Map** functional map, and after **F_Map** completes processing, it writes the results directly to the output of the executable map. The value that **F_Map** produced is not available to the calling rule; therefore, the rule cannot perform the comparison operation.

- [Evaluation of functional maps in an expression](#)

Evaluation of functional maps in an expression

A single evaluation of a functional map requires one object for each input argument. The result of the evaluation is always one output object. As with functions and operators, when a functional map is used in an expression, evaluation sets for that expression might cause the functional map to be evaluated many times.

Consider the following expression:

```
MakeForm ( EntryForm:Input , GroupInfo:Input )
```

If there are two occurrences of **GroupInfo** and four occurrences of **EntryForm**, the map **MakeForm** will be evaluated once per combination of **EntryForm** and **GroupInfo**-a total of $(2 \times 4) = 8$ times.

Consider another expression:

```
Fmap ( a:Input , b:Input , c:Input )
```

The objects **a**, **b**, and **c** have only **Input** in common. The expression that contains the **Fmap** reference will be evaluated once per combination of **a**, **b**, and **c**.

To illustrate how the functional map will be evaluated, use a very simple data file of **a**'s, **b**'s, and **c**'s. Each time the map is evaluated, one **a** object, one **b** object, and one **c** object are selected. The first time the map is evaluated, the data objects are **a1**, **b1**, and **c1**. The next time it is evaluated, the data objects are **a2**, **b1**, and **c1**, and so on.

An important difference in how the Map Designer evaluates a functional map in an expression is when an input argument evaluates to **none**.

See [When an Input Argument of a Function Evaluates to NONE](#).

Expressions that evaluate to NONE

The following discussion explains how the Map Designer evaluates an expression when expressions within it evaluate to "none".

- [When an operand evaluates to NONE](#)
- [When an input argument of a function evaluates to NONE](#)
- [When an input argument of a functional map evaluates to NONE](#)
- [When an input of an executable map evaluates to NONE](#)
- [Impact of track setting on order of output](#)

When an operand evaluates to NONE

The following table explains how an operator expression evaluates when an operand evaluates to none.

The symbol **A** represents one operand of the expression and **B** represents the other operand. The symbol **!=** means "not equal to".

Operator Expression	Condition	Evaluates to:
A / B	A = NONE B = NONE	NONE
A*B	A = NONE B = NONE	NONE
A / B	A != NONE B = NONE	0
A*B	A != NONE B = NONE	0
A + B	A = NONE B = NONE	NONE
A - B	A = NONE B = NONE	NONE
A + B	A = NONE B != NONE	B
A - B	A = NONE B != NONE	-B
A + B	A != NONE B = NONE	A
A - B	A != NONE B = NONE	A
A > B	A = NONE B = NONE	FALSE
A < B	A = NONE B = NONE	FALSE
A = B	A = NONE B = NONE	TRUE
A > B	A = NONE B != NONE	FALSE
A < B	A = NONE B != NONE	TRUE
A = B	A = NONE B != NONE	FALSE
A > B	A != NONE B = NONE	TRUE
A < B	A != NONE B = NONE	FALSE
A = B	A != NONE B = NONE	FALSE

For example, in this operator expression:

```
Total = #Guests + 1
```

if **#Guests** evaluates to "none", the value for **Total** will be 1.

When an input argument of a function evaluates to NONE

If any input argument of a function evaluates to "none", the function might not evaluate to "none". For example, **ABS(NONE)=NONE**, but **COUNT(NONE)=0**.

When an input argument of a functional map evaluates to NONE

If any argument of a functional map evaluates to "none", the functional map will not be evaluated for that evaluation set.

For example:

```
MakeForm ( EntryForm:Input , GroupInfo:Input )
```

Where the object **GroupInfo** is optional; it has a component range of (0:3). If there are no occurrences of **GroupInfo** in the **Input** data object, the map **MakeForm** will not be evaluated at all.

Here is another example:

```
OrderMap ( OrderRecord:Order:Input , SummaryRecord:Order:Input )
```

If **SummaryRecord** has a component range of (0:1) and it is missing in a particular **Order**. The map **OrderMap** will not be evaluated for that **Order**.

If you want **OrderMap** to be evaluated even when **SummaryRecord** is missing, use the entire **Order** as an input argument, rather than **SummaryRecord**. This will ensure that **OrderMap** will be evaluated, even if **SummaryRecord** is missing. Then, in the map **OrderMap**, you can still map from **SummaryRecord**.

The rule you should use is this:

When an input of an executable map evaluates to NONE

The evaluation of a map referenced in an expression differs from the evaluation of that map when it is executed as a functional map and the evaluation of a function because of how the NONE is treated.

The evaluation of a map used as a function is different from the evaluation of that map when it is run as an executable map. When a map is run as an executable, if any input card data object has no content, the map will still run. This is not true when that map is referenced as a function. If any input argument of a functional map evaluates to "none", the map will not evaluate.

Impact of track setting on order of output

This property setting determines whether objects are tracked according to content or according to their position within a series. This setting will determine the sequence in which those objects are evaluated when referenced.

For example, assume that **RunsByInning** is a component of **BallGame**. **BallGame** is defined as an explicit format, delimited (by a colon) group. The data for **BallGame** is as follows:

`2::1:1::1::3`

- **Track Content.** If **BallGame** has a **Track** setting of **Content**, all instances of **RunsByInning** with content will be mapped before any empty instances. So, if **BallGame** was mapped to itself, the output would be:
- `2:1:1:1::3`
- **Track Places.** If **BallGame** has a **Track** setting of **Places**, instances of **RunsByInning** will be mapped in the sequence in which they occur, including those instances that are empty. So, if **BallGame** was mapped to itself, the output would match the input.

`2::1:1::1::3`

Track Places will only have an impact on the sequence of the output when *both* the input group object and output group object are defined as **Track Places**.

However, **Track Places** will have an impact in situations in which an input defined with **Track Places** is referenced by index. For instance, in the above example,

- If **RunsByInning** was defined as **Track Places**, **RunsByInning[2]:BallGame** would have a value of **2** because the value in the second "place" in **BallGame** is a **2**.
- On the other hand, if **RunsByInning** was defined as **Track Content**, **RunsByInning[2]:BallGame** would have a value of **1** because the second instance of **BallGame** with content has a value of **1**.

Evaluated expressions assigned to output items

NONE assigned to an output number

When an output item is defined as a number, its value is determined from the result of the evaluated expression that is then converted to its output form as follows:

Output is specified as:	Evaluated value	Output value
Required	0	0
Required	NONE	0
Optional	0	0
Optional	NONE	NONE

Automatic item format conversions

When an item is assigned an output value, the Map Designer will automatically convert an item with the same interpretation (either number, text, date, or time) from one presentation to another. For example, if you want to map an item that is a number and whose presentation is integer to an item that is also a number but whose presentation is decimal, the Map Designer will automatically convert it.

The Map Designer fills in missing information in date and time conversions in the following way:

- **Dates with a year, but no century.** The century will be determined using the **CenturyDerivation** map setting.
- **Missing part of an output date format.** If a portion of the format of an output date cannot be derived from the input date, the missing part will have pound characters (#) in its place. For example, if **09/98** is the date for an input with a format string of **MM/YY** and that date was mapped to an output date with a format of **MM/DD/YY**, the output date would be built as **09/##/98**.
- **Missing part of an output time format.** If a portion of the format of an output time cannot be derived from the input time, the missing part will have zeroes (0) in its place. For example, if **12:14** is the data for an input with a format string of **HH12:MM** and that time was mapped to an output time with a format of **HH:MM:SS**, the output date would be built as **12:14:00**.

To convert a number to text, a date or time, or vice versa, you must use a function to perform the conversion. For example, to convert a date to text, use the function **DATETOTEXT**.

- [Numeric precision](#)

Numeric precision

All numbers in the input and numeric literals in map rules are converted to multiple precision and converted to the format required at output time.

- If the multiple precision number does not overflow but does not fit into the maximum size of a numeric item, that item is filled with the number sign character (#) up to its maximum size.
- An arithmetic overflow is recognized when an arithmetic expression includes:
 - Divide by zero
 - A multiple precision number greater than 1.0E254

Arithmetic overflow is designated in the output object with up to a maximum of ten number sign (#) characters.

Using functions

A function is an expression that generates an output by performing a certain operation on one or more inputs. Most functions can be used in both component rules and map rules. Those that produce a series can only be used in map rules.

- [Functions in a component rule](#)
- [Functions in a map rule](#)

Functions in a component rule

For example, the object **TotalQty** in your data is the sum of all the **Qty** objects in your data; in a component rule for **TotalQty**, you can use the SUM function to verify this relationship.

Functions in a map rule

Suppose you want to map your data differently according to the presence of a certain input data object. In your map rule, to check the presence of the data object, use the function PRESENT. To create the output according to whether the input data was present, use the function IF. Examples of the IF function can be found in "[Logical Functions](#)". Examples of the PRESENT function can be found in "[Inspection Functions](#)".

Syntax of a function

The following is an example of a function:

```
FUNCTION ( argument1, argument2, ... argumentn )
```

When you use a function, substitute the name of a specific function for FUNCTION and substitute a specific expression for each input argument. Each input argument is separated by a comma and the set of arguments is surrounded by parentheses.

Function argument syntax

Arguments typically have some restrictions on the expression used for that argument. For example, the input argument to the INDEX function can only be an object name. Depending on the function, one or many data objects can be used for an input argument for one function evaluation.

For each function listed in this documentation, the input argument has a syntax definition that indicates:

- the number of objects that can be used for each argument for one function evaluation, expressed as either **single** or **series**

followed by

- the type of expression that can be used for each argument.

For the output argument, the syntax specification is similar to the input argument syntax, except that the output is always used as a specific type of object, not an expression.

For example, single-item-expression means one evaluation operates on a single item that can be defined as an item expression. Series-item-expression means one evaluation operates on one or more input argument items that can be defined as an item expression.

In this documentation, the terms listed in the following table are used to describe the syntax of function arguments.

Term

Explanation

general-expression

Any valid combination of object names, literals, operators, function names, and map names.

condition-expression

Any valid combinations of object names, literals, conditional operators, and functions whose output arguments are specified as true or false.

item-expression

Any valid combination of object names, literals, operators, and functions whose output argument is specified as an item.

If an item must be interpreted as a date, time, or number, the item is referenced by its interpretation. For example, if an item-expression must be text, it is specified as text-expression.

object-expression

An object name or a series producing function.

If an object expression is further limited to be an item, it is specified as item-object-expression.

object-name

The name of a data object, as defined in [Data Object Names](#). If an object name is further limited, it is specified as a prefix. For example, simple-object-name refers to a simple type name.

...

Indicates that the function can take on any number of additional similar arguments.

[]

Indicates that the argument is optional.

{ literal , literal ...}

An enumerated series of literals.

The syntax specification shows each input argument, using terms as described in the previous table, preceded by the word **single** or **series**. For example:

Syntax:

ALL (series-condition-expression)

The arguments are defined using a name that describes its use by the function. A simple example is:

Meaning:

DATETONUMBER (date_to_convert)

Function names are shown in uppercase letters in this documentation. However, function names are not case sensitive.

There is also a description of what each function returns.

Returns:

A single integer

Functions that convert character-set encoding to the native code page

Functions that use the IBM Transformation Extender **EXIT** architecture convert data to the native character set of the platform as part of the function call.

These functions use the IBM Transformation Extender **EXIT** architecture:

- **DBLOOKUP**
- **DBQUERY**
- **DDEQUERY**
- **EXIT**
- **JEXIT**
- **GET**
- **PUT**
- **RUN**
- **XQUERY**
- **XSLT**
- **MATHLIB** functions
- **XMLLIB** functions (except **XPATH** and **XPATHEX**)
- **RESOURCELIB** functions
- Customer-developed functions that use the **EXIT** architecture

Maps that use these functions must take the data conversion into account. For example, a map that uses the **EXIT** architecture converts non-EBCDIC data to EBCDIC when the map runs on a z/OS® platform. To prevent the data conversion, the map must use a function like:

`CTEXT(object, "Native")`

If the function returns the data from the z/OS platform to another platform, the map must specify that the data to return is non-EBCDIC by using a function like:

`CTEXT(object, "data language")`

General functions

• **ASFUNCTION**

When you run a functional map with the **ASFUNCTION** function, the output of the functional map returns to the map rule that invoked the functional map. The invoking rule can work with the output from the functional map. The type of the output card must be an item.

• **CLONE**

The **CLONE** function creates a specified number of copies of some object.

• **DEFAULT**

The **DEFAULT** function allows a predefined value to be introduced into an expression.

• **ECHOIN**

The **ECHOIN** function returns a command (or property) to be used in a **RUN** function argument.

- **HANDLEIN**
The **HANDLEIN** function returns a command (or property) to be used in a RUN function argument.
 - **PARSE**
The **PARSE** function analyzes a data object and validates it against the metadata of a component type. The resulting data structure is available to the calling map for further processing.
 - **REFORMAT**
The **REFORMAT** function returns a type object that results from replacing the syntax of the input type with the syntax of the output type.
 - **RESOLVETYPE**
The **RESOLVETYPE** function resolves the provided component path to a specific type tree reference when the map is being compiled.
-

ASFUNCTION

When you run a functional map with the **ASFUNCTION** function, the output of the functional map returns to the map rule that invoked the functional map. The invoking rule can work with the output from the functional map. The type of the output card must be an item.

Without **ASFUNCTION**, a functional map writes to the output card before the rest of the invoking rule is evaluated.

If **ASFUNCTION** runs a functional map that calls a second functional map, the output of the second map also returns to the map rule, as if the second map had been invoked by **ASFUNCTION**. If the output of each map is a number and the outputs are concatenated, **ASFUNCTION** adds the numbers instead of concatenating them.

Syntax:

ASFUNCTION (*general-expression*)

Meaning:

ASFUNCTION (*FunctionalMapName* (*argument-1*, *argument-2*, ... *argument-n*))

Returns:

item

Example 1

Rule:

```
Output_name = FMAP1 (rule) + " and " + Fmap2 (rule) + "."
```

Result:

```
Output_of_FMap1Output_of_FMap2 and .
```

Rule:

```
Output_name = ASFUNCTION(FMAP1 (rule)) + " and " + Fmap2 (rule) + "."
```

Result:

```
Output_of_FMap1 and Output_of_FMap2.
```

Example 2

Rule:

```
Output_name = IF (FMap1 (rule) = "abc", "equal", "not equal")
```

Result:

- If FMap1 produces **abc**:

```
abcnot equal
```

- If FMap1 produces **def**:

```
defnot equal
```

Rule:

```
Output_name = ASFUNCTION(IF (FMap1 (rule) = "abc", "equal", "not equal"))
```

Result:

- If FMap1 produces **abc**:

```
equal
```

- If FMap1 produces **def**:

```
not equal
```

Example 3

Rule:

```
Output_name = ASFUNCTION(FMAP1(rule1)) + FMAP2(rule2)
```

Result:

- If FMAP1 produces 1 and FMAP2 produces 2:

- If FMAP1 produces A and FMAP2 produces B:

AB

CLONE

The **CLONE** function creates a specified number of copies of some object.

This function can be useful when the number of output objects to be built depends on a data value, rather than the number of objects that exist in the data.

Syntax:

CLONE (single-object-name , single-integer-expression)

Meaning:

CLONE (object_to_copy , number_of_copies)

Returns:

A series-object

The **CLONE** function returns a series of the object specified by *object_to_copy*. The output series consists of as many copies of the object as specified by *number_of_copies*. The value of each member of the resulting output series is the same as *object_to_copy*.

Examples

- LineItem Segment (s) = LotsOf (**CLONE** (Detail Row , Quantity:Detail Row))

In this example, the input contains an object named **Detail Row** that has a **Quantity** item component. You want to create multiple **Line Item Segments** for every **Detail Row** in your input based on the value of **Quantity**. So, if **Quantity** has the value 5, you want to build five **Line Item Segments** for that **Detail Row**.

DEFAULT

The **DEFAULT** function allows a predefined value to be introduced into an expression.

DEFAULT, which is typically used with the **EITHER** function, returns the value assigned as the default value when the type was defined.

Syntax:

DEFAULT(single-type-expression)

Meaning:

DEFAULT(type_whose_defined_default_value_is_desired)

Returns:

A single-object

Examples

- **DEFAULT(ZipCode)**

In this example, the defined default value for ZipCode is returned. When used with **EITHER**, **DEFAULT** might appear like this:

EITHER(ZipCode,DEFAULT(ZipCode))

If the first argument evaluates to a value other than "none", that value is used. If the first argument evaluates to "none", the default value for **ZipCode** is used.

ECHOIN

The **ECHOIN** function returns a command (or property) to be used in a **RUN** function argument.

Syntax:

ECHOIN (single-integer-expression, single-text-expression)

Meaning:

ECHOIN (card#, item_or_group_interpreted_as_text)

Returns:

A single object

Examples

Before **ECHOIN** was introduced, the **RUN** function was used in this way:

```
RUN ("MyMap", "-IE1S"+NUMBERTOTEXT (SIZE(Data))+ "
" + TEXT(Data)))
```

You can use **ECHOIN** with **RUN**, similar to the following example:

```
RUN
("MyMap", ECHOIN(1,(Data)))
```

Related functions

HANDLEIN

The **HANDLEIN** function returns a command (or property) to be used in a RUN function argument.

You can use the **HANDLEIN** function when deriving large amounts of data from a parent map for use in a RUN map. The **HANDLEIN** function works by using the same instance of the parent map's data set in the RUN map, instead of duplicating it as the **ECHOIN** function does.

The result of the **HANDLEIN** function is a text object that defines the characteristics of the data to be used. This includes an internal handle, offset, and length of the data. The result of this can be seen in an execution audit log:

```
<ExecutionSummary MapStatus="Valid" mapreturn="0" ElapsedSec="0.1803"
BurstRestartCount="0">
  <Message>Map completed successfully</Message>
  <CommandLine>install_dir\sdq20.mmc -IH1 4:512:1024 -ae</CommandLine>
  <ObjectsFound>85</ObjectsFound>
  <ObjectsBuilt>23</ObjectsBuilt>
<SourceReport card="1" adapter="Handle" bytes="1024" adapterreturn="0">
  <Message>Success</Message>
  <Settings>4:512:1024</Settings>
  <TimeStamp>05:06:07 January 27, 2004</TimeStamp>
</SourceReport>
```

You cannot use a **GETANDSET** function against an input that is passed using **HANDLEIN**.

Syntax:

HANDLEIN (single-integer-expression, single-text-expression)

Meaning:

HANDLEIN (card_to_override, data_object)

Returns:

A character text item

When the **HANDLEIN** function returns a character text item, the string is prefixed with: **IHx**, where **x** is the first parameter.

String format:

-IHx <internal-handle>.<offset>.<length>

Example:

-IH1 4.4096.512

A *single-integer-expression* is the number item that specifies the card number to override. A *single-text-expression* specifies the data object to use for the override.

In a RUN map rule, you can override a source card with a handle of a current map's card by specifying the card to override and the object to override it with.

Examples

In the following rule, the data is object c:b:a for input card 2 of the RUN map:

HANDLEIN(2, c:b:a)

The second parameter can be more complex if needed. For example,

HANDLEIN(2, SUBSTITUE(c:b:a, "a", "A"))

Any data in object **c:b:a** that contains a lowercase "a" would change it to an uppercase "A" for the call to the RUN map. In cases such as this, where the data cannot be expressed contiguously within the original card or cards, a scratch file is used to hold the expression and the calling map's handle is that of the scratch file.

There is no correlation between the handle returned in the output of **HANDLEIN** and the actual card number of the second parameter.

Related functions

Generally, the **ECHOIN** function is used to return a command or property to be used in a **RUN** function argument. However, for large amounts of data, using the **HANDLEIN** function can be more efficient.

For best results, use the **ECHOIN** function for small quantities of data, such as less than 100K, and the **HANDLEIN** function for larger quantities of data of more than 100K.

PARSE

The **PARSE** function analyzes a data object and validates it against the metadata of a component type. The resulting data structure is available to the calling map for further processing.

The input to the **PARSE** function is any data that can be processed as a text blob, for example, the result of a functional map invocation, a **GET** function, or a **RUN** map.

Syntax

PARSE (general_expression)

Meaning

PARSE (*data_to_parse*)
Returns
A mapped data object in a work file

Examples

The data structure that results from the **PARSE** function is available to the calling map. When you use a **RUN** map to parse data, the resulting data structure is available to the **RUN** map rather than to the calling map. By using the **PARSE** function instead of the **RUN** map implementation, you have fewer maps to configure and maintain.

The following examples:

1. Receive an input document from the SPE adapter
2. Extract the data objects from the document envelope
3. Return a response

Example 1: RUN map implementation

In this example, the DEENVELOPE map calls the **RUN** map named DUMPRESPONSE to extract the data from the document.

The DUMPRESPONSE map has two input cards:

- SpeResponse (Response global XSD)
- Test Root (text root)

The output card of the DUMPRESPONSE map has the following rule:

```
=WriteDocument(Document:sequence:Documents:sequence:SpeResponse,  
TestRoot;  
TEXT(INDEX(Document:sequence:Documents:sequence:SpeResponse))) + "<NL>"
```

The DEENVELOPE map has two input cards:

- DataIn (text root)
- Test Root (text root)

The DEENVELOPE map has two output cards:

- DOCOUTPUT:

```
= RUN ("DumpResponse", ECHOIN( 1 , Deenvelope ) + ECHOIN( 2 , TestRoot ) + " -OE1" )
```
- DEENVELOPE:

```
=VALID(GET("SPE", "-DURL jdbc:derby://localhost:1527/spe2  
-DBUSER derbyuser -DBPSWD derbypw  
-DBDRIVER org.apache.derby.jdbc.ClientDriver -DENV", DataIn),  
FAIL ("SPE Deenvelope failed: " + LASTERRORMSG() ))  
  
/*  
=VALID(GET("SPE", "-DURL jdbc:db2://bobdb9.bcr.ibm.com:50000/spetest  
-DBDRIVER com.ibm.db2.jcc.DB2Driver -DBUSER spetest -DBPSWD Seer5r12  
-DENV -T ", DataIn),  
FAIL ("SPE Deenvelope failed: " + LASTERRORMSG() ))  
*/  
  
/*  
=VALID(GET("SPE", "-DURL jdbc:db2://localhost:60000/SPETEST  
-DBDRIVER com.ibm.db2.jcc.DB2Driver -DBUSER fred -DBPSWD org13asd  
-DENV -T ", DataIn),  
FAIL ("SPE Deenvelope failed: " + LASTERRORMSG() ))  
*/
```

Example 2: PARSE function implementation

In this example, a single map uses the **PARSE** function to perform the processing of the two maps in Example 1.

The DEENVELOPE_PARSE map has two input cards:

- DataIn (text root)
- Test Root (text root)

The DEENVELOPE_PARSE map has two output cards:

- RESPONSEINFO:

```
= WriteDocument( Document:sequence:Documents:sequence:SpeResponse,  
TestRoot,  
TEXT( INDEX( Document:sequence:Documents:sequence:SpeResponse ) ) ) + "<NL>"  
  
= "Response - Status: " + Status:sequence:SpeResponse + ",  
Advanced status: " + AdvancedStatus:sequence:SpeResponse + "<NL>"
```
- SPERESPONSE (Response Global XSD):

```
=PARSE( VALID(GET("SPE", "-DURL jdbc:derby://localhost:1527/spe2  
-DBUSER derbyuser -DBPSWD derbypw  
-DBDRIVER org.apache.derby.jdbc.ClientDriver -DENV", DataIn),  
FAIL ("SPE Deenvelope failed: " + LASTERRORMSG() )) )
```

REFORMAT

The **REFORMAT** function returns a type object that results from replacing the syntax of the input type with the syntax of the output type.

The initiator and terminator of the output type are built. The content result rules are determined based on the Input type and the Output type. Groups and Items can be used with **REFORMAT**.

REFORMAT cannot be used as an argument to a function, operator, or functional map.

Syntax:

REFORMAT (single-object-expression)

Meaning:

REFORMAT (type-to-convert)

Returns:

A type object

REFORMAT returns a type object whose content matches the input type object, but whose syntax matches the output type object. The following table details the expected results based on the input and output.

Input Type	Output Type	Result
Group	Group	The content for each output component results from the content of the corresponding input group component content. For example, output component 1 is matched with input component 1, output component 2 is matched with input component 2, and so on. The components of the groups must be items, not another group. If there is no matching input component for an output component, the output component's required occurrences are built as "none". If there is no matching output component for an input component, the data of that input component is ignored. When a component occurrence is built, the delimiter (if any) of the contained output group is built and an object of the component's output type is built from the corresponding input type using the REFORMAT algorithm.
Group	Item	The text of a corresponding input group content.
Item	Group	The input item content is applied as though it were the first component of the output group.
Item	Item	The input item content is applied to the output item content.

Examples

- MyXMLPurchaseOrder = **REFORMAT**(MyCOBOL_PurchaseOrder)

This function is useful when the same type structure might come from different type trees, such as different versions of the same EDI, or when converting traditional formats to XML.

RESOLVETYPE

The **RESOLVETYPE** function resolves the provided component path to a specific type tree reference when the map is being compiled.

This function returns the type of the component for the component path that you provided as an argument. It is used only with the **IN** keyword, to resolve the provided component path to a specific type tree reference. Use it when more than one type that has the same name occurs within (**IN**) the containing component or card. Specify this function only in map rules, not in component rules.

Syntax:

RESOLVETYPE (single-text-expression)

Meaning:

RESOLVETYPE (component_path_to_resolve)

Returns:

A single type reference

For more information about using the **IN** keyword to reference all occurrences of an object contained in another object, see [Component paths separated by IN](#).

Example

```
NetWorth = SUM (RESOLVETYPE  
(Price:Cars:East:Regions) IN Inventory)
```

In the example, you have an inventory of cars and trucks in dealerships in different geographic regions: East, West, North, and South. The price of the cars and trucks are represented by two different types in the type tree with the same name, Price. You want to determine the net worth of the inventory of just cars from all of the dealerships in all of the regions.

The **RESOLVETYPE** function is used to return the type of the component path. In this example, the function is set up to return the type, Price, for Cars, for the component path, **Price:Cars:East:Regions**, specified in the argument of the function. This function also returns the same type, Price, for Cars, if the argument of the function specifies any of the Regions in the following component paths: **Price:Cars:West:Regions**, **Price:Cars:North:Regions**, and **Price:Cars:South:Regions**. Then Price, for Cars, is used as the criteria for the selection process in your total Inventory. The mapping process calculates the net worth by summing the values of Price for all Cars in all of the Regions in the total Inventory.

If you do not specify the **RESOLVETYPE** function, and instead, specify a rule such as **NetWorth = SUM (Price IN Inventory)**, the mapping process uses either one of the types, Price for Cars, or Price for Trucks, as the criteria for the selection process in your total Inventory. If you do specify the **RESOLVETYPE** function, the mapping process uses the returned type of the component path, specified in the argument of the function, as the criteria for the selection process in your total Inventory.

If you want to determine the net worth of just Trucks in the Inventory from all of the dealerships in all of the Regions, you can specify the **RESOLVETYPE** function in the following way: **NetWorth = SUM (RESOLVETYPE (Price:Trucks:North:Regions))**

IN Inventory). Again, the function also returns the same type, Price, for Trucks, if the argument of the function specifies any of the Regions in the following component paths: **Price:Trucks:West:Regions**, **Price:Trucks:East:Regions**, and **Price:Trucks:South:Regions**.

Bidirectional functions

Bidirectional languages such as Arabic and Hebrew are languages in which the text is presented to the user ordered from right to left, but numbers and Latin alphabetic strings within the text are presented left to right. In addition, the order in which characters appear within program variables can vary. The text is usually stored in logical order, the order in which the characters are entered in the input field. These differences in ordering and in other associated presentation characteristics require the program to have the ability to convert bidirectional text strings from one format to another.

Use the **SETLOGICALORDER**, **SETVISUALORDER**, **SETORIENTATIONLTR**, **SETORIENTATIONRTL**, **SETTEXTSHAPING**, and **SETTEXTSHAPINGOFF** functions for specific transformation needs when dealing with bidirectional data.

About the order of text

The order of text generally corresponds with way it is stored in memory or on disk. In general, the order of text is displayed, on the screen for example, in visual order so that a person can read it the way it is subjectively meant to be read. That is, left-to-right characters are displayed from left-to-right, and right-to-left characters are displayed with the first character of the text block beginning from the right.

Storage in memory or on disk generally occurs in a logical manner so that the left-to-right characters are stored left-to-right (that is, one after another), and require no processing to display properly. However, when right-to-left characters are stored in a logical manner, they are also stored one after another from left-to-right. These text blocks, when in Logical order, are usually enclosed by markers that identify the start and end positions of the right-to-left text blocks.

Logical to visual reordering

To display these logical sequences on a screen would require that any left-to-right text is simply read and displayed. When right-to-left text is encountered, it must be turned around to the appropriate direction. This is logical-to-visual reordering.

Examples

In the following example, lowercase characters are considered left-to-right (LTR) and uppercase characters are considered right-to-left (RTL).

Text=	a	b	c	F	E	D	g	h	i
Visual order	0x61	0x62	0x63	0x46	0x45	0x44	0x67	0x68	0x69
Logical order	0x61	0x62	0x63	0x44	0x45	0x46	0x67	0x68	0x69

- [SETLOGICALORDER](#)
- [SETORIENTATIONLTR](#)
- [SETORIENTATIONRTL](#)
- [SETTEXTSHAPING](#)
- [SETTEXTSHAPINGOFF](#)
- [SETVISUALORDER](#)

SETLOGICALORDER

Use this function to convert a bidirectional data object from visually ordered (characters ordered as they are presented for reading) to logically ordered (the order in which text is typed on a keyboard or phonetic order).

This function is only for use with text objects that have bidirectional properties. The function will have no affect on objects that have no bidirectional properties.

Syntax:

SETLOGICALORDER (single-text-expression , single-text-expression)

Meaning:

SETLOGICALORDER (visually_ordered_text_object , TRUE_or_FALSE_text_object)

Returns:

A logically ordered text object.

The **SETLOGICALORDER** function changes the order of the data as held in memory (order scheme) of a text object that is being passed as a parameter. The function returns a text object that reflects the order scheme change.

The output of the function will be the data from the original bidirectional item, transformed to match the specified layout (as it would be if it were assigned to a bidirectional item with the specified characteristics). This allows you to use text manipulation functions, such as LEFT or MID, on bidirectional data using the proper understanding of the bidirectional data. For example, LEFT of the logical ordering of a bidirectional item would actually take the rightmost characters of the visual ordering of that item.

Example

The properties of the text objects define the orientation and memory order of data objects found in the data stream.

Text objects defined as right-to-left would be considered as starting from the right character and progressing to the left, until the end of the data. However, the data can be held in memory with the start character (the right most character) being held in the left most position. This memory ordering is defined by the ordering scheme of the object.

SETORIENTATIONLTR

The SETORIENTATIONLTR function converts the orientation of a text object from right-to-left (RTL) to left-to-right (LTR).

Syntax:

SETORIENTATIONLTR (single-text-expression, Boolean-expression)

Meaning:

SETORIENTATIONLTR (right_to_left_oriented_text_object, true_or_false_text_object)

Returns:

A text object with left-to-right orientation.

The second parameter, a Boolean expression, indicates whether symmetric swapping should be applied.

Example

```
SETORIENTATIONRTL( VLTRData:In1, "FALSE"  
 ) )
```

SETORIENTATIONRTL

Use the SETORIENTATIONRTL function to convert the orientation of a text object from left-to-right (LTR) to right-to-left (RTL).

Syntax:

SETORIENTATIONRTL (left_to_right_oriented_text_object, true_false_text_object, TRUE_or_FALSE_text_object)

Meaning:

SETORIENTATIONRTL (XML fragment)

Returns:

A text object with right-to-left orientation.

Example

```
SETORIENTATIONRTL( VLTRData:In1, "FALSE")
```

SETTEXTSHAPING

The SETTEXTSHAPING function converts a bidirectional text object from non-shaped to shaped.

Syntax:

SETTEXTSHAPING (single-text-expression)

Meaning:

SETTEXTSHAPING (unshaped_text_object)

Returns:

A shaped text object.

SETTEXTSHAPINGOFF

The SETTEXTSHAPINGOFF function removes the shaping from a bidirectional text object.

Syntax:

SETTEXTSHAPINGOFF (single-text-expression)

Meaning:

SETTEXTSHAPINGOFF (shaped_text_object)

Returns:

A text object with no shaping.

SETVISUALORDER

The SETVISUALORDER function converts a bidirectional data object from logically ordered to visually ordered.

This function can only be used with bidirectional data.

Syntax:

SETORDERVISUAL (single-text-expression , single_text_expression)

Meaning:

SETORDERVISUAL (text_object , text_object)

Returns:

A visually ordered text object.

When IBM Transformation Extender processes bidirectional data, this function changes the order scheme of the text object that is passed as a parameter.

Example

```
SETVISUALORDER(SETORIENTATIONRTL(LSRTLData:In1,"FALSE"), "FALSE")
```

True or false indicates whether the text object has symmetric swapping enabled. A visual ordering scheme would have the data being read and stored with the start character in the rightmost position in both memory and visually.

Bit manipulation and testing functions

- [SETOFF](#)
- [SETON](#)
- [TESTOFF](#)
- [TESTON](#)

SETOFF

The **SETOFF** function sets to zero a specified bit in a binary number. You can use this function to manipulate (turn off) a bit in a binary number.

Syntax:

SETOFF (single-binary-number-expression, single-integer-expression)

Meaning:

SETOFF (*binary_number_to_change*, *bit_to_turn_off*)

Returns:

A single binary number

The **SETOFF** function uses *bit_to_turn_off* to specify the bit of *binary_number_to_change* to be set to the value 0. The result is a binary number item of the same size as *binary_number_to_change*.

The value of *bit_to_turn_off* represents the position of the single bit in *binary_number_to_change* to be set off. (Bits are numbered from left to right, with the leftmost bit being bit 1.) If *bit_to_turn_off* is less than one or greater than the number of bits of *binary_number_to_change*, **SETOFF** returns *binary_number_to_change* unchanged.

Examples

- **SETOFF (A , 16)**
Assume **A** is the two-byte binary value of "1" (which is all zeros except for bit **16**). The binary representation of the value in **A** is 0001.
The result is the two-byte binary value, "0"
- **SETOFF (A , 38)**
Results in the two-byte binary value "1", the original value of **A**, because bit **38** does not exist in **A**.

SETON

You can use the **SETON** function to "turn on" a specific bit in a binary number. **SETON** sets the specified bit in the number to "1".

Syntax:

SETON (single-binary-number-expression, single-integer-expression)

Meaning:

SETON (*binary_number_to_change*, *bit_to_turn_on*)

Returns:

A single binary number

SETON uses the value of *bit_to_turn_on* to specify the bit of *binary_number_to_change* that should be set to the value 1. The result is a binary number item of the same size as *binary_number_to_change*.

Bit_to_turn_on represents the position of the single bit in *binary_number_to_change* to be set on. (Bits are numbered from left to right, with the leftmost bit being bit 1.) If *bit_to_turn_on* is less than one or greater than the number of bits of *binary_number_to_change*, **SETON** returns *binary_number_to_change*, unchanged.

Examples

- **SETON (A , 15)**
In this example, assume **A** is the two-byte binary value of 1 (which is all zeros except for bit 16). The binary representation of the value in **A** is 0001.
The function returns the two-byte binary value, 0011, which is the decimal value of 3.
- **SETON (A , 40)**
Returns the two-byte binary value of 1-the original value of **A**- because bit 40 does not exist in **A**.

TESTOFF

The **TESTOFF** function tests a specified bit in a binary number item to see whether it is off.

Syntax:

TESTOFF (single-binary-number-expression , single-integer-expression)

Meaning:

TESTOFF (binary_number_to_test , bit_to_test)

Returns:

"True" or "false"

The value of *bit_to_test* specifies which bit of *binary_number_to_test* should be tested for the value 0. If *bit_to_test* has the value 1, it refers to the leftmost bit of *binary_number_to_test*.

The **TESTOFF** function returns "true" if the specified bit is off and returns "false" if the specified bit is on.

If *bit_to_test* is less than one or greater than the number of bits of *binary_number_to_test*, **TESTOFF** returns "false".

Examples

- **TESTOFF** (A, 16)

Assume **A** is the two-byte binary value of "1", which is all zeros except for bit **16**. The binary representation of the value in **A** is 0001.

This example returns "false".

- **TESTOFF** (A, 20)

Returns "false" because bit **20** does not exist.

TESTON

The **TESTON** function tests a specified bit in a binary number to see if it is on.

Syntax:

TESTON (single-binary-number-expression, single-integer-expression)

Meaning:

TESTON (binary_number_to_test ,bit_to_test)

Returns:

"True" or "false"

The value of *bit_to_test* specifies the bit of *binary_number_to_test* to test for the value 1. If *bit_to_test* has the value 1, it refers to the leftmost bit of *binary_number_to_test*.

The function returns "true" if the specified bit is on; it has the value 1. It returns "false" if the specified bit is off; it has the value 0.

If *bit_to_test* is less than one or greater than the number of bits of *binary_number_to_test*, **TESTON** returns "false".

Examples

- **TESTON** (A , 16)

Assume **A** is the two-byte binary value of "1", which is all zeros except for bit **16**. The binary representation of the value in **A** is 0001.

This example returns "true".

- **TESTON** (A , 20)

Returns "false" because bit **20** does not exist in a two-byte value.

Cache functions

Use cache functions to specify a value for each of the cache variables defined for a flow, and additionally, to alter an already specified value for a cache variable.

Cache variables act as an means to process information throughout a flow instance without requiring the use of data links.

Cache variables are name value pairs that persist until the flow instance completes execution. Cache variables are case-sensitive. Cache variables are also instance specific. Instance-specific means that no two flow-instances that are running simultaneously can share the same set of variables.

- **CACHEDELETE**

The **CACHEDELETE** function is same as **CACHEREAD**. This function returns a default value for cache variable if specified, or an empty value if the cache variable has not been defined.

- **CACHEREAD**

The **CACHEREAD** function reads the value associated to a cache variable.

- **CACHEWRITE**

The **CACHEWRITE** function sets a new or updates an existing cache variable. This function specifies the name of a cache variable if one does not already exist. Otherwise, it updates the value of an existing cache variable with the new value.

CACHEDELETE

The **CACHEDELETE** function is same as **CACHEREAD**. This function returns a default value for cache variable if specified, or an empty value if the cache variable has not been defined.

Syntax:

CACHEDELETE (single-text-expression [, single-text-expression])

Meaning:

CACHEDELETE (variable_name [, default_value])

Returns:

A single text item if found, otherwise none.

Examples

- `flowlib->CACHEDELETE ("Greeting")`
Deletes cache variable, Greeting, and returns empty (NONE).
 - `flowlib->CACHEDELETE ("Greeting", "Hi")`
Deletes cache variable, Greeting, and returns default value (Hi).
-

CACHEREAD

The **CACHEREAD** function reads the value associated to a cache variable.

This function returns a default value if specified, or an empty value if the cache variable has not been defined, or if it has been deleted prior to making this function invocation. Otherwise, it returns the associated value of the cache variable to the map in text format. The value associated to a internal cache variable can be retrieved with this function for further transformation purposes. Wildcards are not supported in this map rules because there is no good way to return multiple values.

The value associated to a cache variable is a text expression and returns the new value to the map. Empty value skips creating or updating of a cache variable.

Syntax:

CACHEREAD (single-text-expression [, single-text-expression])

Meaning:

CACHEREAD (variable_name [, default_value])

Returns:

A single text item.

Examples

- `flowlib->CACHEREAD ("Greeting")`
Returns the current value of custom cache variable, Greeting, to the map if it exists. Otherwise, returns the default value, empty value.
 - `flowlib->CACHEREAD ("Greeting", "Hi")`
Returns the current value of custom cache variable, Greeting, to the map if it exists. Otherwise, returns the default value, Hi, to the map.
-

CACHEWRITE

The **CACHEWRITE** function sets a new or updates an existing cache variable. This function specifies the name of a cache variable if one does not already exist. Otherwise, it updates the value of an existing cache variable with the new value.

The value associated to a cache variable is a text expression and returns the new value to the map. Empty value skips creating or updating of a cache variable.

Syntax:

CACHEWRITE (single-text-expression, single-text-expression)

Meaning:

CACHEWRITE (variable_name, variable_value)

Returns:

A single text item.

Examples

- `flowlib->CACHEWRITE ("Greeting", "Hello")`
Writes cache variable, Greeting, with Hello and returns Hello.
-

Conversion functions

- [BASE64TOTEXT](#)
- [BCDTOHEX](#)
- [BCDTOINT](#)
- [BCDTOTEXT](#)
- [CONVERT](#)
- [DATETONUMBER](#)
- [DATETOTEXT](#)
- [FROMBASESETEN](#)
- [FROMDATETIME](#)
- [FROMNUMBER](#)
- [HEXTXTTOSTREAM](#)
- [INT](#)

- [NUMBERTODATE](#)
 - [NUMBERTOTEXT](#)
 - [PACK](#)
 - [PACKAGE](#)
 - [QUOTEDTOTEXT](#)
 - [SERIESTOTEXT](#)
 - [STREAMTOHEXTEXT](#)
 - [SYMBOL](#)
 - [TEXTTOBASE64](#)
 - [TEXTTOBCD](#)
 - [TEXTTODATE](#)
 - [TEXTTONUMBER](#)
 - [TEXTTOQUOTED](#)
 - [TEXTTOTIME](#)
 - [TIMETOTEXT](#)
 - [TOBASETEN](#)
 - [TODATETIME](#)
 - [TONUMBER](#)
 - [UNPACK](#)
 - [UNZONE](#)
 - [ZONE](#)
-

BASE64TOTEXT

Use the BASE64TOTEXT decoding function to convert previously encoded data from BASE64 back to the original text. The data is converted to the character set of the output object used. For example, if the output object is EBCDIC the output will be EBCDIC.

Syntax:

BASE64TOTEXT (single-object-expression)

Meaning:

BASE64TOTEXT (BASE64_object_to_convert)

Returns:

A single-text-item

This function receives a text object in Base64-encoded format, converts the object to text, and returns a single data object that represents the original text object that was encoded. If an error occurs during conversion, no data is returned.

Example

```
BASE64TOTEXT("U29tZSBFeGFtcGxlIERhdGE=")
```

Output: Some Example Data

BCDTOHEX

The **BCDTOHEX** function converts an item from binary coded decimal (BCD) format to hexadecimal format. The binary value that is returned always has the high-order byte first, regardless of the operating system. This is useful for testing a specific bit position for a specific value.

Syntax:

BCDTOHEX (single-text-expression, single-integer-expression)

Meaning:

BCDTOHEX (BCD_item_to_convert, length_of_output)

Returns:

A single binary bytestream

When using the **BCDTOHEX** function, the input argument is converted from BCD format to its binary value.

Numbers in BCD format have two decimal digits in each byte. Each half-byte, therefore, can contain a binary value from 0000 (which represents the digit 0) through 1001 (which represents the digit 9). Based on this definition, the following behavior applies:

- If any half-byte of the BCD number contains the binary values 1101 or 1111, that half-byte is ignored.
- If any half-byte contains the binary values 1010, 1011, 1100, or 1110, the output of the function is "none".

The length of the output argument is specified as the value of the second input argument. The length must be 1, 2, or 4. If it is any other value, the length is assumed to be 2.

Examples

- You can use **BCDTOHEX** to test a specific bit position for a specific value. For example, **BCDTOHEX** is useful when the BCD value represents flag bits. The conversion performed by **BCDTOHEX** results in a predictable location for the flag bits across operating systems where the bytes might otherwise be reversed.
- **TESTON (BCDTOHEX (ProcessIndicator, 2) 10)**

Tests for the x`0040' bit in a two-byte result.

Note: This same test, using **BCDToInt** on a PC, tests the x`4000' bit.

Related functions

- **BCDTOTEXT**
 - **BCDToInt**
 - **TEXTTOBCD**
-

BCDToInt

The **BCDToInt** function converts an item from BCD (binary coded decimal) format to integer format. The binary value that is returned has a form native to the machine on which the map is being run.

Syntax:

BCDToInt (single-text-expression , single-integer-expression)

Meaning:

BCDToInt (BCD_item_to_convert , length_of_output)

Returns:

A single-binary-number

BCD_item_to_convert is converted from BCD (binary coded decimal) format to its integer value.

Numbers in BCD format have two decimal digits in each byte. Each half-byte, therefore, can contain a binary value from 0000 (which represents the digit 0) through 1001 (which represents the digit 9). Based on this definition, the following behavior applies:

- If any half-byte of the BCD number contains the binary values 1101 or 1111, that half-byte is ignored.
- If any half-byte contains the binary values 1010, 1011, 1100, or 1110, the output of the function is "none".

The length of the output binary number is specified by *length_of_output*. The length must be 1, 2, or 4. If it is any other value, the length is assumed to be 2.

BCDToInt returns a binary value of the form native to the machine on which the map is being run. The value can therefore be used in arithmetic operations or compared to other numeric values. It should *not* be used for bit manipulations, because the order of the bytes in the number is dependent on the operating system on which the map is being run. For example, on a personal computer, the low-order byte is first, while on a mainframe, the high-order byte is first.

Examples

- **BCDToInt** (bcdAmount Field , 2)
If the **bcdAmount** Field contains x'37' in BCD format, this example returns the integer value of 37.

Related functions

- **BCDTOTEXT**
 - **BCDToHEX**
 - **TEXTTOBCD**
-

BCDTOTEXT

The **BCDTOTEXT** function converts the digits in a BCD (Binary Coded Decimal) item to a text item containing the digits of the BCD-encoded item as a string of characters.

Syntax:

BCDTOTEXT (single-text-expression)

Meaning:

BCDTOTEXT (BCD_item_to_convert)

Returns:

A single text item

BCD_item_to_convert is converted from BCD format to a text string containing the digits of the BCD-encoded value as a string of characters.

Numbers in BCD format have two decimal digits in each byte. Each half-byte, therefore, can contain a binary value from 0000 (which represents the digit 0) through 1001, which represents the digit 9). Based on this definition, the following behavior applies:

- If any half-byte of the BCD number contains the binary value 1101 or 1111, that half-byte is ignored.
- If the BCD item contains the binary value 1010, 1011, 1100, or 1110, the output of the function is "none".

Examples

- **BCDTOTEXT** (Qty:Item)
If **Qty** is x`1234', the result is 1234.
- **BCDTOTEXT** (DiscountAmt)
If **DiscountAmt** is x`0123', the result is 0123.
- **BCDTOTEXT** (TotalDollars)

If **Total** is x' F123', the result is 123.

Related functions

- **BCDTOHEX**
 - **BCDTOINT**
 - **TEXTTOBCD**
-

CONVERT

The **CONVERT** function replaces each byte of a byte stream or text expression with a byte from another byte stream or text expression. The decimal value of each byte of the first argument is used to locate the corresponding byte in the second argument that will replace it.

Syntax:

CONVERT (single-byte-stream-or-text-expression , single-byte-stream-or-text-expression)

Meaning:

CONVERT (bytes_to_replace , replacement_bytes)

Returns:

A single byte stream or text item

The **CONVERT** function replaces each byte of *bytes_to_replace* with a byte from *replacement_bytes*. The byte chosen from *replacement_bytes* is the one whose index is the decimal value of *bytes_to_replace*. The first byte of *replacement_bytes* has the index value of zero. If there is no corresponding byte in *replacement_bytes*, **CONVERT** returns "none".

You can use the **CONVERT** function to convert ASCII to EBCDIC or EBCDIC to ASCII data. See the **convert.mms** map source file in the **examples\general\portdata** folder of the product installation directory. One of the maps in **convert.mms** is **ASCII_TO_EBCDIC**, which converts ASCII to EBCDIC. The **EBCDIC_TO_ASCII** map converts EBCDIC to ASCII.

ASCII to EBCDIC conversion can be performed automatically by defining the appropriate data language (ASCII or EBCDIC) for each data item.

Examples

- **CONVERT (SYMBOL (0),"AB")**
Returns A
 - **CONVERT (SYMBOL (1),"AB")**
Returns B
 - **CONVERT (SYMBOL (2),"AB")**
Returns "none"
 - **CONVERT (ASCII , ATOETable)**
Converts ASCII text to EBCDIC based on values in a table named **ATOETable**
-

DATETONUMBER

Use **DATETONUMBER** to perform arithmetic on dates.

The **DATETONUMBER** function returns an integer that results from counting the number of days from December 31, 1864, to the specified date.

Syntax:

DATETONUMBER (single-date-expression)

Meaning:

DATETONUMBER (date_to_convert)

Returns:

A single integer

The **DATETONUMBER** function converts a date to an integer. The resulting integer represents the number of days since December 31, 1864, where using **DATETONUMBER** with a date of January 1, 1865 returns the integer value 1.

If the input argument is in error, the function returns the value "none".

If the date format specified by the input argument does not include century, the century is determined based on the **Century** map setting using the **CCLookup** parameter or the current century.

Examples

- DaysBetween = **DATETONUMBER** (StopDate) - **DATETONUMBER** (StartDate)
This expression could be used in a map rule to produce a value for **DaysBetween**. It converts the **StopDate** and the **StartDate** to integers, then subtracts the resulting two integers, and returns the result as **DaysBetween**.
-

Related functions

- **ADDDAYS**
- **NUMBERTODATE**

DATETOTEXT

The **DATETOTEXT** function converts a date object or expression to a text item.

Syntax:

DATETOTEXT (single-date-expression)

Meaning:

DATETOTEXT (date_to_convert)

Returns:

A single text item

If *date_to_convert* is a date object name, this returns the date as a text item formatted according to the presentation of the date object.

If *date_to_convert* is a date expression produced by a function, this returns the date as a text item formatted according to the presentation of the output argument of that function.

Examples

- **DATETOTEXT** (ShipDate)

In this example, **ShipDate** is converted from a date to text. If **ShipDate** has a CCYYMMDD presentation, the resulting text item will have that presentation, as well.

- **DATETOTEXT** (**CURRENTDATETIME** ("{MM/DD/CCYY}"))

In this example, **CURRENTDATETIME** evaluates and returns a date in MM/DD/CCYY format. Then **DATETOTEXT** evaluates and returns a text string that is that date in MM/DD/CCYY format.

For example, use **DATETOTEXT**, to do text concatenation. The **FROMDATETIME** function provides greater flexibility in specifying the format of the resulting text item.

Related Functions

- **FROMDATETIME**
- **NUMBERTOTEXT**
- **TEXT**
- **TEXTTODATE**
- **TEXTTONUMBER**
- **TEXTTOTIME**
- **TIMETOTEXT**
- **TODATETIME**

FROMBASESETEN

You can use **FROMBASESETEN** when you need to convert numbers to a base other than 10.

The **FROMBASESETEN** function converts an integer to a text item that can be interpreted as a number, using positional notation of the base specified.

Syntax:

FROMBASESETEN (single-integer-expression, single-integer-expression)

Meaning:

FROMBASESETEN (positive_integer_to_convert, base_to_convert_to)

Returns:

A single text item

FROMBASESETEN returns a text item that results from converting *positive_integer_to_convert* to a text item that can be interpreted as a number using positional notation of the base specified by *base_to_convert_to*.

If *base_to_convert_to* is less than 2 or greater than 36, **FROMBASESETEN** evaluates to "none". Resulting text item characters A-Z are interpreted as digits having decimal values from 10-35, respectively. The characters returned are uppercase.

Example

- **FROMBASESETEN** (18, 2)
Returns the value 10010
- **FROMBASESETEN** (123, 8)
Returns the value 173

Related function

- **TOBASESETEN**

FROMDATETIME

The **FROMDATETIME** function converts a date/time item to a text string of a specified format. For example, you can use **FROMDATETIME** to convert a date-time item into a string for parsing or concatenation. You can also use this function to access the individual parts of a date or time, such as the month, day, year, and so forth.

Syntax:

```
FROMDATETIME (single-character-date-time-item  
[ , single-text-expression ])
```

Meaning:

```
FROMDATETIME (date_time [ , date_time_format_string ])
```

Returns:

A single character text item

FROMDATETIME returns the date/time specified by *date_time* as a text item with the format specified by *date_time_format_string*. If *date_time_format_string* is not specified, *date_time* will be returned in the same format as the *date_time*.

The *date_time_format_string* must conform to the date/time format strings as described in the "Format strings" topic.

Examples

- HeaderLine="Today is" + **FROMDATETIME (CURRENTDATE (), "{MON DD, CCYY}")**
If the current system date is March 3, 1999, this rule evaluates to Today is Mar 03, 1999.
- FROMDATETIME (TransactionTimeStamp Column:::TransHistory)**
If the value of **TransactionTimeStamp Column** is 10:14 A.M. on February 3, 2000 and its format is defined as "{CCYYMMDDHH24MM}", this rule evaluates to 200002031014.
- EXTRACT (Expense:Report, FROMDATETIME (Date:Expense:Report, "{MM}") = "03")**
In this example, the **FROMDATETIME** function is used in the condition expression of the **EXTRACT** function to identify all expenses in the month of March.

Related functions

- CURRENTDATE**
- CURRENTDATETIME**
- CURRENTTIME**
- DATETOTEXT**
- TODATETIME**

FROMNUMBER

The **FROMNUMBER** function converts a number to a text string of a specified format.

Use **FROMNUMBER** when you need to convert a number item into a string for parsing or concatenation.

Syntax:

```
FROMNUMBER (single-character-number-item  
[ , single-text-expression ])
```

Meaning:

```
FROMNUMBER (number_to_convert [ , number_format_string ])
```

Returns:

A single character text item

FROMNUMBER returns the number specified by *number_to_convert* as a text item with the format specified by *number_format_string*. If *number_format_string* is not specified, *number_to_convert* will be returned in the same format as the *number_to_convert*.

The *number_format_string* must conform to the number format strings as described in the "Format strings" topic.

Examples

- Greeting = "You are caller number " + **FROMNUMBER (SeqNo:::History, "#'###'!" + "!"**
If the value of **SeqNo** is 2348192, this rule evaluates to "You are caller number 2,348,192!"
- "\$" + **FROMNUMBER(CurrentRate Field:::Schedule, "#'###.'2##2")**
If the value of **CurrentRate Field** is 15.875, "\$15.88".

Related functions

- DATETONUMBER**
- NUMBERTODATE**
- NUMBERTOTEXT**
- TONUMBER**

HEXTXTTOSTREAM

HEXTXTTOSTREAM is the reverse of **STREAMTOHEXTXT**. You can use the **HEXTXTTOSTREAM** function to assign a binary text value to a character text item represented by hexadecimal pairs.

HEXTEXTTOSTREAM returns a binary text stream whose value is the evaluation of input character text represented by hexadecimal pairs.

Syntax:

HEXTEXTTOSTREAM (single-text-expression)

Meaning:

HEXTEXTTOSTREAM (series_of_hex_pairs)

Returns:

A single byte stream item

This function returns a binary text stream item whose value is the evaluation of input character text in *series_of_hex_pairs*, ignoring <WSP> characters between the hexadecimal pairs. White space characters include space, horizontal tab, carriage return, and line feed characters.

Input formats

The following table shows an example of input in its character text representation as viewed through the character editor, and in its ASCII code representation (binary text stream) as viewed through the hex editor. Each pair of binary text in the hex view represents one character in the character view of the character text.

Input ("41 42 43 44")	Editor View	Value
Character text (hex pairs)	Character	"41 42 43 44"
ASCII code representation (binary text stream)	Hex	0x3431203432203433203434

Examples

- **HEXTEXTTOSTREAM** ("41 42 43 44")
Returns the evaluated value of the input (ASCII) character text string "41 42 43 44" as the output (ASCII) character text string "ABCD" as viewed in the character editor. (The hex view of the input is 0x3431203432203433203434. The hex view of the output is 0x41424344.)
- **HEXTEXTTOSTREAM** ("0D 0A 00")
Returns the evaluated value of the input (ASCII) character text string "0D 0A 00" as the output (ASCII) character text string "<CR><LF><NULL>" as viewed in the character editor. (The hex view of the input is 0x3044203041203030. The hex view of the output is 0xD0A00.)

See Design Studio Introduction documentation for a list of special symbols.

Related functions

- **SYMBOL**
- **STREAMTOHEXTEXT**

INT

You can use the **INT** function when you need only the integer portion of a number.

Syntax:

INT (single-number-expression)

Meaning:

INT (number_to_convert)

Returns:

A single integer

INT returns the integer portion of a number. The result is the integer part of *number_to_convert*. Any fractional part after the decimal point is dropped.

Examples

- **INT** (1.45)
Returns 1
- **INT** (3.6)
Returns 3
- **INT** (Purchase:Amt - Discount:Amt)
Subtracts **Discount:Amt** from **Purchase:Amt** and returns the result as a whole number.

Related functions

- **MOD**
- **ROUND**
- **TRUNCATE**

NUMBERTODATE

You can use the **NUMBERTODATE** function to perform a calculation on a date.

NUMBERTODATE converts an integer to a date, where the integer is the number of days from December 31, 1864, to the specified date. Only positive, non-zero values can be specified as valid parameters.

Syntax:

NUMBERTODATE (single-integer-expression)

Meaning:

NUMBERTODATE (integer_to_convert)

Returns:

A single date

After converting an integer to a date, if the result is being assigned to a date object, the resulting date is in the presentation for that output. If the resulting date is not being assigned to an object, it has a CCYYMMDD presentation.

When the year exceeds four digits, the output will display hash characters (#) for the CCYY values because the field is defined to be only four digits in length.

Examples

- **NUMBERTODATE** (StartDate)

This example converts the **StartDate** value from an integer to a date that is in CCYYMMDD format.

Related functions

- **ADDDAYS**
 - **DATETONUMBER**
 - **FROMDATETIME**
 - **TODATETIME**
-

NUMBERTOTEXT

The **NUMBERTOTEXT** function converts a character number to a text item that looks like the original object.

You can use **NUMBERTOTEXT** when you need an object that is defined as a number converted to an object defined as text. This is useful when you need to concatenate text, however, the **FROMNUMBER** function provides greater flexibility in specifying the format of the resulting text item.

Syntax:

NUMBERTOTEXT (single-number-expression)

Meaning:

NUMBERTOTEXT (number_to_convert)

Returns:

A single text item

The resulting text looks like the input argument. The result is truncated, if necessary.

Examples

- **NUMBERTOTEXT** (ROUND (1000 - 24.75, 3))

This example converts the result of the calculation (rounded to 3 decimal places) to text, resulting in 975.250.

- **NUMBERTOTEXT** (PurchaseNumber)

This example converts **PurchaseNumber** from a number to text.

Related functions

- **FROMNUMBER**
 - **TEXTTONUMBER**
 - **TODATETIME**
 - **TONUMBER**
-

PACK

The **PACK** function converts an integer to a text item that can be interpreted as a packed decimal number.

The sign values for packed data are as follows:

- C for positive (+)
- D for negative (-)
- F for unsigned, which is read as positive

Syntax:

PACK (single-integer-expression)

Meaning:

PACK (integer_to_convert)

Returns:
A single text item

In a packed decimal number, each half-byte is a digit, except for the last half-byte of the rightmost byte, which contains a sign.

Examples

- **PACK (314)**
Returns "1L" and results in the hex value 31 4C (which, in ASCII, looks like 1L)
- **PACK ((Unit Price * Quantity) * 100)**
In this example, the packed number has two implied decimal places. Because **PACK** does not accept decimal places, including implied ones, the nested arithmetic expression, **Unit Price * Quantity**, is multiplied by 100 before rounding.
Define items as having a packed decimal number presentation. Then, when mapping to or from these items, the conversion to and from packed decimal is automatically performed as needed.

Related function

- **UNPACK**
-

PACKAGE

The **PACKAGE** function converts a group or item object to a text item, including its initiator, terminator, and any delimiters it contains.

Syntax:
PACKAGE (single-object-expression)

Meaning:
PACKAGE (object_to_convert)

Returns:
A single text item

The **PACKAGE** function converts *object_to_convert*, which must be a type reference to a text item, including the type reference's initiator, terminator, and all delimiters. **PACKAGE** differs from **TEXT** in that it includes the initiator and terminator of the specified type reference.

Examples

- **PACKAGE (Record:Card)**
Returns: #1339X10A,491.38,Green,42x54@

For this example, the group **Record** has an initiator of "#", a terminator of "@" and a delimiter of ",". The data looks like this: "#1339X10A,491.38,Green,42x54@".

Related functions

- **DATETOTEXT**
- **NUMBERTOTEXT**
- **SERIESTOTEXT**
- **TIMETOTEXT**
- **TEXT**

PACKAGE differs from **TEXT** because it includes the initiator and terminator of the input object.

JSON native schema package

JSON schema fields with _V following the name and fields starting with oneOf, allOf and anyOf are the virtual fields.

Virtual JSON fields do not appear in the data and have no syntax. Syntax is controlled by real fields above these virtual fields.

Real fields are fields that appear in the JSON data.

Using a virtual field in a package command will give incomplete syntax. Use a real field either above or below the virtual fields for complete syntax.

QUOTEDTOTEXT

The **QUOTEDTOTEXT** decoding function converts a single (text or binary) data object in Quoted-Printable format to a single data object that represents the original text or binary object.

Syntax:
QUOTEDTOTEXT (single-object-expression)

Meaning:
QUOTEDTOTEXT (quoted_printable_object_to_convert)

Returns:
A single-text-item

You can use this function to decode data that was previously encoded to RFC 1767 Quoted-Printable encoding. Any hexadecimal representations are decoded to the appropriate ASCII character and any soft breaks are removed.

If an error occurs during conversion, no data is returned.

Example

```
QUOTEDTOTEXT("Unencoded data=0CEncoded")
```

Output: Unencoded data<FF>Encoded

SERIESTOTEXT

You can use the **SERIESTOTEXT** function to project your input data as a series and to interpret it as a text item for output.

SERIESTOTEXT converts a contiguous or non-contiguous series to a text item.

Syntax:

```
SERIESTOTEXT (series-object-expression)
```

Meaning:

```
SERIESTOTEXT (series_to_convert)
```

Returns:

A single text item

SERIESTOTEXT returns a text item containing the concatenation of the series of the input argument, including nested delimiters but excluding initiators and terminators.

Examples

In this example, you have the following data that represents bowler information for a bowling league:

Andrews, Jessica:980206:JBC:145:138:177:159

Little, Randy:980116:BBK:175:168

Wayne, Richard:980102:JBC:185:204:179:164:212

Each record consists of the bowler's name, the date of their last game played, a team code and one or more bowling scores. **Record** is defined as a group that is infix-delimited by a colon.

Using the rule:

```
= SERIESTOTEXT (Score Field:Bowler:Input)
```

the following results are produced, which is the concatenation of all of the scores for all of the bowlers, even though the scores are not all contiguous within the data:

145138177159175168185204179164212

However, if the rule was changed, for instance, to concatenate the list of scores to the bowler's name:

```
= BowlerName Field:Bowler:Input + " ->" +  
SERIESTOTEXT (Score Field:Bowler:Input)
```

the following output would be produced:

Andrews, Jessica -> 145138177159

Little, Randy -> 175168

Wayne, Richard -> 185204179164212

In this example, you have an input number that is of variable size, followed by a name. There is no syntax that separates the number from the name. You can define the number as a group with **Byte(s)** as a component and provide a component rule for **Byte(s)**, such as:

```
ISNUMBER ($)
```

Related functions

- PACKAGE
 - TEXT
-

STREAMTOHEXTEXT

Use **STREAMTOHEXTEXT** to assign a character text value to a binary text stream item.

The **STREAMTOHEXTEXT** function returns a character text string represented by hex pairs whose value is the evaluation of an input binary text stream.

This function is the reverse of the **HEXTXTTOSTREAM** function.

Syntax:
STREAMTOHEXTEXT (single-byte-stream-item)

Meaning:
STREAMTOHEXTEXT (single-byte-stream-item)

Returns:
A series of hex pairs

STREAMTOHEXTEXT returns a string of hex pairs whose value is the evaluation of input binary text in *series_of_hex_pairs*.

Input formats

The following table shows an example of input in its ASCII code representation (binary text stream) as viewed through the hex editor, and in its character text representation as viewed through the character editor. Each four-character grouping of binary text in the hex view represents one character in the character view.

Input (0x41424344)	Editor View	Value
ASCII code representation (binary text stream)	Hex	0x41424344
	Character	"ABCD"

Examples

- **STREAMTOHEXTEXT** (0x41424344)

Returns the evaluated value of the input (ASCII) binary text stream **0x41424344** as the output (ASCII) binary text stream 0x3431343234333434 as viewed in the hex editor. (The character view of the input is "ABCD". The character view of the output is "41424344".)

- **STREAMTOHEXTEXT** (0x0D0A00)

Returns the evaluated value of the input binary text stream **0x0D0A00** as the output 0x304430413030 as viewed in the hex editor. (The character view of the input is "<CR><LF><NULL>". The character view of the output is "0D0A00".)

Related functions

- **HEXTXTTOSTREAM**
- **SYMBOL**

SYMBOL

You can use **SYMBOL** to include a symbol in your output.

The **SYMBOL** function returns a one-byte character that is the ASCII character equivalent of a specified decimal value.

Syntax:
SYMBOL (single-integer-expression)

Meaning:
SYMBOL (decimal_value)

Returns:
A single one-byte text item

The value of the resulting item is the ASCII character equivalent of *decimal_value*. Valid input values are 0 to 255. Values outside of this range return "none".

A listing of decimal values (0-127) and their ASCII character equivalents is included in the Design Studio Introduction documentation.

Examples

- **SYMBOL** (13)
Produces a carriage return
- **SYMBOL** (13) + **SYMBOL** (10)
Produces a carriage return/linefeed

You can accomplish the same result using the angle-brackets around the decimal value. For example, the second example above would be equivalent to <CR><LF> or <>0D><>0A>>

Related functions

- **HEXTXTTOSTREAM**
- **STREAMTOHEXTEXT**

TEXTTOBASE64

The TEXTTOBASE64 encoding function converts a text item to Base64 format.

Syntax:
TEXTTOBASE64 (single-object-expression)
Meaning:

TEXTTOBASE64 (text_object_to_convert)
Returns:
A single-text-item

The **TEXTTOBASE64** function receives a text object and uses the MIME implementation to convert the text to Base64 format. The MIME implementation adds the <CR><LF> character sequence after every 76 bytes and also at the end of the output string.

TEXTTOBASE64 returns a single data object that represents the Base64 encoding of the original text object. If an error occurs during conversion, no data is returned.

Example

```
TEXTTOBASE64("Some Example Data")
Output: UwBvAG0AZQAgAEUAeABhAG0AcABsAGUAIABEAGEAdABhAA==
```

TEXTTOBCD

The **TEXTTOBCD** function converts a text item from decimal digits to BCD (Binary Coded Decimal) format.

Syntax:
TEXTTOBCD (single-integer-text-expression)
Meaning:
TEXTTOBCD (text_to_be_converted)
Returns:
A single BCD-formatted text item

TEXTTOBCD converts *text_to_be_converted* (which consists of decimal digits) to BCD format. In this format, each byte contains two decimal digits represented as binary numbers. If there is an odd number of decimal digits in the input, the high-order half-byte of the leftmost output byte will contain the decimal value 15 (hex "F").

If anything other than a decimal digit is encountered in the input, **TEXTTOBCD** returns "none".

Examples

- **TEXTTOBCD ("1234")**
Returns the hexadecimal value x'1234'
- **TEXTTOBCD ("123A")**
Returns "none"
- **TEXTTOBCD ("123")**
Returns the hexadecimal value x'F123'

In this example, the values shown as input ("123") are meant to represent character items in the native character set to the machine on which the map is running. On a personal computer, "123" would contain the ASCII characters for the digits that have the hexadecimal values "31", "32", and "33". The output, described as "the hexadecimal value 'F123'", consists of the two binary bytes "F1" and "23".

On an IBM mainframe the input string would contain EBCDIC characters for the digits that have the hexadecimal values "F1", "F2", "F3", etc., but the output would be the same as the personal computer output.

Related functions

- **BCDTOHEX**
- **BCDToint**
- **BCDTOTEXT**

TEXTTODATE

The **TEXTTODATE** function converts text item from CCYYMMDD or YYMMDD format to a date.

You can use the **TEXTTODATE** to convert a text item to a date, however, the **TODATETIME** function provides greater flexibility for specifying the resulting date-time format.

Syntax:
TEXTTODATE (single-text-expression)
Meaning:
TEXTTODATE (text_to_convert_to_date)
Returns:
A single date

The *text_to_convert_to_date* must be in either CCYYMMDD or YYMMDD format. If the *text_to_convert_to_date* is in error (for example, it is not a valid date), the result is "none".

If *text_to_convert_to_date* is in YYMMDD format and is being assigned to a date format that includes a century, the century is determined based on the century run option setting using the **CCLookup** parameter or the current century.

Examples

- **TEXTTODATE ("990114")**
Returns a single date item with the value of January 14 in the year 99
- **TEXTTODATE (OrderDate)**
Returns **OrderDate** as a single date item.

Related functions

• DATETOTEXT	• TEXTTOTIME
• NUMBERTOTEXT	• TIMETOTEXT
• TEXTTONUMBER	• TODATETIME

TEXTTONUMBER

Use **TEXTTONUMBER** to convert text to a number.

The **TONUMBER** function provides greater flexibility for specifying the format of the text item that is to be converted to a number.

Syntax:

TEXTTONUMBER (single-text-expression)

Meaning:

TEXTTONUMBER (text_to_convert_to_number)

Returns:

A single character number

The *text_to_convert_to_number* must be in integer or ANSI-formatted (floating point) presentation. The resulting number looks like the input argument, however, nonsignificant zeroes to the right of the decimal separator will be truncated. If the input argument is in error (for example, it is not a recognizable as a valid number), the result is "none".

When specified as in ANSI-formatted presentation, the text string must meet the following requirements:

- The decimal point can be a period, a comma, or "none".
- The leading sign can be a plus sign, a minus sign, or "none".
- No thousands separator is allowed.

Examples

- **TEXTTONUMBER** (OrderQty)
Returns **OrderQty** as a character number item

Related functions

- **DATETOTEXT**
- **FROMNUMBER**
- **NUMBERTOTEXT**
- **TEXTTODATE**
- **TEXTTOTIME**
- **TIMETOTEXT**

TEXTTOQUOTED

The TEXTTOQUOTED encoding function converts a text or binary item to Quoted-Printable format.

Syntax:

TEXTTOQUOTED (single_object_expression)

Meaning:

TEXTTOQUOTED (object_to_convert)

Returns:

A single text or binary item

You can use this function to encode data that largely consists of octets that correspond to printable characters in the US-ASCII character set. It encodes the data in such a way that the resulting octets are unlikely to be modified by mail transport.

Quoted-Printable lines cannot be longer than 76 characters. Data with lines greater than 76 characters are broken up and indicated with soft breaks of "<CR><LF>".

If an error occurs during conversion, no data is returned.

Example

This example converts the <FF> (form feed) character, into a hexadecimal representation, and indicates the end of data with a soft break.

```
TEXTTOQUOTED("Unencoded data<FF>Encoded")
```

Output: Unencoded data=0CEncoded=

The data is converted using the Quoted-Printable encoding as per RFC 1767.

TEXTTOTIME

Use **TEXTTOTIME** when you want to convert an object defined as text that is in HHMM or HHMMSS presentation, to an item defined as time. For greater flexibility, use the **TODATETIME** function for specifying the format of the text item that is to be converted to a date/time.

Syntax:

```
TEXTTOTIME (single-text-expression)
```

Meaning:

```
TEXTTOTIME (text_to_convert_to_time)
```

Returns:

A single time

The *text_to_convert_to_time* must be in HHMM or HHMMSS presentation. HH is a two-digit hour in a 24-hour format. If the result is being assigned to a time object, the resulting time looks like the output object. Otherwise, the resulting time looks like the input argument. If the input argument is in error (for example, it is not a valid time), the result is "none".

Examples

- **TEXTTOTIME (CallTime)**

Returns **CallTime** as a time item

Related functions

• DATETOTEXT	• TEXTTONUMBER
• FROMDATETIME	• TIMETOTEXT
• NUMBERTOTEXT	• TODATETIME
• TEXTTODATE	

TIMETOTEXT

You can use the **TIMETOTEXT** function to perform text concatenation. For greater flexibility, use the **FROMDATETIME** function for specifying the format of the resulting text item.

TIMETOTEX converts a time object or expression to a text item.

Syntax:

```
TIMETOTEXT (single-time-expression)
```

Meaning:

```
TIMETOTEXT (time_to_convert_to_text)
```

Returns:

A single text item

If *time_to_convert_to_text* is a time object name, this returns the time as a text item formatted according to the presentation of the input date object.

If *time_to_convert_to_text* is a time expression produced by a function, this returns the time as a text item formatted according to the presentation of the output argument of that function.

Examples

- **TIMETOTEXT (LeadTime)**

In this example, **LeadTime** is converted from a time to text. If **LeadTime** has an HH:MM presentation, the resulting text item will be of that presentation.

- **TIMETOTEXT (CURRENTDATETIME ("{HH:MM:SS}"))**

Here, **CURRENTDATETIME** evaluates and returns a time in HH:MM:SS format. Then, **TIMETOTEXT** evaluates and returns a text string that is that time in HH:MM:SS format.

Related functions

• DATETOTEXT
• FROMDATETIME
• NUMBERTOTEXT
• TEXTTODATE

- TEXTT tonumber
 - TEXTT otim
 - TODATETIME
-

TOBASETEN

The **TOBASETEN** function converts a text item that can be interpreted as a number, using positional notation of the base specified, to a base 10 number.

Syntax:

TOBASETEN (single-text-expression, single-integer-expression)

Meaning:

TOBASETEN (text_to_convert, base_to_convert_from)

Returns:

A single integer

TOBASETEN returns a number that results from converting *text_to_convert* that can be interpreted as a number, using positional notation of the base specified by *base_to_convert_from*, to its base 10 representation. Text item characters A-Z are interpreted as digits having decimal values from 10-35, respectively.

If *base_to_convert_from* is less than 2 or greater than 36, **TOBASETEN** evaluates to "none". If *text_to_convert* contains a character that is not alphanumeric or is not in the range specified by *base_to_convert_from*, **TOBASETEN** returns "none".

Examples

- **TOBASETEN** ("A", 16)
Returns the value 10
- **TOBASETEN** ("10", 36)
Returns the value 36
- **TOBASETEN** ("A0", 15)
Returns the value 150
- **TOBASETEN** ("A0", 5)
Returns the value "none"

Related functions

- **FROMBASETEN**
-

TODATETIME

The **TODATETIME** function converts a text string of a specified format to a date-time item.

Syntax:

TODATETIME (single-character-text-expression
[, single-text-expression])

Meaning:

TODATETIME (text_to_convert [, date_time_format_string])

Returns:

A single character date item

TODATETIME returns the date-time that corresponds to the value specified by *text_to_convert*, which is in the format specified by *date_time_format_string*. If *date_time_format_string* is not specified, it will be assumed that *text_to_convert* is in [CCYYMMDDHH24MMSS] format.

The *date_time_format_string* must conform to the date-time format strings as described in "Format strings".

Examples

- **TODATETIME** ("05/14/1999@10:14pm", "[MM/DD/CCYY]@[HH12:MMAM/PM]")
In this example, a text string containing a date and time is converted to a date-time item.
- RptDate = **TODATETIME** (**RIGHT** (**GETRESOURCENAME** () , 8) , "CCYYMMDD")
Assume that you receive a file that contains historical data. The name of the file identifies the date of the historical data. For example, a filename of **19960424** indicates that the data was produced on April 24, 1996. To map this date to **RptDate**, the **TODATETIME** function could be used with the **RIGHT** and **GETRESOURCENAME** functions.

Related functions

- **CURRENTDATE**
- **CURRENTDATETIME**
- **CURRENTTIME**
- **TEXTTODATE**
- **TEXTTOTIME**

TONUMBER

The **TONUMBER** function converts a text string of a specified format to a number.

Syntax:

```
TONUMBER ( single-character-text-expression  
[ , single-text-expression ] )
```

Meaning:

```
TONUMBER ( text_to_convert [ , number_format_string ] )
```

Returns:

A single character number item

TONUMBER returns the number that corresponds to the value specified by *text_to_convert*, which is in the format specified by *number_format_string*. If *number_format_string* is not specified, it will be assumed that *text_to_convert* is in ANSI decimal format (for example, "{L-####[.'##]}").

The *number_format_string* must conform to the number format strings as described in the "Format strings" topic.

Examples

- **TONUMBER(text_to_convert, "{L+'\$'#[.###}")**
L+\$ indicates the leading dollar sign is positive. That leading sign and the comma separators are removed when the text is converted to a number.

Input String: \$123,000,000

Output: 123000000

- **TONUMBER(text_to_convert, "####[T-]")**
Four number signs are required for each whole number, regardless of the actual number of digits in the number.

Input string:	Output:	Note:
12345-	-12345	The output becomes a negative number.
67890	67890	No change occurs.
345-	-345	The output becomes a negative number.

- **TONUMBER(text_to_convert, "####[T+'K'-]")**

If an invalid character, such as an X, is encountered, nothing is returned.

If a **K** is encountered, it is treated as a positive indicator.

Input string:	Output:	Note:
11212-	-11212	The output becomes a negative number.
67890X		The X is an invalid character. No number is returned.
54354	54354	No change occurs.
34567K	34567	The K is recognized as a positive sign. The character is removed and the number is returned as a positive.
345-	-345	The output becomes a negative number.

- **TONUMBER(text_to_convert, "L-'('#[.###[T-]')")**

The parentheses indicating a negative number are removed and replaced with a negative sign.

Comma separators are removed when the text is converted to a number.

Input string:	Output:	Note:
(12,345)	-12345	The output becomes a negative number. The comma separator is removed.
67,890	67890	The comma separator is removed.
(345)	-345	The output becomes a negative number.

- **TONUMBER(text_to_convert, "#['#[.###[T+'K'-]')")**

The optional comma separators are removed, but the decimal points and decimal values are retained.

Input string:	Output:	Note:
54,345.098	54354.098	The comma separator is removed.
67890.0X		The X is an invalid character. No number is returned.
11213-	-11213	The output becomes a negative number.
34567K	34567	The K is recognized as a positive sign. The character is removed and the number is returned as a positive.
345.1-	-345.1	The output becomes a negative number.

Related functions

- **DATETONUMBER**
- **FROMNUMBER**
- **NUMBERTODATE**
- **NUMBERTOTEXT**

UNPACK

You can use the **UNPACK** function to do arithmetic with a packed decimal number or to move a packed decimal value into a numeric item.

UNPACK converts text that can be interpreted as a packed decimal number to a signed integer item.

The sign values for packed data are as follows:

- C for positive (+)
- D for negative (-)
- F for unsigned, which is read as positive

Syntax:

UNPACK (single-fixed-size-text-expression)

Meaning:

UNPACK (*text_to_unpack*)

Returns:

A single signed integer

UNPACK returns a signed integer representing the value *text_to_unpack*, which is a packed decimal number. If the *text_to_unpack* cannot be interpreted as a valid packed decimal, **UNPACK** evaluates to "none".

In a packed decimal number, each half-byte is a digit, except for the last half-byte of the rightmost byte, which contains a sign.

Examples

- **UNPACK** ("1L") returns 314

The ASCII string "1L" in hex is 31 4C, which, when interpreted as a packed number, results in (positive) 314. This example returns the value "+314".

The hexadecimal representation of the value "1L" is x`14C', where C in the rightmost half-byte represents a positive sign.

- **UNPACK** (*UnitPrice*) / 100 * *QuantityOrdered*

UnitPrice is unpacked and divided by 100 (to convert it from an integer to a number with two decimal places) and then multiplied by the *QuantityOrdered*.

You can define items as having a packed decimal number presentation. Then, when mapping to or from these items, the conversion to and from packed decimal is automatically performed as needed.

Related functions

- **PACK**

UNZONE

You can use the **UNZONE** function to convert a text item that represents a zoned (signed) number to an integer.

UNZONE converts a text item that can be interpreted as a number with a super-imposed sign in the rightmost byte (called zoned or signed data) to a signed integer item.

Syntax:

UNZONE (single-text-expression)

Meaning:

UNZONE (*text_to_unzone*)

Returns:

A single integer

UNZONE returns a signed integer representing the value *text_to_unzone* that is an integer in zoned (signed) format. If the *text_to_unzone* cannot be interpreted as a valid zoned number, **UNZONE** evaluates to "none".

Zoned integers have a series of digits except for the rightmost byte. The rightmost byte is a digit with a super-imposed sign. See "Positive zoned values" for a list of rightmost byte values.

Examples

- **UNZONE** ("123D")

Returns 1234

- **UNZONE** ("1234")

Returns 1234

- **UNZONE** (*TaxRate*) / 1000 * *Income*

TaxRate is converted from zoned format to a signed decimal and divided by 1000 (to convert it to a number with three decimal places), and then multiplied by **Income**.

You can define items as having a zoned character number presentation. Then, when mapping to or from these items, the conversion to and from zoned decimal is automatically performed as needed.

Related functions

- **ZONE**

ZONE

Use **ZONE** to convert a number to a zoned (signed) number.

The **ZONE** function converts a signed integer item to a text item that can be interpreted as a number with a superimposed sign in the rightmost byte (called zoned or signed).

You can define items as having a zoned character number presentation. Then, when mapping to or from these items, the conversion to and from zoned decimal is performed automatically, as needed.

Syntax:

ZONE (single-integer-expression , single-integer-expression)

Meaning:

ZONE (integer_to_convert , sign_indicator)

Returns:

A single text item

ZONE returns a text string that represents a zoned (signed) number representing *integer_to_convert*.

Zoned integers have a sequence of digits except for the rightmost byte. The rightmost byte is a digit with a super-imposed sign. The *sign_indicator* specifies whether a super-imposed sign is required for positive integers, where 0 specifies no sign is required and any other value specifies that a sign is required for positive integers.

The following tables show positive and negative values of the numbers 1230 to 1239, with a sign indicator of 0 (no sign is required for positive values) or a sign of 1 (include a sign for both positive and negative values) in the rightmost byte.

Table 1. Positive Zoned Values

Integer Value	sign_indicator = 0	sign_indicator = 1
1230	1230	123{
1231	1231	123A
1232	1232	123B
1233	1233	123C
1234	1234	123D
1235	1235	123E
1236	1236	123F
1237	1237	123G
1238	1238	123H
1239	1239	123I

Table 2. Negative Zoned Values

Integer Value	sign_indicator = 0	sign_indicator = 1
-1230	123{	123}
-1231	123J	123J
-1232	123K	123K
-1233	123L	123L
-1234	123M	123M
-1235	123N	123N
-1236	123O	123O
-1237	123P	123P
-1238	123Q	123Q
-1239	123R	123R

Examples

- **ZONE** (1234,0)
Returns 1234
- **ZONE** (1234,1)
Returns 123D
- **ZONE** (-1234,1)
Returns 123M
- **ZONE** (INT(UnitPrice * 100),1)
UnitPrice is multiplied by 100 to move the first two decimal places to the left of the decimal sign. The result of this calculation is converted to an integer using the **INT** function. Finally, the result of the **INT** is converted to zoned format, using a sign for positive values.

Related function

- **UNZONE**

Date/time functions

- [ADDDAYS](#)
- [ADDHOURS](#)
- [ADDMINUTES](#)
- [CURRENTDATE](#)
- [CURRENTDATETIME](#)

- [CURRENTTIME](#)
 - [DATETONUMBER](#)
 - [DATETOTEXT](#)
 - [FROMDATETIME](#)
 - [MAX](#)
 - [MIN](#)
 - [NUMBERTODATE](#)
 - [TEXTTODATE](#)
 - [TEXTTOTIME](#)
 - [TIMETOTEXT](#)
 - [TODATETIME](#)
-

ADDDAYS

The **ADDDAYS** function adds a specified number of days to a given date.

You cannot use **ADDDAYS** on a date/time object that does not specify the day of the month because an error will occur.

Syntax:

ADDDAYS (single-date-expression, single-integer-expression)

Meaning:

ADDDAYS (any_date, number_of_days_to_add)

Returns:

A single date

The **ADDDAYS** function returns the date, which results from adding *number_of_days_to_add* to *any_date*. If *any_date* has a presentation that does not contain a century, the century is determined based on the Century > CCLookup map setting (when the Century > Switch = ON), or the current century (when the Century > Switch = OFF).

When the year exceeds four digits, the output will display hash characters (#) for the CCYY values because the field is defined to be only four digits in length.

Examples

- You can use the **ADDDAYS** function to increment a date by a fixed number of days, such as to calculate a **DueDate** that is always 30 days after the **InvoiceDate**. To produce a date in the output, such as a ship date, you might need to add a variable number of days to a date in the input. For example: **ADDDAYS** (PODate, LeadTime).
- **ADDDAYS** (InvoiceDate, 10)
Returns the date, which results from adding 10 days to the value of **InvoiceDate**.
- **ADDDAYS** (InvoiceDate, DaysTilDue)
Returns the date that results from adding the value of **DaysTilDue** to the **InvoiceDate** value.
- **ADDDAYS** (TODATETIME ("000101"), -1)
Returns 991231
- **ADDDAYS** (TODATETIME ("20000101"), 1)
Returns 20000102
- **ADDDAYS** (TODATETIME ("10/20/1996", "[MM/DD/CCYY"])), 5
Returns 10/25/1996

In the examples containing **TODATETIME**, the text literal is first converted to a date, and then the specified number of days is added to that date.

Related functions

- [ADDHOURS](#)
 - [DATETONUMBER](#)
 - [NUMBERTODATE](#)
 - [TEXTTODATE](#)
 - [TODATETIME](#)
-

ADDHOURS

The **ADDHOURS** function returns a time value that is the result of adding a specified number of hours to a given time.

Syntax:

ADDHOURS (single-datetime-item-expression, hours-expressed-as-signed-integer-expression)

Meaning:

ADDHOURS (any_datetime, number_of_hours_to_add)

Returns:

A single datetime item

The **ADDHOURS** function returns a single datetime item that is advanced from the original value of the single-datetime-item-expression by the specified number of hours-expressed-as-signed-integer-expression.

Examples

- **ADDHOURS (TODATETIME ("Dec 31, 1999 23:59:00"), 2)**
Returns: Jan 1, 2000 01:59:00
- **ADDHOURS (TODATETIME ("Dec 31, 1999 23:59:00"), -25)**
Returns: Dec 30, 1999 22:59:00
- **ADDHOURS (TODATETIME ("23:59:00"), 2)**
Returns: 01:59:00

If either argument is invalid, the result is "none". If the second argument is "none", the result is the first argument. If the first argument contains only a time portion, date changes are ignored. If the first argument contains only a date portion, the time portion 00:00:00 will be used in the calculation. If both arguments are "none", the result is "none".

Related functions

- **ADDDAYS**
- **DATETONUMBER**
- **NUMBERTODATE**
- **TEXTTODATE**
- **TODATETIME**

ADDMINUTES

The **ADDMINUTES** function returns a time value that is the result of adding a specified number of minutes to a given time.

Syntax:

ADDMINUTES (single-datetime-item-expression, minutes-expressed-as-signed-integer-expression)

Meaning:

ADDMINUTES (any_datetime, number_of_minutes_to_add)

Returns:

A single datetime item

The **ADDMINUTES** function returns a single datetime item, advanced from the original value of the single-datetime-item-expression by the specified number of minutes-expressed-as-signed-integer-expression.

Examples

- **ADDMINUTES (TODATETIME ("Dec 31, 1999 23:59:00"), 2)**
Returns: Jan 1, 2000 00:01:00
- **ADDMINUTES (TODATETIME ("Dec 31, 1999 23:59:00"), -25)**
Returns: Dec 31, 1999 23:34:00
- **ADDMINUTES (TODATETIME ("23:59:00"), 2)**
Returns: 00:01:00

If either argument is invalid, the result is "none". If the second argument is "none", the result is the first argument. If the first argument contains only a time portion, date changes are ignored. If the first argument contains only a date portion, the time portion 00:00:00 will be used in the calculation. If both arguments are "none", the result is "none".

Related Functions

- **DATETONUMBER**
- **NUMBERTODATE**
- **TEXTTODATE**
- **TODATETIME**

CURRENTDATE

You can use **CURRENTDATE** when you need the current date as a transaction processing date, an order received date, or other date that reflects when the data was mapped.

When you need to parse the system date, use **CURRENTDATE** with the **TEXT** or **FROMDATETIME** functions. The function returns the current system date.

Syntax:

CURRENTDATE ()

Meaning:

CURRENTDATE ()

Returns:

A single date

CURRENTDATE has no arguments but it does require parentheses.

If being assigned to a date/time output item, the current date is returned in the format specified by that output item. Otherwise, the system date is returned in an MM/DD/YY presentation.

Examples

- **StartDate** = CURRENTDATE ()

In this example, **StartDate** is assigned the value of the current **date**. In this example, because **CURRENTDATE** is assigned to an output, it is automatically converted to the presentation of **StartDate**.

If **CURRENTDATE** evaluates to 06/24/37 and **StartDate** has a YYMMDD presentation, the result is 370624.

- **FROMDATETIME (CURRENTDATE () , "DD.MON.CCYY")**

In this example, the current date is returned in DD.MON.CCYY format. If today's date is January 5, 1999, the date returned would be 05.JAN.1999.

Related functions

- CURRENTDATETIME
- CURRENTTIME
- FROMDATETIME
- DATETOTEXT

CURRENTDATETIME

The **CURRENTDATETIME** function returns the current system date and time. If the system is configured to use daylight saving time, then it is used if daylight saving time is in effect.

You can use this function when you need to map the current system date and time to an item that includes both a date and time portion.

Syntax:

CURRENTDATETIME ([single-text-expression])

Meaning:

CURRENTDATETIME ([date_time_format_string])

Returns:

A single date-time

CURRENTDATETIME has no arguments but it does require parentheses.

The **CURRENTDATETIME** function returns the system date and time in the format specified by *date_time_format_string* or with a CCYYMMDDHHMMSS presentation if no *date_time_format_string* is provided.

The *date_time_format_string* must conform to the date/time format strings as described in "Format strings".

Note: The output format for the native schema XML date/time field is always:

{CCYY-MM-DD}T{HH24:MM:SS.3-3+/-ZZ:ZZ}

To specify a different output format, use Xerces XML instead.

Examples

- **StartTime** = CURRENTDATETIME ()

In this example, **StartTime** is assigned the value of the current date and time. Because **CURRENTDATETIME** is assigned to an output, it is automatically converted to the presentation of **StartTime**.

If **CURRENTDATETIME** evaluates to 3:04pm on 6/24/1999 and **StartTime** has a YYMMDDHH12MM presentation, the result is 9906240304.

- **CURRENTDATETIME ("MM.DD.CCYY HH24:MM")**

In this example, the current date is returned in MM.DD.CCYY HH24:MM format. If it is currently 4:12 pm on January 5, 1999, the date returned would be 01.05.1999 16:12.

Related Functions

- CURRENTDATE
- CURRENTTIME
- FROMDATETIME
- DATETOTEXT

CURRENTTIME

The **CURRENTTIME** function returns the current system time. If the system is configured to use daylight saving time, then it is used if daylight saving time is in effect.

You can use **CURRENTTIME** when you need the system time as a transaction processing time, an order-received time, or a time that reflects when the data was mapped.

Syntax:
CURRENTTIME ()

Meaning:
CURRENTTIME ()

Returns:
A single time

The **CURRENTTIME** function returns the system time. If assigned to a date-time output item, the current time is returned in the format specified by that output item. Otherwise, the system time is returned in HH:MM:SS presentation.

Note: **CURRENTTIME** has no arguments but it does require parentheses.

Examples

- End Time = CURRENTTIME ()

In this example, **End Time** is assigned the current time. Because **CURRENTTIME** is assigned to an output, it is automatically converted to the presentation of **End Time**.

If **CURRENTTIME** evaluates to 10:15:02 and **End Time** has an HH12:MM presentation, the result is 1015.

- FROMDATETIME (CURRENTTIME () , "{HH24MMSS}")

In this example, the current time is returned in HH24MMSS format. If the current time is 10:15:02 pm, the result is 221502.

Related functions

- CURRENTDATETIME
- FROMDATETIME
- DATETOTEXT

DATETONUMBER

Use **DATETONUMBER** to perform arithmetic on dates.

The **DATETONUMBER** function returns an integer that results from counting the number of days from December 31, 1864, to the specified date.

Syntax:
DATETONUMBER (single-date-expression)

Meaning:
DATETONUMBER (date_to_convert)

Returns:
A single integer

The **DATETONUMBER** function converts a date to an integer. The resulting integer represents the number of days since December 31, 1864, where using **DATETONUMBER** with a date of January 1, 1865 returns the integer value 1.

If the input argument is in error, the function returns the value "none".

If the date format specified by the input argument does not include century, the century is determined based on the **Century** map setting using the **CCLookup** parameter or the current century.

Examples

- DaysBetween = **DATETONUMBER** (StopDate) - **DATETONUMBER** (StartDate)

This expression could be used in a map rule to produce a value for **DaysBetween**. It converts the **StopDate** and the **StartDate** to integers, then subtracts the resulting two integers, and returns the result as **DaysBetween**.

Related functions

- ADDDAYS
- NUMBERTODATE

DATETOTEXT

The **DATETOTEXT** function converts a date object or expression to a text item.

Syntax:
DATETOTEXT (single-date-expression)

Meaning:
DATETOTEXT (date_to_convert)

Returns:
A single text item

If *date_to_convert* is a date object name, this returns the date as a text item formatted according to the presentation of the date object.

If *date_to_convert* is a date expression produced by a function, this returns the date as a text item formatted according to the presentation of the output argument of that function.

Examples

- **DATETOTEXT** (*ShipDate*)

In this example, **ShipDate** is converted from a date to text. If **ShipDate** has a CCYYMMDD presentation, the resulting text item will have that presentation, as well.

- **DATETOTEXT** (**CURRENTDATETIME** ("{MM/DD/CCYY}"))

In this example, **CURRENTDATETIME** evaluates and returns a date in MM/DD/CCYY format. Then **DATETOTEXT** evaluates and returns a text string that is that date in MM/DD/CCYY format.

For example, use **DATETOTEXT**, to do text concatenation. The **FROMDATETIME** function provides greater flexibility in specifying the format of the resulting text item.

Related Functions

- **FROMDATETIME**
- **NUMBERTOTEXT**
- **TEXT**
- **TEXTTODATE**
- **TEXTTONUMBER**
- **TEXTTOTIME**
- **TIMETOTEXT**
- **TODATETIME**

FROMDATETIME

The **FROMDATETIME** function converts a date/time item to a text string of a specified format. For example, you can use **FROMDATETIME** to convert a date-time item into a string for parsing or concatenation. You can also use this function to access the individual parts of a date or time, such as the month, day, year, and so forth.

Syntax:

FROMDATETIME (*single-character-date-time-item*
[, *single-text-expression*])

Meaning:

FROMDATETIME (*date_time* [, *date_time_format_string*])

Returns:

A single character text item

FROMDATETIME returns the date/time specified by *date_time* as a text item with the format specified by *date_time_format_string*. If *date_time_format_string* is not specified, *date_time* will be returned in the same format as the *date_time*.

The *date_time_format_string* must conform to the date/time format strings as described in the "Format strings" topic.

Examples

- HeaderLine="Today is" + **FROMDATETIME** (**CURRENTDATE** (), "{MON DD, CCYY}")
If the current system date is March 3, 1999, this rule evaluates to Today is Mar 03, 1999.

- **FROMDATETIME** (*TransactionTimeStamp Column::TransHistory*)

If the value of **TransactionTimeStamp Column** is 10:14 A.M. on February 3, 2000 and its format is defined as "{CCYYMMDDHH24MM}", this rule evaluates to 200002031014.

- **EXTRACT** (*Expense:Report*, **FROMDATETIME** (*Date:Expense:Report*, "[MM]") = "03")

In this example, the **FROMDATETIME** function is used in the condition expression of the **EXTRACT** function to identify all expenses in the month of March.

Related functions

- **CURRENTDATE**
- **CURRENTDATETIME**
- **CURRENTTIME**
- **DATETOTEXT**
- **TODATETIME**

MAX

The **MAX** function returns the maximum value from a series of number, date, time, or text values.

Syntax:

MAX (*series-item-expression*)

Meaning:

MAX (*series_of_which_to_find_max*)

Returns:

A single number

The result is the maximum value in the input argument series: number, text, or date/time.

Examples

- **MAX** (UnitPrice:Input)
If the values for **UnitPrice** are {20, 10, 100}, MAX returns 100.
- **MAX(EXTRACT(** DueDate:Book:Library, CheckedOut:Book:Library = "Y"))
Returns the maximum (latest) **DueDate** for a book that is checked out from the library.

Related functions

- **MIN**

MIN

Use MIN when you need the minimum value from a series of number, date, time, or text values.

The MIN function returns the minimum value from a series.

Syntax:
MIN (series-item-expression)

Meaning:
MIN (series_of_which_to_find_min)

Returns:
A single number

The result is the minimum value of the input series: number, text, or date/time.

Examples

- **MIN** (UnitPrice:Input)
If the values for **UnitPrice** are {20,10,100}, MIN returns 10.
- **MIN** (StartTime:::Schedule)
Returns the minimum (earliest) **StartTime** in **Schedule**.

Related functions

- **MAX**

NUMBERTODATE

You can use the **NUMBERTODATE** function to perform a calculation on a date.

NUMBERTODATE converts an integer to a date, where the integer is the number of days from December 31, 1864, to the specified date. Only positive, non-zero values can be specified as valid parameters.

Syntax:
NUMBERTODATE (single-integer-expression)

Meaning:
NUMBERTODATE (integer_to_convert)

Returns:
A single date

After converting an integer to a date, if the result is being assigned to a date object, the resulting date is in the presentation for that output. If the resulting date is not being assigned to an object, it has a CCYYMMDD presentation.

When the year exceeds four digits, the output will display hash characters (#) for the CCYY values because the field is defined to be only four digits in length.

Examples

- **NUMBERTODATE** (StartDate)
This example converts the **StartDate** value from an integer to a date that is in CCYYMMDD format.

Related functions

- **ADDDAYS**
- **DATETONUMBER**

- **FROMDATETIME**
- **TODATETIME**

TEXTTODATE

The **TEXTTODATE** function converts text item from CCYYMMDD or YYMMDD format to a date.

You can use the **TEXTTODATE** to convert a text item to a date, however, the **TODATETIME** function provides greater flexibility for specifying the resulting date-time format.

Syntax:

TEXTTODATE (single-text-expression)

Meaning:

TEXTTODATE (text_to_convert_to_date)

Returns:

A single date

The *text_to_convert_to_date* must be in either CCYYMMDD or YYMMDD format. If the *text_to_convert_to_date* is in error (for example, it is not a valid date), the result is "none".

If *text_to_convert_to_date* is in YYMMDD format and is being assigned to a date format that includes a century, the century is determined based on the century run option setting using the **CCLookup** parameter or the current century.

Examples

- **TEXTTODATE ("990114")**
Returns a single date item with the value of January 14 in the year 99
- **TEXTTODATE (OrderDate)**
Returns **OrderDate** as a single date item.

Related functions

• DATETOTEXT	• TEXTTOTIME
• NUMBERTOTEXT	• TIMETOTEXT
• TEXTTONUMBER	• TODATETIME

TEXTTOTIME

Use **TEXTTOTIME** when you want to convert an object defined as text that is in HHMM or HHMMSS presentation, to an item defined as time. For greater flexibility, use the **TODATETIME** function for specifying the format of the text item that is to be converted to a date/time.

Syntax:

TEXTTOTIME (single-text-expression)

Meaning:

TEXTTOTIME (text_to_convert_to_time)

Returns:

A single time

The *text_to_convert_to_time* must be in HHMM or HHMMSS presentation. HH is a two-digit hour in a 24-hour format. If the result is being assigned to a time object, the resulting time looks like the output object. Otherwise, the resulting time looks like the input argument. If the input argument is in error (for example, it is not a valid time), the result is "none".

Examples

- **TEXTTOTIME (CallTime)**
Returns **CallTime** as a time item

Related functions

• DATETOTEXT	• TEXTTONUMBER
• FROMDATETIME	• TIMETOTEXT
• NUMBERTOTEXT	• TODATETIME
• TEXTTODATE	

TIMETOTEXT

You can use the **TIMETOTEXT** function to perform text concatenation. For greater flexibility, use the **FROMDATETIME** function for specifying the format of the resulting text item.

TIMETOTEX converts a time object or expression to a text item.

Syntax:

TIMETOTEXT (single-time-expression)

Meaning:

TIMETOTEXT (time_to_convert_to_text)

Returns:

A single text item

If *time_to_convert_to_text* is a time object name, this returns the time as a text item formatted according to the presentation of the input date object.

If *time_to_convert_to_text* is a time expression produced by a function, this returns the time as a text item formatted according to the presentation of the output argument of that function.

Examples

- **TIMETOTEXT** (LeadTime)

In this example, **LeadTime** is converted from a time to text. If **LeadTime** has an HH:MM presentation, the resulting text item will be of that presentation.

- **TIMETOTEXT** (CURRENTDATETIME ("{HH:MM:SS}"))

Here, **CURRENTDATETIME** evaluates and returns a time in HH:MM:SS format. Then, **TIMETOTEXT** evaluates and returns a text string that is that time in HH:MM:SS format.

Related functions

- **DATETOTEXT**
- **FROMDATETIME**
- **NUMBERTOTEXT**
- **TEXTTODATE**
- **TEXTTONUMBER**
- **TEXTTOTIME**
- **TODATETIME**

TODATETIME

The **TODATETIME** function converts a text string of a specified format to a date-time item.

Syntax:

TODATETIME (single-character-text-expression
[, single-text-expression])

Meaning:

TODATETIME (text_to_convert [, date_time_format_string])

Returns:

A single character date item

TODATETIME returns the date-time that corresponds to the value specified by *text_to_convert*, which is in the format specified by *date_time_format_string*. If *date_time_format_string* is not specified, it will be assumed that *text_to_convert* is in [CCYYMMDDHH24MMSS] format.

The *date_time_format_string* must conform to the date-time format strings as described in "Format strings".

Examples

- **TODATETIME** ("05/14/1999@10:14pm" , "{MM/DD/CCYY}@{HH12:MMAM/PM}")

In this example, a text string containing a date and time is converted to a date-time item.

- RptDate = **TODATETIME** (**RIGHT** (**GETRESOURCENAME** () , 8) , "CCYYMMDD")

Assume that you receive a file that contains historical data. The name of the file identifies the date of the historical data. For example, a filename of **19960424** indicates that the data was produced on April 24, 1996. To map this date to **RptDate**, the **TODATETIME** function could be used with the **RIGHT** and **GETRESOURCENAME** functions.

Related functions

- **CURRENTDATE**
- **CURRENTDATETIME**
- **CURRENTTIME**
- **TEXTTODATE**
- **TEXTTOTIME**

Error handling functions

- [**CONTAINSERRORS**](#)
- [**FAIL**](#)
- [**GETXMLERRORMSG**](#)
- Use this function on an output card to return an XML validation error message when a map runs against invalid XML input.
- [**ISERROR**](#)
- [**ONERROR**](#)
- [**REJECT**](#)
- [**QUIT**](#)
- [**VALID**](#)

CONTAINSERRORS

The **CONTAINSERRORS** function tests a valid object to see whether it contains any objects in error.

Syntax:

CONTAINSERRORS (single-object-expression)

Meaning:

CONTAINSERRORS (object_to_test)

Returns:

True or false

The **CONTAINSERRORS** function returns "true" if any object contained in *object_to_test* is in error; it returns "false" if the *object_to_test* is completely valid.

The input object, itself, is a *valid* input object. This function does *not* evaluate for invalid objects. Therefore, if you have map rule:

CONTAINSERRORS (Invoice:InputFile)

and **Invoice[1]** is valid, **Invoice[2]** is invalid, and **Invoice[3]** is valid, then **CONTAINSERRORS** evaluates only twice-once for each *valid* instance of **Invoice**.

Examples

- `Msg (s) = IF (CONTAINSERRORS (Msg:MailBag) &`
`Type:Msg:MailBag = "PRIORITY" ,`
`Msg:MailBag ,`
`"none")`

In this example, if **Msg:MailBag** contains any object in error and **Type:Msg:MailBag** has a value of "PRIORITY", **Msg:MailBag** is mapped; otherwise, "none" is returned for this occurrence of **Msg**. This map rule returns all valid messages (**Msg**) that contain errors with a **Type** of "PRIORITY".

Related functions

- [**ISERROR**](#)
- [**REFORMAT**](#)

FAIL

You can use the **FAIL** function to abort a map based on map or application specific logic.

Syntax:

FAIL (single-text-expression)

Meaning:

FAIL (message_to_return)

Returns:

"None"

The **FAIL** function returns "none" to the output to which the function is assigned, aborts the map, and returns *message_to_return* as the map completion error message included in the execution audit. The map return code will be "30", indicating that the map failed through the **FAIL** function.

Examples

- `AcctID = EITHER (LOOKUP (CustomerID::Xref , MyKey::Xref = ID::Input) ,`
`FAIL ("Unknown Customer (" + ID::Input + "). Processing Terminated.")`

In this example, the **FAIL** function is being used in conjunction with the **EITHER** function to conditionally fail the map if a record in the customer cross-reference file does not exist for a given **CustomerID**.

For example, the **ID** in **Input** is ABC123. If the **LOOKUP** succeeds, the **CustomerID** result is assigned to **AcctID** and the map continues.

If the **LOOKUP** fails, **AcctID** is assigned a value of "none", the map aborts, and the message Unknown Customer (ABC123). Processing Terminated.is written to the execution audit log.

- Message = **VALID** (**RUN** ("Map1Msg.mmc", "-AE -OMMSMQ1B ` -QN .\aqueue -CID 2001"),
FAIL ("Failure on **RUN** (" + TEXT (**LASTERRORCODE** ()) + ":" +
LASTERRORMSG ()))

In this example, the **FAIL** function is being used in conjunction with the **VALID**, **LASTERRORCODE**, and **LASTERRORMSG** functions to fail (abort) the map if the map executed by the **RUN** function (**Map1Msg.mmc**) fails. In this example, the map fails and returns the error code and error message reported by the **RUN** function using the **LASTERRORCODE** and **LASTERRORMSG** functions.

If **Map1Msg** fails because one or more of its inputs was invalid, **Message** is assigned a value of "none". The map aborts and the following message is reported in the execution audit log:

"Failure on **RUN** (8): One or more inputs was invalid."

Related functions

- **LASTERRORCODE**
- **LASTERRORMSG**
- **QUIT**
- **VALID**

GETXMLERRORMSG

Use this function on an output card to return an XML validation error message when a map runs against invalid XML input.

Create the input card with either a native XML schema or an XML-based type tree, and specify the Restart attribute. On the output card, use the **GETXMLERRORMSG** function, passing the number of the input card as the parameter. When the map runs and the input contains invalid XML data, the output card returns a standard XML error message. For example:

```
Error (-1), "XMLParser: Input XML data is invalid."
SAXParseException, Error [line: 162 column: 36]
Datatype error: Type:SchemaDateTimeException,
Message:Incomplete Date ! '2006-09-' .
```

See the Type Designer documentation or the Map Designer documentation for details about the Restart attribute.

Syntax

GETXMLERRORMSG (single-number-expression)

Meaning

GETXMLERRORMSG (input_card_number)

Returns

A text string

Related functions

- **VALIDATE**

ISERROR

The **ISERROR** function tests an object to see if it is in error

You can use **ISERROR** to output your data in exactly the same order as it occurs in your input, both the valid data and the data in error. You can also use **ISERROR** to produce error messages for bad data in the same file in which you map your good data.

Syntax:

ISERROR (single-object-name)

Meaning:

ISERROR (object_to_test)

Returns:

"True" or "false"

ISERROR returns "true" when *object_to_test* is in error and returns "false" when *object_to_test* is completely valid.

Examples

- InfoRec (s) = IF (**ISERROR** (Record:SomeFile), "Bad --> " + **REJECT** (Record:SomeFile), "Ok --> " + TEXT (Record:SomeFile))

In this example, **ISERROR** is used to produce a report for all **Record** objects in **SomeFile**. If the record is in error, the **InfoRec** will have the text Bad --> followed by the data from the input **Record**. If the record is valid, the **InfoRec** will have the text Ok --> followed by the data from the input **Record**, such as:

```
Ok --> SZ-68839,486 Upgrade Microprocessor,186.86,100,W200
Bad --> MK-19309,,369.43,417,W100
Ok --> KL-20349,PCMCIA Network Adaptor,174.82,29,N300
Ok --> WP-37679,AC Adaptor,39.48,245,E100
Bad --> IL-39890,8MB Memory PCMCIA,390.48,0,S100
```

Related functions

- **CONTAINSERRORS**

ONERROR

Use ONERROR in component rules to add user-defined error messages to the data section of the audit log.

The **ONERROR** function adds a user-defined error message to the data section of the audit log. This function is relevant only in Type Designer Component rules.

Syntax:

ONERROR (single-condition-expression , single-text-expression)

Meaning:

ONERROR (condition_to_evaluate , message_to_display)

Returns:

"True" or "false"

If *condition_to_evaluate* evaluates to "true", the function returns "true".

If *condition_to_evaluate* evaluates to "false", the function returns "false".

If data audit is enabled and the object to which the component rule applies will result in a failed component rule message (status E09), *message_to_display* is written to the data audit section of the audit log with an entry type of U, representing user-defined.

If the failed component rule message (E09) appears in the data section of the audit log, one or more user-defined error messages can also be included in the data section of the audit log. If a component rule has one **ONERROR** function, one user-defined error message or none is included in the audit log. If a component rule has two **ONERROR** functions, two user-defined error messages at most are included in the audit log, and so on.

Examples

- Here is a component rule without **ONERROR**:

Claim Date Field > Accident Date Field &

Claim Date Field < **ADDDAYS** (Accident Date Field, 365)

If the component rule fails, the data section of the audit log shows the following information:

```
<DataLog>
<input card="1">
  <object ... status="E07">InsuranceClaim</object>
  <object ... status="E09">Claim Date Field</object>
    <Text>980725</Text>
  <object ... status="E07">InsuranceClaim</object>
  <object ... status="E09">Claim Date Field</object>
    <Text>990526</Text>
</input>
</DataLog>
```

Status code E09 for the **Claim Date Field** indicates that the data for that object failed its component rule, but does not provide any further detail.

- Using **ONERROR**, one or more error messages can be added to the data section of the audit log to provide more information as to how or why the component rule failed. For example,
ONERROR (Claim Date Field > Accident Date Field ,
"Claim date before accident.") &
ONERROR (Claim Date Field < **ADDDAYS** (Accident Date Field , 365) ,
"Claim is more than one year old.")

If the component rule fails and **ONERROR** is used, the data section of the audit log can show the following messages:

```
<DataLog>
<input card="1">
  <object ... status="E07">InsuranceClaim</object>
  <object ... status="E09">Claim Date Field</object>
    <Text>980725</Text>
    <User>Claim date before accident.</User>
  <object ... status="E07">InsuranceClaim</object>
  <object ... status="E09">Claim Date Field</object>
    <Text>990526</Text>
    <User>Claim is more than one year old.</User>
</input>
</DataLog>
```

REJECT

The **REJECT** function returns the content of an object in error as a text item. Use **REJECT** in conjunction with the restart attribute. You can use the **REJECT** function in map rules only, not in component rules.

See the Type Designer and Map Designer documentation for information about the restart attribute.

Syntax:

REJECT (series-object-expression)

Meaning:

REJECT (series_to_look_for_bad_objects)

Returns:
A series text item

REJECT evaluates to a series of text items consisting of all the input series members in error.

Examples

- **REJECT** (Record:File)
This example extracts all **Records** that are in error.
- IF (COUNT (REJECT (Msg IN Batch))) = 0, "OK", "ERROR")
In this example, the total number of invalid (rejected) Msg objects is counted. If the total number of invalid Msg objects is equal to zero, it indicates that there were no invalid Msg objects counted and a message of OK results. Otherwise, a message of ERROR results.

Related functions

- **CONTAINSERRORS**
- **ISERROR**
- **VALID**

QUIT

You can use the **QUIT** function to stop a map at a specific point during map execution. The **QUIT** function logs a successful return code and a custom message in the map audit log.

Syntax:
QUIT (single-text-expression)

Meaning:
QUIT (message_to_return)

Returns:
"None"

The **QUIT** function returns "none" to the output, stops the map, and logs map return code 41 and the message **QUIT** function terminated map successfully: *message_to_return* in the map audit log.

Related functions

- **FAIL**
- **LASTERRORCODE**
- **LASTERRORMSG**
- **VALID**

VALID

You can use the **VALID** function to perform conditional processing based on whether an external interface function executes successfully.

VALID returns the result of the first argument if it is valid; otherwise, returns the second argument.

Syntax:
VALID (single-text-expressions , single-general-expression)
Meaning:
VALID (function_that_can_fail, return_value_if_function_fails)
Returns:
A single text expression

VALID returns the result of the evaluation of *function_that_can_fail* if it is valid. If the function fails, **VALID** returns *return_value_if_function_fails*.

The following functions can fail:

• DBLOOKUP	• GET
• DBQUERY	• PUT
• DDEQUERY	• RUN
• EXIT	

Examples

- SomeObject = VALID (RUN ("mymap.mmc" , "-OF1 mydata.txt") , FAIL ("My RUN failed!"))

If the **RUN** function returns an error return code, the **VALID** functions returns `none`, the map aborts, and the message `My RUN failed!` is reported under Execution Summary in the execution audit log.

Related functions

- [**DBLOOKUP**](#)
 - [**DBQUERY**](#)
 - [**DDEQUERY**](#)
-

External interface functions

- [**DBLOOKUP**](#)
- [**DBQUERY**](#)
- [**DDEQUERY**](#)
- [**EXIT**](#)
- [**GET**](#)
- [**JEXIT**](#)
- [**PUT**](#)
- [**RUN**](#)
- [**RENAMEFILE**](#)

The **RENAMEFILE** function renames the source file to that of the destination file. If the operation succeeds, the returned value is 0; if it fails, then an operating system-specific error number is returned.

- [**HARDLINKFILE**](#)

A hard link points or references to a specific space on the hard drive. Multiple files can be hard linked to the same space in the hard drive. If some data is changed on one of the hard linked files, all the other files reflect that change. The **HARDLINKFILE** function returns the hard link file's command status. On successful execution of this function: 0 is returned; on failure: an operating system-specific error code is returned.

- [**SOFTLINKFILE**](#)

A soft link points to a specific file, which in turn, references to a particular location on the hard drive. The soft link, also known as symbolic link is a second file that exists independently of its target. The **SOFTLINKFILE** function returns the soft (symbolic) link file's command status. On successful execution of this function: 0 is returned; on failure: an operating system-specific error code is returned.

- [**COPYFILE**](#)

The **COPYFILE** function copies the source file to the destination file. It returns the copy file command status. On successful execution of this function: 0 is returned; on failure: an operating system-specific error code is returned.

- [**SLEEP**](#)

The **SLEEP** function suspends the execution of the current map until the time-out interval elapses. The time-out or sleep interval is represented in milliseconds. On successful execution of this function: 0 is returned; on failure: -1 is returned.

DBLOOKUP

The **DBLOOKUP** function executes an SQL statement against a database. The SQL statement can be any permitted by your database management system or ODBC driver.

When the **DBLOOKUP** function is used in a map, the default **OnSuccess** action is adapter specific. The default **OnFailure** action is to rollback any changes made during map processing. The default **Scope** will be integral unless the map is defined to run in bursts (which is the case when one or more inputs have the **FetchAs** property set to **Burst**).

There are two ways to specify arguments for **DBLOOKUP**.

You can use **DBLOOKUP** to execute an SQL statement when you want to execute a **SELECT** statement to retrieve a specific column value in a large table in a database using the value of another input, rather than defining the entire table as an input card and using the **LOOKUP**, **SEARCHDOWN**, or **SEARCHUP** functions.

You can use DBLOOKUP to execute an SQL statement when you want to execute a SELECT statement to retrieve a specific column value from a table or database that might vary based on a parameter file. Using Meaning 2 of the **DBLOOKUP** function allows these parameters to be dynamically specified at run time.

Syntax:

DBLOOKUP (single-text-expression , single-text-expression , [single-text-literal])

Meaning:

1. **DBLOOKUP** (SQL_statement , mdq_filename , database_name)
2. **DBLOOKUP** (SQL_statement , parameters)

Returns:

A single text item

The **DBLOOKUP** function returns the results of the query in the same format as a query specified for a map input card, except that it does not include the last carriage return/linefeed. Because this information is removed, it is easier to make use of a single value extracted from a database.

Arguments for meaning 1

DBLOOKUP (SQL_statement , mdq_filename , database_name)

- **SQL_statement**

The first argument is an SQL statement as a text string. This can be any valid SQL statement permitted by your database management system and supported by your database-specific driver. In addition to a fixed SQL statement, this argument can be a concatenation of text literals and data objects, enabling the concatenation of data values into your SQL statement.

- **mdq_filename**

The second argument is the name of a database query file (**.mdq**) produced by the Database Interface Designer. It contains the definition of the database that the SQL statement is to be executed against. If the **.mdq** file is in a directory other than the directory of the map, the path must be specified.

Note: The **.mdq** file is accessed at map build time and is not needed at run time.

- **database_name**

The third argument is the name of a database in the database query file (**.mdq**) as defined in the Database Interface Designer.

If used in this way, both the **.mdq** filename and database name must be literals.

Arguments for meaning 2

DBLOOKUP (SQL_statement , parameters)

- **SQL_statement**

The first argument is an SQL statement as a text string. This can be any valid SQL statement permitted by your database management system and supported by your database-specific driver. In addition to a fixed SQL statement, this argument can be a concatenation of text literals and data objects, enabling the concatenation of data values into your SQL statement.

- **parameters**

The second argument is a set of parameters, either:

- -MDQ mdqfilename -DBNAME dbname
 - or-

- -DBTYPE database_type [database specific parameters]

The keyword -MDQ is followed by the name of the database query file (**.mdq**) produced by the Database Interface Designer. This **.mdq** file contains the definition of the database. If the **.mdq** file is in a directory other than the directory of the map, the path must be specified. The **.mdq** filename is followed by the keyword -DBNAME and the database name as specified in the Database Interface Designer.

Using this syntax, the **.mdq** file is accessed at run time and must be present.

The keyword -DBTYPE is followed by a keyword specifying the database type (for example, ODBC or ORACLE) followed, optionally, by database-specific parameters.

This syntax does not use an **.mdq** file, because the database-specific parameters provide the information required to connect to the database. See the Resource Adapters documentation for detailed information about the database-specific parameters that can be specified.

When used with Meaning 2, **DBLOOKUP** must conform to these rules:

- All keywords (for example, -DBTYPE) can be upper or lowercase, but not mixed.
- A space is required between the keyword and its value (for example, -DBTYPE ODBC).
- The order of the keywords is not important.
All database-specific parameters are optional.

Examples

Assume that you have a table named "PARTS" that contains the following data:

PART_NUMBER	PART_NAME
1	1/4" x 3" Bolt
2	1/4" x 4" Bolt

Assume that this database has been defined using the Database Interface Designer in a file named **mytest.mdq** and that the name of the database, as specified in the **.mdq** file, is **PartsDB**.

- **DBLOOKUP** ("SELECT PART_NAME from PARTS where PART_NUMBER =1",
"mytest.mdq",
"PartsDB")
Returns: 1/4" x 3" Bolt

Using Meaning 2, you can specify the **DBLOOKUP** this way:

- **DBLOOKUP**("SELECT PART_NAME from PARTS where PART_NUMBER =1",
"-MDQ mytest.mdq -DBNAME PartsDB")
where both the **.mdq** file name and database name is specified.

Using Meaning 2, you can also specify the database type and the appropriate database-specific parameters:

- **DBLOOKUP**("SELECT PART_NAME from PARTS where PART_NUMBER =1",
"-DBTYPE ORACLE -CONNECT MyDB -USER janes")

Related functions

- **DBQUERY**
- **EXTRACT**
- **FAIL**
- **LASTERRORCODE**
- **LASTERRORMSG**
- **LOOKUP**
- **SEARCHDOWN**
- **SEARCHUP**
- **VALID**

For more examples using the **DBLOOKUP** function, see the Database Interface Designer documentation.

DBQUERY

The **DBQUERY** function executes an SQL statement against a database. The SQL statement can be any permitted by your database management system or ODBC driver.

When the **DBQUERY** function is used in a map, the default **OnSuccess** action is adapter specific. The default **OnFailure** action is to rollback any changes made during map processing. The default **Scope** will be integral unless the map is defined to run in bursts (which is the case when one or more inputs have the **FetchAs** property set to **Burst**).

There are two ways to specify the arguments for **DBQUERY**. You can use **DBQUERY [Meaning 1]** to execute an SQL statement when you want to look up information in a database using a parameterized query that is based on another value in your data. If your SQL statement is a SELECT statement, the **DBQUERY** function might be used in conjunction with the **RUN** function to issue dynamic SELECT statements whose results can be used as input to another map.

You can also use the DBQUERY function [*Meaning 2*] to execute an SQL statement when the database, table, or other database parameters might vary; perhaps being supplied by a parameter file.

Syntax:

```
DBQUERY (single-text-expression , single-text-expression ,
[ single-text-literal ])
```

Meaning:

1. **DBQUERY** (SQL_statement , mdq_filename , database_name)
2. **DBQUERY** (SQL_statement , parameters)

Returns:

A single text item

If your SQL statement is a SELECT statement, the results of the query in the same format as a query specified as a map input card, including row delimiters and terminators, and so on.

If your SQL statement is anything other than a SELECT statement, "none".

Arguments for meaning 1

DBQUERY (SQL_statement , mdq_filename , database_name)

- **SQL_statement**

The first argument is an SQL statement as a text string. This can be any valid SQL statement that is permitted by your database management system and supported by your database-specific driver. In addition to a fixed SQL statement, this argument can be a concatenation of text literals and data objects, enabling the concatenation of data values into your SQL statement.

- **mdq_filename**

The second argument is the name of a database query file (**.mdq**) produced by the Database Interface Designer. It contains the definition of the database that the SQL statement is to be executed against. If the **.mdq** file is in a directory other than the directory of the map, the path must be specified.

Note: The **.mdq** file is accessed at map build time and is not needed at run time.

- **database_name**

The third argument is the name of a database in the database query file (**.mdq**) as defined in the Database Interface Designer.

If used in this way, both the **.mdq** filename and database name must be literals.

Arguments for meaning 2

DBQUERY (SQL_statement , parameters)

- The first argument is an SQL statement as a text string. This can be any valid SQL statement that is permitted by your database management system and supported by your database-specific driver. In addition to a fixed SQL statement, this argument can be a concatenation of text literals and data objects, enabling the concatenation of data values into your SQL statement.

- The second argument is a set of parameters, either:

- -MDQ mdqfilename -DBNAME dbname
 - or-

- -DBTYPE database_type [database specific parameters]

The keyword -MDQ is followed by the name of the database query file (**.mdq**) produced by the Database Interface Designer. This **.mdq** file contains the definition of the database. If the **.mdq** file is in a directory other than the directory of the map, the path must be specified. The **.mdq** filename is followed by the keyword -DBNAME and the database name as specified in the Database Interface Designer.

Note: Using this syntax, the **.mdq** file is accessed at run time and must be present.

The keyword -DBTYPE is followed by a keyword specifying the database type (for example, ODBC or ORACLE) followed, optionally, by database-specific parameters.

Note: This syntax does not use an **.mdq** file, because the database-specific parameters provide the information required to connect to the database. See the appropriate database adapter documentation for detailed information about database-specific parameters.

When used with Meaning 2, **DBQUERY** must conform to these rules:

- All keywords (for example, -DBTYPE) can be upper or lower case, but not mixed.
- A space is required between the keyword and its value (for example, -DBTYPE ODBC).
- The order of the keywords is not important.

All database-specific parameters are optional.

Examples

Assume that you have a table named "PARTS" that contains the following data:

PART_NUMBER	PART_NAME
1	1/4" x 3" Bolt
2	1/4" x 4" Bolt

Also assume that this database has been defined using the Database Interface Designer in a file named **mytest.mdq** and that the name of the database, as specified in the **.mdq** file, is **PartsDB**.

```
DBQUERY ( "SELECT * from  
PARTS" , "mytest.mdq" , "PartsDB" )  
Returns 1|1/4" x 3" Bolt<cr><lf>2|1/4" x 4" Bolt<cr><lf>
```

where <cr><lf> is a carriage return followed by a line feed.

Using Meaning 2, you can also specify the DBQUERY this way:

```
DBQUERY ( "SELECT * from PARTS" , "-MDQ mytest.mdq -DBNAME PartsDB"  
)
```

where both the **.mdq** file name and database name are specified.

Or, specify it this way, using Meaning 2 by specifying the database type and the appropriate database-specific parameters:

```
DBQUERY ( "SELECT * from PARTS" , "-DBTYPE ORACLE -CONNECT MyDB -USER janes"  
)
```

Assume that you have an input file containing one order record. To map that order to another proprietary format, you also have a parts table with pricing information for every part for every customer, a very large table. Rather than using the entire parts table as the input to your map, you might use the **RUN** function with a **DBQUERY** to dynamically select only those rows from the parts table corresponding to the customer in the order file, as follows:

```
RUN ( "MapOrder.MMC" ,  
"IE2" + DBQUERY ( "SELECT * FROM Parts WHERE CustID = "  
+ CustomerNo:OrderRecord:OrderFile + " ORDER BY PartNo" ,  
"PartsDB.MDQ" , "PartsDatabase" ) )
```

Related functions

- DBLOOKUP
- EXTRACT
- FAIL
- LASTERRORCODE
- LASTERRORMSG
- LOOKUP
- SEARCHUP
- SEARCHDOWN
- VALID

DDEQUERY

The **DDEQUERY** function allows you to interface to other Windows applications such as Trading Partner PC, Excel, and so forth, provided that certain criteria are met. For example, if you receive an Excel spreadsheet file, you must have the appropriate version of the Excel application installed (that is compatible with the file received) and the application must be open.

Syntax:

```
DDEQUERY (single-text-expression , single-text-expression , single-text-expression)
```

Meaning:

```
DDEQUERY (application_name , topic , text)
```

Returns:

A single text item from an application

Examples

- **DDEQUERY ("excel" , "[MKTPRICE.XLS]Sheet1" , "R8C1:R14C3")**

In this example, **DDEQUERY** is used to get data from an Excel spreadsheet. The third argument, **R8C1:R14C3**, specifies the location of the data in the spreadsheet. (In Excel, the 8th row, 1st column to the 14th row, 3rd column is A8:C14.) The content of this spreadsheet range is returned as a single text item.

This example assumes that the application, the map, and the spreadsheet all reside in the same directory. If they are not in the same directory you must add the path. For example:

```
DDEQUERY ("excel" , "c:\spreadsheet[MKTPRICE.XLS]Sheet1" , "R8C1:R14C3")
```

- **DDEQYUERY ("tppc","PartnerX","BGyourEDIode")**

In this example, **DDEQUERY** is used as a request to Trading Partner PC.

Related functions

- EXIT

- FAIL
 - GET
 - LASTERRORCODE
 - LASTERRORMSG
 - PUT
 - RUN
 - VALID
-

EXIT

The **EXIT** function allows you to interface with a function in an external library or application.

You can use EXIT when you need information from an existing function in a library or a program or when you need to use a general function that is not available.

Depending on the execution operating system, there are two different methods for the **EXIT** function: 1) the library method and 2) the program method. The program method is not supported on Windows operating systems.

Syntax:

EXIT (single-text-expression, single-text-expression,

single-text-expression)

Meaning:

1. **EXIT** (library_name, function_name, input_to_the_function)
2. **EXIT** (program_name, command_line_arg1, command_line_arg2)

Returns:

A single text item

Meaning 1 - library method

At run time, the *function_name* function will execute in the library specified by *library_name* passing *input_to_the_function* as a text string. The result of *function_name* is returned as a text item by means of *lpep->lpdataFromApp*.

Set *lpep->nReturn* equal to 0 if the function is to succeed or set it equal to -1 to fail.

For detailed information about the requirements of the library function that is executed by the **EXIT**, see [Implementing a library EXIT function](#).

AIX operating system

On the AIX operating system, the shared library that contains *library_function* must also contain an *entry_point* function. The *entry_point* function must be prototyped in the same manner as *library_function*. The server will invoke the *entry_point* function before invoking *library_function*. The server passes the name of the *library_function* to the *entry_point* function by way of the *szFile* EXITPARAM structure member. The *entry_point* function must then determine the address of the *library_function* and pass this address back to the server by way of the *lpv* EXITPARAM structure member. The server will then use the contents of *lpv* as an address to invoke the *library_function*.

The following example depicts a shared library called **mcshex.so** which contains the *entry_point* function and the *library_function*.

```
#include <stdio.h>
#include <string.h>
#include "runmerc.h"
void AIXEntry(LPEXITPARAM);
void bin2hex(LPEXITPARAM);
unsigned int i;
char *syms[] = { "bin2hex", "AIXEntry" };
void *adr[] = { (void *)bin2hex, (void *)AIXEntry };
void AIXEntry(LPEXITPARAM lpInputStruct)
{
    printf("AIXEntry called\n");
    for (i = 0; i < (sizeof(syms)) / (sizeof(char *)); i++)
        if (!strcmp(lpInputStruct->szFile, syms[i])) {
            lpInputStruct->lpv = adr[i];
            break;
        }
}
void bin2hex(LPEXITPARAM lpInputStruct)
{
    printf("bin2hex called\n");
    printf("argument to library function: %s\n", lpInputStruct->lpszCmdLine);
    lpInputStruct->nReturn = 0;
}
```

To build this shared library, run the following commands:

cc -c share1.c

cc -o mcshex.so share1.o -bE:shrsu.exp -bM:SRE -eAIXEntry

Meaning 2 - program method

The program method of the **EXIT** function is *not* supported on Windows operating systems.

At execution time, the program specified by *program_name* executes and passes the concatenation of *command_line_arg1 + " " + command_line_arg2* as a text string.

Whatever is returned by *program_name* to the standard output device is returned as text.

Examples

- **EXIT** (*program_name*, *command_line_arg1*, *command_line_arg2*)
Returns a text string from the function or application that is executed. If the EXIT function is not available for a particular operating system, EXIT returns `none`.
- **EXIT** ("mydll.dll", "myfunction", "12")
This Windows library example passes the value 12 to **myfunction**, a function in **mydll.dll**. The value of the item returned depends on what **myfunction** does with the 12 passed to it.
- IF (**EXIT** ("mylib.sl", "ckCust", CustID Column:Row:DB) = "OK", MapKnownCust (Row:DB), MapUnknownCust (Row:DB))
Similarly, this UNIX library example passes the value of **CustID** Field to **ckCust**, a function in a UNIX shared library called **mylib.sl**. If the value returned by **ckCust** is OK, a functional map is called to map **Row** for a known customer. Otherwise, another functional map is executed to map **Row** for an unknown customer.
- **EXIT** ("pwd", "", "")
This UNIX program example executes the UNIX print working directory (pwd) command to determine the current directory. The name of the current working directory is then returned as a text string. Notice that although the pwd command does not require additional command line arguments, *command_line_arg1* and *command_line_arg2* must be included as a space enclosed in double quotation marks ("").
- **TEXTT tonumber**((**EXIT** ("GetIncome", Applicant:Form, "*Mortgage")))
This program example passes the value of Applicant concatenated to the text literal *Mortgage to an application called GetIncome. The result is converted to a number.

Related functions

- **DDEQUERY**
- **FAIL**
- **GET**
- **LASTERRORCODE**
- **LASTERRORMSG**
- **PUT**
- **RUN**
- **VALID**
- [Implementing a library EXIT function](#)

Implementing a library EXIT function

You can develop a function in a library to be executed from an **EXIT** function.

- [EXIT function's library interface](#)
- [Using the EXITPARAM Structure](#)

EXIT function's library interface

Library functions are used within an **EXIT** function using information contained in the EXITPARAM structure. This method provides great flexibility for data passed to and from a map. For example, the map can pass binary data containing nulls, and there is no limitation on the length of the returned data. The method also allows functions to report additional information by providing a return code and error message.

Using the EXITPARAM Structure

Function prototype

The function to be executed must be a function in a library with the following prototype:

```
void MyFunc (LPEXITPARAM lpep);
```

Definition of the EXITPARAM structure

The definition of the EXITPARAM structure is as follows:

```
struct tagExitParamStruct
{
    DWORD      dwSize;
    DWORD      dwToLen;
    DWORD      dwFromLen;
    DWORD      dwMapInstance;
    void FAR *  lpv;
    LPSTR      lpszCmdLine;
    BYTE HUGE * lpDataToApp;
    BYTE HUGE * lpDataFromApp;
    UINT       uRetryCount;
```

```

    UINT          uRetryInterval;
    BOOL          bRollback;
    BOOL          bCleanup;
    int           nReturn;
    char          szErrMsg[100];
    char          szFile[260];
    void FAR *   lpMapHandle;
    void FAR *   lpInternal;
    void FAR *   lpCmdStruct;
    void FAR *   lpAdaptParms;
    void FAR *   lpContext;
    void FAR *   lpWildcard;
    void FAR *   lpfnMS;
    void FAR *   lpMS;
    DWORD         dwWildcardSize;
    LPSTR         lpszMapDirectory;
    WORD          wCardNum;
    WORD          wCleanupAction;
    WORD          wScope;
    WORD          uUnitSize;
    BOOL          bBurst;
    BOOL          bFromRule;
    BOOL          bSource;
    DWORD         dwRecords;
};

typedef struct tagExitParamStruct EXITPARAM;
typedef struct tagExitParamStruct FAR * LPEXITPARAM;

```

The engine environment sets up the EXITPARAM structure and the function should fill in the result. The engine will allocate and free the memory associated with **lpDataToApp**. The function must allocate the memory for **lpDataFromApp**. For all Windows operating systems, use the Windows macro **GlobalAllocPtr** defined in **windowsx.h**. For all other operating systems, use the C-runtime **malloc** function. The engine will free this memory.

Table 1. Components of EXITPARAM as used with the EXIT function.

Component	Used as	Usage
dwSize	Input	Size (in bytes) of EXITPARAM to assure correct compatibility
dwToLen	Input	Length (size in bytes) of lpDataToApp
dwFromLen	Output	Length (size in bytes) of lpDataFromApp
dwMapInstance		Not used
lpv		Not used
LpszCmdLine		Not used
lpDataToApp	Input	Data sent to the function. This is the third parameter of the EXIT function
lpDataFromApp	Output	Data sent back to the server from the function
uRetryCount		Not used
uRetryInterval		Not used
bRollback		Not Used
bCleanup		Not used
nReturn	Output	Return code based on the outcome of the function
szErrMsg	Output	String message based on nReturn
szFile		Not used
lpInternal		Not used
lpCmdStruct		Not used
lpAdaptParms		Not used
lpContext		Not used
lpWildcard		Not used
dwWildcardSize		Not used
lpszMapDirectory		Not used
wCardNum		Not used
wCleanupAction		Not used
wScope		Not used
uUnitSize		Not used
bBurst		Not used
lpfnMS		Not used.
lpMS		Not used
dwRecords		Not used

There is no interaction with the **nReturn** or **szErrMsg** fields. However, this might change in a future release.

GET

You can use the **GET** function to retrieve data using one of the source adapters, such as a messaging system, a database, a file, and so forth, within the course of your map.

When the **GET** function is used in a map, the default **OnSuccess** action is adapter specific. The default **OnFailure** action is to rollback any changes made during map processing. The default **Scope** will be integral unless the map is defined to run in bursts (which is the case when one or more inputs have the **FetchAs** property set to **Burst**).

GET can also be used for adapters that support request and reply, such as Socket.

Syntax:

GET (single-text-expression , single-text-expression
[, single-text-expression])

Meaning:

GET (adapter_alias, adapter_commands
[, data_request_to_send_to_adapter])

Returns:

A single character text item

GET returns the data that is returned by the source adapter. The adapter identified by *adapter_alias* is called using the specified *adapter_commands* and passing *data_request_to_send_to_adapter* as data to the adapter. See the Resource Adapters documentation.

You cannot use the **GET** function to retrieve a file that was built by an earlier output card within the same map. Instead, map the output of the earlier card to the later card.

To identify whether the function was successful, you can generally use the **VALID** function with the **GET** function. However, for certain adapters such as E-mail and FTP, the success state is not known until after map completion and using the **VALID** function in such cases might consistently return "success".

Examples

- Reply (s) = **GET** ("SOCKET", "-HOST localhost -PORT 8015 -EOF -LSN 10 -T", **PACKAGE** (Request Object..:Input))

In this example, the **GET** function would connect to the socket-based server on localhost using port 8015. The Socket Adapter would submit the "Input" data to the server, and then wait up to 10 seconds for a reply to that message.

- Acct# = **GET** ("DB", "-dbtype ORACLE -connect shasta -user rjc -pw vm70" + "SELECT Acct# FROM CustMaster WHERE CustID = " + SenderID Field:Identification Segment:Msg + "")

In this example, the **GET** function could be used instead of a **DBLOOKUP** or **DBQUERY** function to retrieve data from a database from within in a map rule.

Related functions

- FAIL**
- LASTERRORCODE**
- LASTERRORMSG**
- PUT**
- VALID**

JEXIT

The JEXIT function provides a means to interface with a method in an external Java class. Using this function, you can manipulate simple primitive types, simple objects, and complex objects.

There are two required text parameters for a JEXIT function. The first text parameter is the class name or key, depending on which meaning you use. The second text parameter is the method name. There are two optional text parameters for the data.

The first use of the JEXIT function creates an object that is based on the specified class name. Then, it invokes the named Java method, with any specified parameters, on that object. After the Java object's method is invoked, then JEXIT automatically destroys the Java object. If multiple invocations on the same Java object are required, then the predefined "<init>" function can be used in the second parameter of Meaning 1. The returned key from the "<init>" function now references a specific Java object and can then be used multiple times as the first parameter in Meaning 2 function calls.

The second use of the JEXIT function invokes the named Java method, with any specified parameters, on a specific Java object. The method is invoked by using the specified key. This method allows Java objects to be manipulated throughout the map's execution cycle. These Java objects are stored in the IBM Transformation Extender object pool and are referenced during map execution by using the specified key. The specified key can be returned from either a JEXIT meaning 1 or meaning 2 invocation, particularly when a non-scalar, complex Java object must be returned.

JEXIT is not supported on CICS, Batch, or MVS.

Prerequisites

To use JEXIT, do the following configuration steps, depending on the IBM Transformation Extender product you are using.

For IBM Transformation Extender with Command Server or IBM Transformation Extender with Launcher, either add the Java classes to your system class path or add the JAR files to the config.yaml file.

Add the JAR files to the **jar** statements under the /runtime/External Jar Files path of the config.yaml file. For each JAR file, uncomment a **jar** statement and add the fully qualified path and JAR file name, including the file extension. If a **jar** statement is already set to a value, add the JAR file to the next **jar** statement in the sequence. As an example, if the **jar1** statement is set to a value, and the **jar2** statement is not set to a value, add your JAR file to the **jar2** statement.

For IBM Transformation Extender for IBM Integration Bus, on Windows operating systems, copy the JAR file to the classes folder in the path where your version of IBM Integration Bus is installed. As an example, copy the JAR file to %MQSI_WORKPATH%\shared-classes. On UNIX operating systems, copy the JAR file to the \$MQSI_WORKPATH/shared-classes.

If you are using Meaning 1, a public default constructor must be declared in the external Java class. If the default constructor is not available or needed, you can use the "<init>" method to instantiate the Java object, but only when there is a class constructor that is defined with one or two string-type parameters.

Syntax

```
JEXIT ("single-text-expression" , "single-text-expression" ,  
["single-text-expression"] , ["single-text-expression"])
```

Returns

A single text item.

Meaning 1

```
JEXIT (class_name, method_name,  
[input_to_the_function_or_key], [input_to_the_function_or_key])
```

Using this approach, you provide a Java class and method name, and two optional inputs to the function.

When JEXIT is used in a map rule, at run time this function instantiates a Java virtual machine (JVM). The JVM reads your .class or .jar file and creates an object that is based on the defined Java class. The method, along with any specified parameters, is then invoked on the object.

When the returned object is a complex object, a unique identifier (key) is created that can be used to reference the object in the object pool.

All JEXIT parameters are text strings. Class name and method name are the two required parameters. The two optional parameters are generally input data. At run time, the strings are converted to UTF-8 and passed to the Java method, and the output from Java method is a UTF-8 text string.

The following simple Java class objects are internally converted to strings and returned to the map:

```
java.lang.Void  
java.lang.Short  
java.lang.Number  
java.lang.Long  
java.lang.Integer  
java.lang.Float  
java.lang.Double  
java.lang.Byte  
java.lang.Boolean  
java.lang.String  
java.lang.Character
```

Meaning 2

```
JEXIT (key, method_name, [input_to_the_function_or_key],  
[input_to_the_function_or_key])
```

The second use of JEXIT is similar to Meaning 1. The difference is that it assumes that you already have a key that references an object in the object pool. In this scenario, you use JEXIT to invoke a method on an object by using the key as the first required parameter instead of the class name.

Whenever the JEXIT function returns a key, the associated Java object must be deleted before the map completes processing. The object can be deleted by using the predefined JEXIT "<destroy>" method. Failure to invoke the "<destroy>" method results in a JVM memory leak. To prevent the JVM heap from getting fragmented, exhausted, or both, make sure that all object references returned from the JEXIT function are deleted. To delete those object references, use the JEXIT function in the following way:

```
JEXIT("key", "<destroy>")
```

Troubleshooting

To retrieve error messages for JEXIT, you can use the VALID and FAIL functions in your map rule. For example:

```
=VALID(JEXIT("key", "method_name", "data"), FAIL(LASTERROREMSG( ) ) )
```

- [JEXIT Examples](#)

JEXIT Examples

The JEXIT examples are based on the following defined Java classes.

```
class :  
package com.ibm.websphere.dtx.test;  
public class TestJExit {  
    Person person = new Person();  
    public TestJExit() {}  
    public String toUpper(String test) {  
        return test.toUpperCase();  
    }  
    public String concat(String test1, String test2) {  
        return test1 + test2;  
    }  
    public Integer toNumber(String test1) {  
        return new Integer(test1);  
    }  
    public Double toDouble(String test1) {  
        return new Double(test1);  
    }  
    public void toNothing() {}  
    public int returnInt() {  
        return 9;  
    }  
    public byte returnByte() {  
        return 8;  
    }
```

```

        public Person getPerson() {
            return person;
        }

    class :
package com.ibm.websphere.dtx.test;
public class Person{
    private String ext = new String("DTX");
    private String name = new String("Websphere");
    public Person() {
    }
    public String getName() {
        return name;
    }
    public String getExt() {
        return ext;
    }
}

```

Simple primitive types and simple objects

- JEXIT ("com.ibm.websphere.dtx.test.TestJExit", "toUpperCase", "IBM")

Returns IBM
- JEXIT ("com.ibm.websphere.dtx.test.TestJExit", "Concat", "IBM Transformation", "Extender")

Returns IBM Transformation Extender
- JEXIT ("com.ibm.websphere.dtx.test.TestJExit", "toDouble", "12")

Returns 12.0
- JEXIT ("com.ibm.websphere.dtx.test.TestJExit", "toNumber", "12")

Returns 12
- JEXIT ("com.ibm.websphere.dtx.test.TestJExit", "returnint")

Returns 9
- JEXIT ("com.ibm.websphere.dtx.test.TestJExit", "returnbyte")

Returns 8

Complex objects

When the function is returning a complex object, the returning object is added to the object pool and the key is returned to the map rule.

- JEXIT ("com.ibm.websphere.dtx.test.TestJExit", "getPerson")

Returns a key (as a string)
- JEXIT ("key", "getName")

Returns Websphere
- JEXIT ("key", "<destroy>")

Deletes the Person object from the JVM memory. The "**<destroy>**" method must be invoked before the map completes processing.

PUT

The **PUT** function passes data to a target adapter.

Use **PUT** to route data within your map using one of the target adapters such as to a messaging system, a database, a file, and so forth.

When the **PUT** function is used in a map, the default **OnSuccess** action is adapter specific. The default **OnFailure** action is to rollback any changes made during map processing. The default **Scope** will be integral unless the map is defined to run in bursts (which is the case when one or more inputs have the **FetchAs** property set to **Burst**).

Syntax:

PUT (single-text-expression, single-text-expression, single-text-expression)

Meaning:

PUT (adapter_alias, adapter_commands, data_to_send_to_adapter)

Returns:

"None"

The adapter identified by *adapter_alias* is called using the specified *adapter_commands* and passing *data_to_send_to_adapter* as data to the adapter.

The **PUT** function is not processed if the third argument is zero-sized data or NONE.

To identify whether the function was successful, you can generally use the **VALID** function with the **PUT** function. However, for certain adapters such as E-mail and FTP, the success state is not known until after map completion and using the **VALID** function in such cases might consistently return "success".

Examples

This example illustrates a mapping situation in which a set of messages is being produced in the output (output card #1). However, the ultimate target for these messages is a message queue and the messages need to be placed on the queue one at a time.

To accomplish this, the first output card builds the set of messages and its **Target** is defined as **Sink**. So, this set of messages is constructed in memory, but is discarded after the map completes. A second output card, uses the following **PUT** function to put each output message (from output card# 1), individually, on the output queue.

Message(s) = **PUT** ("MQS", "-QM myqueuemgr -QN chips_queue -T", **PACKAGE** (PaymentMessage:CHIPS_Payment_Message))

- The first argument, MQS, identifies that the data is to be routed using the IBM WebSphere MQ (server) adapter.
- The second argument, "-QM myqueuemgr -QN chips_queue -T", provides the adapter commands needed by the adapter to put the message on the queue.
- The third argument, **PACKAGE**, passes the data that is to be sent as the body of the message.

Related functions

- **FAIL**
 - **GET**
 - **LASTERRORCODE**
 - **LASTERRORMSG**
 - **VALID**
-

RUN

The **RUN** function allows you to execute another compiled map from a component or map rule.

You can use **RUN** to dynamically name source or destination files, or both, or to dynamically pass data to a map. You can also use the **RUN** function to split the output data into separate files based on some value in the input.

Syntax:

RUN (single-text-expression [, single-text-expression])

Meaning:

RUN (map_to_run [, command_option_list])

Returns:

A single text item

The first argument, *map_to_run*, is an expression identifying the name of the compiled map (.mmc) to be run.

The *command_option_list* argument is an optional argument that you can use to specify execution commands applicable to the map to be run. *Command_option_list* is a text item containing a series of execution commands separated by a space. Any execution command can be used as part of the *command_option_list* argument. For example, you can send data to another map by using the echo command option (-IE).

See the Execution Commands documentation for a list of command options.

The result of the **RUN** function depends on the command options in *command_option_list*.

Echo command option

- If you use the Echo command option for an output card, the data from that card will be passed back as a text item to the object in the map from which it was run.
- If you use the Echo command option for more than one output card, the data from all echoed cards will be concatenated together and passed back as a text-item to the object in the map from which it was run.
- If you do not use the Echo command option, the return code indicating the status of the map that was run will be passed back to the object in the map from which it was run.

Examples

- **RUN** ("MyMap", "-IE1s502 " + Invoice:File + "-OF1 install_dir\\" + CustomerID:Invoice)

This example runs the **MyMap** map, sending 502-byte fixed-size **Invoice** data as the data source for input card **1**, overriding the filename of output card **1** based on **Customer** data and returns the map return code as the result.

- **RUN** ("GetDbOpt" , " ")

This example runs the **GetDbOpt** map (with no command options specified) and returns the map return code as the result.

- **RUN** ("DoOneSet" , "-A -GR" +
ECHOIN(1 , Set:InFile) +

" -OF1 OUT_SET." + **NUMBERTOTEXT** (INDEX (\$)) +
" -OE2")

This example runs the **DoOneSet** map. Command options include the following choices:

- "-A -GR"

The **DoOneSet** map will produce no **Audit Log** and restrictions will be ignored.

- **ECHOIN**(1 , Set:InFile)

The **ECHOIN** function creates the -IE command option for echoing the data represented by Set:InFile to input **1** of the RUN map.

- " -OF1 OUT_SET." + **NUMBERTOTEXT** (INDEX (\$))

The output file for card **1** of the **DoOneSet** map will be called "OUT_SET." plus a sequence number based on the index of the Set in **InFile**. For example, the first output set will be OUT_SET.1, and so forth.

- " -OE2"

Using the output echo command option, the data built for output card **2** of the **DoOneSet** map will be returned as the result of the **RUN** function.

An alternative to using the **ECHOIN** function shown above is using the long version. For example, replace **ECHOIN**(1 , Set:InFile) with:

" -IE1S" + **NUMBERTOTEXT** (**SIZE** (**Set:InFile**) + " " + **TEXT** (**Set:InFile**)

Using the Echo input command option (**-IE**) with the sizing method (**Sn**) or the **ECHOIN** function, one **Set** object of **InFile** is passed to input card **1** for the **DoOneSet** map.

Related functions

- [DDEQUERY](#)
-

RENAMEFILE

The **RENAMEFILE** function renames the source file to that of the destination file. If the operation succeeds, the returned value is 0; if it fails, then an operating system-specific error number is returned.

Syntax:

RENAMEFILE (*single-text-expression*, *single-text-expression*, *single-text-expression*, *single-text-expression*)

Meaning:

RENAMEFILE (source, destination, number_of_retries (max 100), pause_between_retries_in_ms (max 100000))

Returns:

A single number

Related reference

- [COPYFILE](#)
-

HARDLINKFILE

A hard link points or references to a specific space on the hard drive. Multiple files can be hard linked to the same space in the hard drive. If some data is changed on one of the hard linked files, all the other files reflect that change. The **HARDLINKFILE** function returns the hard link file's command status. On successful execution of this function: 0 is returned; on failure: an operating system-specific error code is returned.

Syntax:

HARDLINKFILE (*single-text-expression*, *single-text-expression*, *single-text-expression*, *single-text-expression*)

Meaning:

HARDLINKFILE [oldpath, newpath, number_of_retries (max 100), pause_between_retries_in_ms (max 100000)]

Returns:

A single number

Related reference

- [SOFTLINKFILE](#)
-

SOFTLINKFILE

A soft link points to a specific file, which in turn, references to a particular location on the hard drive. The soft link, also known as symbolic link is a second file that exists independently of its target. The **SOFTLINKFILE** function returns the soft (symbolic) link file's command status. On successful execution of this function: 0 is returned; on failure: an operating system-specific error code is returned.

Syntax:

SOFTLINKFILE (*single-text-expression*, *single-text-expression*, *single-text-expression*, *single-text-expression*)

Meaning:

SOFTLINKFILE [(oldpath, newpath, number_of_retries (max 100), pause_between_retries_in_ms (max 100000))]

Returns:

A single number

Related reference

- [HARDLINKFILE](#)
-

COPYFILE

The **COPYFILE** function copies the source file to the destination file. It returns the copy file command status. On successful execution of this function: 0 is returned; on failure: an operating system-specific error code is returned.

Syntax:

COPYFILE (*single-text-expression*, *single-text-expression*, *single-text-expression*, *single-text-expression*)

Meaning:

COPYFILE [source, destination, number_of_retries (max 100), pause_between_retries_in_ms (max 100000)]
Returns:
A single number

Related reference

- [RENAMEFILE](#)
-

SLEEP

The **SLEEP** function suspends the execution of the current map until the time-out interval elapses. The time-out or sleep interval is represented in milliseconds. On successful execution of this function: 0 is returned; on failure: -1 is returned.

Syntax:

SLEEP(*single-number-expression*)

Meaning:

SLEEP (the sleep time is in milliseconds)

Returns:

A single number

Flow functions

Use Flow functions to specify a value for each of the flow variables defined for a flow, and additionally, to alter an already specified value for a flow variable.

Flow variables act as an alternative means to pass information throughout a flow instance without requiring the use of data links.

Flow variables are name value pairs that persist until the flow instance completes execution. Flow variables are case-sensitive. Flow variables are also instance-specific. Instance-specific means that no two flow-instances that are running simultaneously can share the same set of variables.

Flow functions return the output in UTF-8 character encoding format. The flow variable name and value are converted to UTF-8 character encoding before a flow function is executed.

- [DECVARIABLE](#)
 - [DELETEVARIABLE](#)
 - [GETVARIABLE](#)
 - [INCVARIABLE](#)
 - [RESOLVEVARIABLE](#)
 - [SETVARIABLE](#)
-

DECVARIABLE

The **DECVARIABLE** function decrements the numeric value a flow variable.

This function defines a new flow variable if one does not exist already and sets the specified decrement value as the initial value for the new flow variable. If the decrement value is not specified, it defaults to 1, and returns the set value in text format.

If the flow variable exists already and holds a numeric value, this function decrements by the specified increment value. If the decrement value is not specified, it decrements the numeric value associated to a flow variable by 1 and returns the set value in text format to the map. The value associated to a flow variable is non-numeric, then this function skips performing decrement operation on the value of the flow variable, leaves the non-numeric value intact.

Syntax:

DECVARIABLE (*single-text-expression* [, *single-text-expression*])

Meaning:

DECVARIABLE (*variable_name* [, *decrement_by*])

Returns:

A single text item.

Examples

- **flowlib DECVARIABLE ("interest.rate")**
Decrements the value of flow variable, interest.rate, by 1 and returns the decremented value in text format. If the flow variable, interest.rate, does not exist already, creates one by that name, assigns 1 and returns "1".
- **flowlib DECVARIABLE "interest.rate", 10**
Decrements the value of flow variable, interest.rate, by 10 and returns the decremented value in text format. If the flow variable, interest.rate, does not exist already, creates one by that name, assigns 10 and returns "10".
- **flowlib DECVARIABLE ("city")**
Returns empty value, if the flow variable, city, holds a non-numeric text value as 'New York'. Decrement operation is not performed and retains the value of the flow variable intact as 'New York'. If the flow variable, city, does not exist already, then a new flow variable, city, is created and set with value 1, returns "1" to the map.

DELETEVARIABLE

The **DELETEVARIABLE**

This function deletes a flow variable defined for a flow. Special (reserved or internal) flow variables cannot be deleted with this function. All flow variables, custom and special, are deleted by default when flow instance finishes its execution. Use this function when a flow variable must be removed from the visible scope of nodes in a flow execution. A flow variable has to be deleted as setting empty value does not affect the value or existence of the flow variable.

Syntax:

DELETEVARIABLE (single-text-expression)

Meaning:

DELETEVARIABLE (variable_name)

Returns:

NONE

Examples

- `flowlib DELETEVARIABLE ("interest.rate")`
Deletes flow variable, interest.rate, and returns empty (NONE)

GETVARIABLE

The **GETVARIABLE** function retrieves the value associated to a flow variable.

This function returns an empty value if the flow variable has not been defined, or if it has been deleted prior to making this function invocation. Otherwise, it returns the associated value of the flow variable to the map in text format. The value associated to a special (reserved or internal) flow variable can be retrieved with this function for further transformation purposes.

Syntax:

GETVARIABLE GETVARIABLE (single-text-expression)

Meaning:

GETVARIABLE GETVARIABLE (variable_name)

Returns:

A single text item

Examples

- `flowlib GETVARIABLE "interest.rate", "10"`
Returns the current value of custom flow variable, interest.rate, to the map.
- `flowlib GETVARIABLE(" _FLOWINSTANCE_ ")`
Special flow variables start and end with underscore. Returns the value of the special (reserved or internal) flow variable, _FLOWINSTANCE_ to the map.

INCVARIABLE

The **INCVARIABLE** function increments the numeric value for a flow variable.

This function defines a new flow variable if it does not exist already and sets the specified increment value as the initial value for the new flow variable. If the increment value is not specified, it defaults to 1, and returns the set value in text format.

If the flow variable exists already and holds a numeric value, this function increments by the specified increment value. If the increment value is not specified, it increments the numeric value associated to a flow variable by 1, and returns the set value in text format to the map. The value associated to a flow variable is non-numeric, then this function skips performing increment operation on the value of the flow variable, leaves the non-numeric value intact.

Syntax:

INCVARIABLE (single-text-expression [, single-text-expression])

Meaning:

INCVARIABLE (variable_name [, increment_by])

Returns:

A single text item

Examples

- `flowlib >INCVARIABLE ("interest.rate")`
Increments the value of flow variable, interest.rate, by 1 and returns the incremented value in text format. If the flow variable, interest.rate, does not exist already, creates one by that name, assigns 1 and returns "1".
- `flowlib >INCVARIABLE ("interest.rate", 10)`
Increments the value of flow variable, interest.rate, by 10 and returns the incremented value in text format. If the flow variable, interest.rate, does not exist already, creates one by that name, assigns 10 and returns "10".
- `flowlib >INCVARIABLE ("city")`

Returns empty value, if the flow variable, city, holds a non-numeric text value as 'New York'. Increment operation is not performed and retains the value of the flow variable intact as 'New York'. If the flow variable, city, does not exist already, then a new flow variable, city, is created and set with value 1, returns "1" to the map.

RESOLVEVARIABLE

The **RESOLVEVARIABLE** function resolves all flow variables in a given text.

A flow variable to resolve in the text is specified in the format %variable_name%. The value associated with a flow variable is a text expression that is substituted for the variable name in a given text, if defined for the flow.

A text input can have more than one flow variable to resolve, and each flow variable can occur any number of times within the text. Any variable name that cannot be resolved is returned unchanged in the text. Special flow variables (reserved or internal) are also resolved for further transformation purposes.

Syntax:

RESOLVEVARIABLE (single-text-expression)

Meaning:

RESOLVEVARIABLE (text_to_resolve)

Returns:

A single text expression.

Examples

- `flowlib->RESOLVEVARIABLE ("greeting% World!")`

If the greeting flow variable has been previously defined as 'Hello', then the text is resolved and it is returned as 'Hello World!'. If the greeting flow variable is not defined, then the function returns '%greeting% World!'.

- `flowlib->RESOLVEVARIABLE ("Flow instance number is %_FLOWINSTANCE_% and UUID is %_FLOWUUID_%)`

Each flow instance being run is associated with an instance number and UUID. Special variables `_FLOWINSTANCE_` and `_FLOWUUID_` are resolved to their values and resolved text is returned.

For example, if `_FLOWINSTANCE_` is 12345 and `_FLOWUUID_` is abcxyz1234ABCxyz, the resolved text is as follows:

'Flow instance number is 12345 and UUID is abcxyz1234ABCxyz.'

- `flowlib->RESOLVEVARIABLE ("A total of %recordcount% X12 transactions for ANSI version %version% have been processed")`

If recordcount and version flow variables have been defined as 100 and 4050 respectively, then the resolved text returned is as follows:

'A total of 100 X 12 transactions for ANSI version 4050 have been processed'.

SETVARIABLE

The **SETVARIABLE** function sets a new, or updates an existing flow variable. This function specifies the name of a flow variable if one does not already exist. Otherwise, it updates the value of an existing flow variable with the new value.

The value associated to a flow variable is a text expression and returns the new value to the map. Empty value skips creating or updating of a flow variable. Flow variables that are considered special (reserved or internal) cannot be set with this function.

Syntax:

SETVARIABLE (single-text-expression, single-text-expression)

Meaning:

SETVARIABLE (variable_name, variable_value)

Returns:

A single text item

Examples

- `flowlib SETVARIABLE "interest.rate", "10")`

Sets flow variable, interest.rate, with a value 10. Creates a new flow variable if interest.rate does not exist already, otherwise updates and returns the new value 10

- `flowlib SETVARIABLE ("message.id", "IDXYZ123")`

Sets or updates flow variable, message.id, with IDXYZ123 and returns IDXYZ123

Inspection functions

- **ABSENT**
- **CONTAINSERRORS**
- **DELETEFILE**

The **DELETEFILE** function returns the status of a file deletion command as a single number. **DELETEFILE** returns 0 for successful file deletion or the appropriate error code if file deletion was not successful.

- **ISALPHA**
- **ISERROR**
- **ISLOWER**
- **ISNUMBER**

- [ISUPPER](#)
 - [MEMBER](#)
 - [NOT](#)
 - [OFFSET](#)
 - [OR](#)
 - [PARTITION](#)
 - [PRESENT](#)
 - [SIZE](#)
 - [TESTOFF](#)
 - [TESTON](#)
 - [VALID](#)
-

ABSENT

The **ABSENT** function tests for the absence of an object.

Syntax:

ABSENT (single-object-expression)

Meaning:

ABSENT (object_to_test)

Returns:

True or false

ABSENT returns "true" if the object_to_test evaluates to "none". If the object_to_test does not evaluate to "none", the function returns "false".

Examples

- You can use this function to map an object only if another object is absent. For example, you might want to map **BillTo** information to the **ShipTo** fields if the **ShipToName** is absent.
- **ABSENT** (AreaCode:Phone)
This example evaluates to "true" when **AreaCode:Phone** evaluates to "none" or evaluates to "false" when **AreaCode:Phone** does not evaluate to "none".

Related Functions

- [PRESENT](#)
-

CONTAINSERRORS

The **CONTAINSERRORS** function tests a valid object to see whether it contains any objects in error.

Syntax:

CONTAINSERRORS (single-object-expression)

Meaning:

CONTAINSERRORS (object_to_test)

Returns:

True or false

The **CONTAINSERRORS** function returns "true" if any object contained in *object_to_test* is in error; it returns "false" if the *object_to_test* is completely valid.

The input object, itself, is a *valid* input object. This function does *not* evaluate for invalid objects. Therefore, if you have map rule:

CONTAINSERRORS (Invoice:InputFile)

and **Invoice[1]** is valid, **Invoice[2]** is invalid, and **Invoice[3]** is valid, then **CONTAINSERRORS** evaluates only twice-once for each *valid* instance of **Invoice**.

Examples

- `Msg (s) = IF (CONTAINSERRORS (Msg:MailBag) &`
`Type:Msg:MailBag = "PRIORITY" ,`
`Msg:MailBag ,`
`"none")`

In this example, if **Msg:MailBag** contains any object in error and **Type:Msg:MailBag** has a value of "PRIORITY", **Msg:MailBag** is mapped; otherwise, "none" is returned for this occurrence of **Msg**. This map rule returns all valid messages (**Msg**) that contain errors with a **Type** of "PRIORITY".

Related functions

- [ISERROR](#)
 - [REFORMAT](#)
-

DELETEFILE

The **DELETEFILE** function returns the status of a file deletion command as a single number. **DELETEFILE** returns 0 for successful file deletion or the appropriate error code if file deletion was not successful.

Syntax:

resourcelib->**DELETEFILE** (single-text-expression, single-text-expression, single-text-expression)

Meaning:

DELETEFILE (*filename, number_of_retries, pause_between_retries*)

Returns:

A single number

number_of_retries

The number of times the **DELETEFILE** function is to check for the deleted file. The maximum is 100 retries.

pause_between_retries

The amount of time, in milliseconds, that the function waits between checking for the deleted file. The maximum is 100,000.

Example

```
(resourcelib->DELETEFILE (GETDIRECTORY () + "myfile1.txt", "10", "1000"))
```

ISALPHA

You can use **ISALPHA** when you need to know whether a text string is all alphabetic characters.

Syntax:

ISALPHA (single-text-expression)

Meaning:

ISALPHA (*text_to_test*)

Returns:

This function evaluates to a Boolean "true" or "false" and should only be used as a conditional expression within a logical function.

The **ISALPHA** function tests a text object to see if it contains all alphabetic characters.

If *text_to_test* contains only alphabetic characters (for example, A-Z and a-z), **ISALPHA** evaluates to "true".

If *text_to_test* contains other than just alphabetic characters, **ISALPHA** returns "false".

Examples

- **IF(ISALPHA ("AnywhereUSA"))**
Returns "true"
- **IF(ISALPHA ("Anywhere USA"))**
Returns "false"
- **IF(ISALPHA ("Mr. Brown"))**
Returns "false"

Related functions

• ISLOWER	• LEAVEALPHANUM
• ISNUMBER	• LEAVENUM
• ISUPPER	• LEAVEPRINT
• LEAVEALPHA	

ISERROR

The **ISERROR** function tests an object to see if it is in error

You can use **ISERROR** to output your data in exactly the same order as it occurs in your input, both the valid data and the data in error. You can also use **ISERROR** to produce error messages for bad data in the same file in which you map your good data.

Syntax:

ISERROR (single-object-name)

Meaning:

ISERROR (*object_to_test*)

Returns:

"True" or "false"

ISERROR returns "true" when *object_to_test* is in error and returns "false" when *object_to_test* is completely valid.

Examples

- InfoRec (s) = IF (**ISERROR** (Record:SomeFile), "Bad --> " + **REJECT** (Record:SomeFile), "Ok --> " + TEXT (Record:SomeFile))

In this example, **ISERROR** is used to produce a report for *all* Record objects in **SomeFile**. If the record is in error, the **InfoRec** will have the text Bad --> followed by the data from the input **Record**. If the record is valid, the **InfoRec** will have the text Ok --> followed by the data from the input **Record**, such as:

```
Ok --> SZ-68839,486 Upgrade Microprocessor,186.86,100,W200
Bad --> MK-19309,,369.43,417,W100
Ok --> KL-20349,PCMCIA Network Adaptor,174.82,29,N300
Ok --> WP-37679,AC Adaptor,39.48,245,E100
Bad --> IL-39890,8MB Memory PCMCIA,390.48,S100
```

Related functions

- CONTAINERRORS**

ISLOWER

The **ISLOWER** function tests a text object to see if it contains all lowercase alphabetic characters.

Syntax:

ISLOWER (series-text-expression)

Meaning:

ISLOWER (text_to_test)

Returns:

This function evaluates to a Boolean "true" or "false" and should only be used as a conditional expression within a logical function.

If *text_to_test* contains only lowercase alphabetic characters (for example, a-z), **ISLOWER** evaluates to "true".

If *text_to_test* contains other than lowercase alphabetic characters, **ISLOWER** returns "false".

Examples

- IF(**ISLOWER** ("company"))

Returns "true"
- IF(**ISLOWER** ("pots and pans"))

Returns "false"
- IF(**ISLOWER** ("Andrew"))

Returns "false"

Related functions

<ul style="list-style-type: none"> • ISALPHA 	<ul style="list-style-type: none"> • LEAVEALPHANUM
<ul style="list-style-type: none"> • ISNUMBER 	<ul style="list-style-type: none"> • LEAVENUM
<ul style="list-style-type: none"> • ISUPPER 	<ul style="list-style-type: none"> • LEAVEPRINT
<ul style="list-style-type: none"> • LEAVEALPHA 	

ISNUMBER

The **ISNUMBER** function tests a text object to determine whether it contains all numeric characters.

Syntax:

ISNUMBER (single-text-expression)

Meaning:

ISNUMBER (text_to_test)

Returns:

This function evaluates to a Boolean "true" or "false" and should only be used as a conditional expression within a logical function.

If *text_to_test* contains only digits (for example, 0-9), **ISNUMBER** evaluates to "true".

If *text_to_test* contains other than digits, **ISNUMBER** evaluates to "false".

Examples

- IF(**ISNUMBER** ("1"))

Returns "true"

Related functions

<ul style="list-style-type: none"> • ISALPHA 	<ul style="list-style-type: none"> • LEAVEALPHANUM
--	--

• ISLOWER	• LEAVENUM
• ISUPPER	• LEAVEPRINT
• LEAVEALPHA	

ISUPPER

The **ISUPPER** function tests a text object to determine whether it contains all uppercase alphabetic characters.

Syntax:

ISUPPER (series-text-expression)

Meaning:

ISUPPER (text_to_test)

Returns:

This function evaluates to a Boolean "true" or "false" and should only be used as a conditional expression within a logical function.

If *text_to_test* contains only uppercase alphabetic characters (for example, A-Z), ISUPPER evaluates to "true".

If *text_to_test* contains other than uppercase alphabetic characters, ISUPPER returns "false".

Examples

- **IF(ISUPPER ("BOMBAY"))**
Returns "true"
- **IF(ISUPPER ("CD-ROM"))**
Returns "false"
- **IF(ISUPPER ("Map Designer"))**
Returns "false"

Related functions

• ISALPHA	• LEAVEALPHANUM
• ISLOWER	• LEAVENUM
• ISNUMBER	• LEAVEPRINT
• LEAVEALPHA	

MEMBER

Use **MEMBER** when you need to know whether an object occurs within a series.

The **MEMBER** function searches a series, looking for a single specified object in the series. If any object in the series matches the specified object, **MEMBER** returns "true". If there is no match, **MEMBER** returns "false".

Syntax:

MEMBER (single-object-expression , series-object-expression)
MEMBER (single-object-expression , { literal, literal ... })

Meaning:

MEMBER (object_to_look_for , series_of_objects_to_look_at)

Returns:

"True" or "false"

MEMBER returns "true" if *object_to_look_for* matches one of the values in *series_of_objects_to_look_at*.

It returns "false" if *object_to_look_for* does not match at least one of the values in *series_of_objects_to_look_at*.

The two arguments, *object_to_look_for* and *series_of_objects_to_look_at*, must be objects of the same item interpretation or the same group type. For example, if *object_to_look_for* is a date/time item, *series_of_objects_to_look_at* must be a series of date/time items.

Examples

- **MEMBER** (EntityIDCode:Name, {"BT", "ST"})
This example tests whether **EntityIDCode** has one of a particular set of literal values.
- **MEMBER** (Store# , EntityIDCode:Name)
This example tests whether **Store#** has the same value as any **EntityIDCode:Name**.

Related functions

- **EXTRACT**
 - **LOOKUP**
-

NOT

Use the **NOT** function to test a condition and have it return the inverse of its "true" or "false" result. For example, you want the function to return "true" if the condition results in "false".

Syntax:

NOT (single-condition-expression)

Meaning:

NOT (condition_to_evaluate)

Returns:

"True" or "false"

NOT returns "true" if the condition evaluates to "false" and returns "false" if the condition evaluates to "true".

Examples

- **NOT** (Qty::InputFile = 0)

This example returns "false" if the **Qty** equals 0 (the condition is true) and returns "true" if **Qty** does not equal 0 (the condition is false).

- **IF (NOT (PRESENT (StartDate)), "Unknown", "none")**

This example returns "unknown" if **StartDate** is not present and returns "none" if **StartDate** is present.

Another way to test that an object is not present is to use the **ABSENT** function.

OFFSET

Use the **OFFSET** function when you need to know the position of a particular data object within its card object.

OFFSET returns an integer representing the offset of the specified object within the data.

Syntax:

OFFSET (single-object-expression)

Meaning:

OFFSET (object_whose_offset_is_needed)

Returns:

A single integer

OFFSET returns the offset, in bytes, of the specified object within its card object, beginning at offset 0. For an object that has an initiator, the offset will apply to the first byte of the data. For an object that is right-justified with pad characters, **OFFSET** will return the offset of the first byte of data.

The **OFFSET** function works the same for output objects as it does for input objects.

Examples

- **OFFSET** (Application:LoanData)

In this example, if the first character of the first occurrence of the object **Application** occurs 210 bytes from offset 0 within **LoanData**, the **OFFSET** function returns 210.

Related function

- **SIZE**
-

OR

Use the **OR** function to test whether one of a series of conditions is true.

OR evaluates a series of conditions and returns "true" if at least one evaluates to "true"; otherwise returns "false".

Syntax:

OR (series-condition-expression)

Meaning:

OR (conditions_to_evaluate)

Returns:

"True" or "false"

OR evaluates to "true" if *any* member of the argument evaluates to "true" and evaluates to "false" if *all* members of the argument evaluate to "false".

Examples

- Order(s)=IF (OR (Store:Table = Store#:Order:Input), Order:Input)
This example produces an **Order** if the **Store#** of an **Order** in Input matches any **Store** in **Table**.

Related function

- **ALL**

PARTITION

The **PARTITION** function checks to see if an occurrence of an object belongs to a certain partition. If the object is that partition, "true" is returned. Otherwise, "false" is returned.

Syntax:

PARTITION (single-object-expression, single-simple-object-name)

Meaning:

PARTITION (partitioned_object, simple_name_of_partition_to_check_for)

Returns:

"True" or "false"

PARTITION returns "true" if the data object of *partitioned_object* belongs to the partition represented by *simple_name_of_partition_to_check_for*. Otherwise, **PARTITION** returns "false".

Examples

- Assume that **Transaction** has been partitioned into three partitioned subtypes: **Invoice**, **Order**, and **Remittance**, as shown.
The following rule could be used to detect whether a given **Transaction** is an **Invoice**:

PARTITION (Transaction::Batch, Invoice)

If the **Transaction** is an **Invoice**, **PARTITION** returns "true". If the **Transaction** is not an **Invoice** (for example, it is an **Order** or a **Remittance**), **PARTITION** returns "false" for that **Transaction**.

Related function

- **GETPARTITIONNAME**

PRESENT

The **PRESENT** function tests for the presence of an object.

PRESENT is commonly used with the **IF** function in a map rule to provide conditional logic. For example, if the object is present, do this; otherwise, do something else.

Similarly, **PRESENT** is commonly used with the **WHEN** function in component rules to provide conditional validation logic.

Syntax:

PRESENT (single-object-expression)

Meaning:

PRESENT (object_to_look_for)

Returns:

"True" or "false"

PRESENT returns "true" if the input argument does *not* evaluate to "none"; the object is present. It returns "false" if the input argument evaluates to "none"; the object is not present.

Examples

- **PRESENT** (Trailer:File)
This example returns "true" if **Trailer** is present and returns "false" if **Trailer** is absent.
- **IF (PRESENT(MiddleInitial::Input), MiddleInitial::Input, "***")**
This example in a map rule maps **MiddleInitial** if it is present. If **MiddleInitial** is not present, three asterisks are mapped.
- **WHEN (PRESENT (AreaCode Field), PRESENT (PhoneNo Field))**
In this example, **PRESENT** is being used in conjunction with the **WHEN** function in a component rule to determine whether a particular object is valid.

Related functions

- **ABSENT**
- **IF**
- **WHEN**

SIZE

The **SIZE** function returns an integer representing the size of a specified object, exclusive of any pad characters.

Syntax:

SIZE (single-object-expression)

Meaning:

SIZE (object_whose_size_is_needed)

Returns:

A single integer

The **SIZE** function returns the size, in bytes, of *object_whose_size_is_needed*. The byte size returned does not include any pad characters that might be in the object, but does include separators and signs.

The size of a group is the size, beginning with the first character of the first component and ending with the last character of the last component, of the group. If the group has delimiters, the infix delimiters are included in the size.

Examples

- **SIZE** (Transaction)
Returns 8000 if the size of **Transaction** (without pad characters) is 8000 bytes

Using SIZE with NORMXML

If the **SIZE** function is used with the **NORMXML** function to determine the size of the specified object after **NORMXML** has removed the XML formatting from the input XML fragment, the **SIZE** operation calculates the size of the data in Unicode, since the **NORMXML** function converts the input data to Unicode before it removes the XML formatting. The returned size does not match the size that would have been calculated if the data remained in its original character set, unless the original character set was Unicode.

TESTOFF

The **TESTOFF** function tests a specified bit in a binary number item to see whether it is off.

Syntax:

TESTOFF (single-binary-number-expression, single-integer-expression)

Meaning:

TESTOFF (binary_number_to_test, bit_to_test)

Returns:

"True" or "false"

The value of *bit_to_test* specifies which bit of *binary_number_to_test* should be tested for the value 0. If *bit_to_test* has the value 1, it refers to the leftmost bit of *binary_number_to_test*.

The **TESTOFF** function returns "true" if the specified bit is off and returns "false" if the specified bit is on.

If *bit_to_test* is less than one or greater than the number of bits of *binary_number_to_test*, **TESTOFF** returns "false".

Examples

- **TESTOFF** (A, 16)
Assume **A** is the two-byte binary value of "1", which is all zeros except for bit **16**. The binary representation of the value in **A** is 0001.
This example returns "false".
- **TESTOFF** (A, 20)
Returns "false" because bit **20** does not exist.

TESTON

The **TESTON** function tests a specified bit in a binary number to see if it is on.

Syntax:

TESTON (single-binary-number-expression, single-integer-expression)

Meaning:

TESTON (binary_number_to_test, bit_to_test)

Returns:

"True" or "false"

The value of *bit_to_test* specifies the bit of *binary_number_to_test* to test for the value 1. If *bit_to_test* has the value 1, it refers to the leftmost bit of *binary_number_to_test*.

The function returns "true" if the specified bit is on; it has the value 1. It returns "false" if the specified bit is off; it has the value 0.

If *bit_to_test* is less than one or greater than the number of bits of *binary_number_to_test*, **TESTON** returns "false".

Examples

- **TESTON (A , 16)**
Assume **A** is the two-byte binary value of "1", which is all zeros except for bit **16**. The binary representation of the value in **A** is 0001.
This example returns "true".
 - **TESTON (A , 20)**
Returns "false" because bit **20** does not exist in a two-byte value.
-

VALID

You can use the **VALID** function to perform conditional processing based on whether an external interface function executes successfully.

VALID returns the result of the first argument if it is valid; otherwise, returns the second argument.

Syntax:

VALID (single-text-expressions , single-general-expression)

Meaning:

VALID (function_that_can_fail , return_value_if_function_fails)

Returns:

A single text expression

VALID returns the result of the evaluation of *function_that_can_fail* if it is valid. If the function fails, **VALID** returns *return_value_if_function_fails*.

The following functions can fail:

• DBLOOKUP	• GET
• DBQUERY	• PUT
• DDEQUERY	• RUN
• EXIT	

Examples

- **SomeObject = VALID (RUN (**
"mymap.mmc" , "-OF1 mydata.txt") ,
FAIL ("My RUN failed!")

If the **RUN** function returns an error return code, the **VALID** function returns **none**, the map aborts, and the message **My RUN failed!** is reported under Execution Summary in the execution audit log.

Related functions

- **DBLOOKUP**
 - **DBQUERY**
 - **DDEQUERY**
-

Logical functions

- [**ALL**](#)
 - [**BINDABS**](#)
 - [**EITHER**](#)
 - [**IF**](#)
 - [**ISALPHA**](#)
 - [**ISLOWER**](#)
 - [**ISNUMBER**](#)
 - [**ISUPPER**](#)
 - [**NOT**](#)
 - [**OR**](#)
 - [**WHEN**](#)
-

ALL

The **ALL** function evaluates a series of conditions and returns "true" if they all evaluate to "true"; otherwise returns "false".

Syntax:

ALL (series-condition-expression)

Meaning:

ALL (conditions_to_evaluate)
Returns:
True or false
The **ALL** function evaluates to "true" if *all* members of the input argument evaluate to "true"; it evaluates to "false" if *any* member of the input argument is "false".

Examples

- You can use **ALL** when you want to test whether all conditions of a series are true.
- PO(s)=IF (ALL(PO#:Line:Order = PO#:Line:Order[1]), Order, "none")
If each and every **PO#** matches the **PO#** of the first **Order**, **ALL** evaluates to "true". Otherwise, **ALL** evaluates to "false".

Related functions

- **NOT**
 - **OR**
-

BINDABS

Use this function in a component rule for the implementation of binding and iterative evaluation. The implementation of binding ensures that when there are multiple references to a component in a rule, all references are to the same instance of that component. The implementation of iterative evaluation ensures that the rule is evaluated using all possible combinations of type references.

Do not use this function unless you have a clear understanding of the behavior.

Syntax:
BINDABS (Boolean_expression)

Meaning:
BINDABS (component rule)
Returns:
A Boolean "true" or "false"

Note: It is not necessary to use the **BINDABS** function for iterative behavior with the following functions because these functions already iterate: COUNT, COUNTABS, INDEX, MEMBER, MAX, MIN, SUM, and SERIESTOTEXT.

How it works

When the evaluation starts and the **BINDABS** function is encountered at the beginning of the component rule, binding is enforced and the rule is iterated across all valid combinations of objects specified in the rule.

BINDABS must be positioned first in the component rule and the entire component rule must be enclosed. Any other use of this function will have no effect.

When using **BINDABS**, the evaluation order is the same as mapping. Rules that contain both a normal instance and a last instance for the first instance of the type are not evaluated.

Example

```
BINDABS (Qty:LineItem:Order
* Price:LineItem:Order = ExtendedPrice:LineItem:Order)
```

As a result of binding, the Qty, Price, and ExtendedPrice elements in this example are all in the same LineItem of the same Order. As a result of iteration, all line items of all orders are checked.

EITHER

The **EITHER** function returns the result of the first argument that does not evaluate to "none".

Syntax:
EITHER (single-general-expression { , single-general-expression })
Meaning:
EITHER (try_this { , if_none_try_this })
Returns:
A single object

The **EITHER** return methods work in the following ways:

Returns...
When...
try_this
try_this does not evaluate to "none"
if_none_try_this
try_this evaluates to "none"
the next *if_none_try_this*
the first *if_none_try_this* evaluates to "none"

Note: The Design Studio compiler does not validate the second argument or subsequent arguments against the type tree.

Examples

- **EITHER** (OrderDate Field , **CURRENTDATE** ())
If **OrderDate Field** does not evaluate to "none", it is returned. If **OrderDate Field** evaluates to "none", the current system date (using **CURRENTDATE**) is returned.
- **EITHER** (LOOKUP (PriorityCd:Msg , CustID:Msg = "93X") , 8)
This example returns the result of the LOOKUP if that result does not evaluate to "none"; otherwise, it returns the second argument of 8.
- **EITHER** (IF (SomeCode = "C" , CustomerID:Input) ,
IF (SomeCode = "S" , SupplierID:Input) ,
IF (SomeCode = "O" , Reference#:Input) ,
"#####")
If **SomeCode** is **C** and **CustomerID:Input** has a value, **EITHER** returns the value of **CustomerID**. Otherwise, if **SomeCode** is **S** and **CustomerID:Input** has a value, **EITHER** returns the value of **SupplierID**. Otherwise, if **SomeCode** is **O** and **Reference#:Input** has a value, **EITHER** returns the value of **Reference#**. If none of those conditions result in a value, **EITHER** returns "#####".
- You can use **EITHER** when you want a default value when an expression evaluates to "none" and the expression might cause common arguments to produce an unintended result. For example, use:
EITHER (LOOKUP (PriorityCd:Msg , CustID:Msg = "93X") , 8)
-instead of-
IF (PRESENT (LOOKUP (PriorityCd:Msg , CustID:Msg = "93X")) ,
LOOKUP (PriorityCd:Msg , CustID:Msg = "93X") , 8)

IF

You can use the **IF** function for conditional logic. For example, to return one of two objects depending on the evaluation of a condition.

IF evaluates a conditional expression, returning one value if true, another if false.

Syntax:

IF (single-condition-expression , single-general-expression
[, single-general-expression])

Meaning:

IF (test_this , result_if_true [, result_if_false])

Returns:

A single item or single group

If *test_this* evaluates to "true", the result is *result_if_true*.

Otherwise, if *test_this* evaluates to "false", the result is *result_if_false*. If no *result_if_false* is specified, this evaluates to "none".

The *result_if_true* and *result_if_false* arguments must correspond to the same item interpretation or the same group type or either can be "none". For example, if *result_if_true* evaluates to a number, *result_if_false* must also evaluate to a number. If *result_if_true* evaluates to a **LineItem** group, *result_if_false* must also evaluate to a **LineItem** group.

Examples

- **IF** (Quantity:LineItem:PO > 500 , "PRIORITY" , "REGULAR")
This example tests the **Quantity** value. If that **Quantity** is > 500, the **IF** function evaluates to the value PRIORITY. If that **Quantity** is <= 500, the **IF** function evaluates to the value, REGULAR.
- **IF** (Status:Order = "Special" , "SPCL")
This example tests the **Status** value. If the **Status** is Special, the **IF** function evaluates to the value, SPCL. Otherwise, it evaluates to "none". Notice that this rule is equivalent to
IF (Status:Order = "Special" , "SPCL" , "none")

ISALPHA

You can use **ISALPHA** when you need to know whether a text string is all alphabetic characters.

Syntax:

ISALPHA (single-text-expression)

Meaning:

ISALPHA (text_to_test)

Returns:

This function evaluates to a Boolean "true" or "false" and should only be used as a conditional expression within a logical function.

The **ISALPHA** function tests a text object to see if it contains all alphabetic characters.

If *text_to_test* contains only alphabetic characters (for example, A-Z and a-z), **ISALPHA** evaluates to "true".

If *text_to_test* contains other than just alphabetic characters, **ISALPHA** returns "false".

Examples

- **IF(ISALPHA ("AnywhereUSA"))**
Returns "true"
- **IF(ISALPHA ("Anywhere USA"))**
Returns "false"
- **IF(ISALPHA ("Mr. Brown"))**
Returns "false"

Related functions

• ISLOWER	• LEAVEALPHANUM
• ISNUMBER	• LEAVENUM
• ISUPPER	• LEAVEPRINT
• LEAVEALPHA	

ISLOWER

The **ISLOWER** function tests a text object to see if it contains all lowercase alphabetic characters.

Syntax:

ISLOWER (series-text-expression)

Meaning:

ISLOWER (text_to_test)

Returns:

This function evaluates to a Boolean "true" or "false" and should only be used as a conditional expression within a logical function.

If *text_to_test* contains only lowercase alphabetic characters (for example, a-z), **ISLOWER** evaluates to "true".

If *text_to_test* contains other than lowercase alphabetic characters, **ISLOWER** returns "false".

Examples

- **IF(ISLOWER ("company"))**
Returns "true"
- **IF(ISLOWER ("pots and pans"))**
Returns "false"
- **IF(ISLOWER ("Andrew"))**
Returns "false"

Related functions

• ISALPHA	• LEAVEALPHANUM
• ISNUMBER	• LEAVENUM
• ISUPPER	• LEAVEPRINT
• LEAVEALPHA	

ISNUMBER

The ISNUMBER function tests a text object to determine whether it contains all numeric characters.

Syntax:

ISNUMBER (single-text-expression)

Meaning:

ISNUMBER (text_to_test)

Returns:

This function evaluates to a Boolean "true" or "false" and should only be used as a conditional expression within a logical function.

If *text_to_test* contains only digits (for example, 0-9), **ISNUMBER** evaluates to "true".

If *text_to_test* contains other than digits, **ISNUMBER** evaluates to "false".

Examples

- **IF(ISNUMBER("1"))**
Returns "true"

Related functions

• ISALPHA	• LEAVEALPHANUM
• ISLOWER	• LEAVENUM
• ISUPPER	• LEAVEPRINT
• LEAVEALPHA	

ISUPPER

The **ISUPPER** function tests a text object to determine whether it contains all uppercase alphabetic characters.

Syntax:

ISUPPER (series-text-expression)

Meaning:

ISUPPER (text_to_test)

Returns:

This function evaluates to a Boolean "true" or "false" and should only be used as a conditional expression within a logical function.

If *text_to_test* contains only uppercase alphabetic characters (for example, A-Z), ISUPPER evaluates to "true".

If *text_to_test* contains other than uppercase alphabetic characters, ISUPPER returns "false".

Examples

- **IF(ISUPPER ("BOMBAY"))**
Returns "true"
- **IF(ISUPPER ("CD-ROM"))**
Returns "false"
- **IF(ISUPPER ("Map Designer"))**
Returns "false"

Related functions

• ISALPHA	• LEAVEALPHANUM
• ISLOWER	• LEAVENUM
• ISNUMBER	• LEAVEPRINT
• LEAVEALPHA	

NOT

Use the **NOT** function to test a condition and have it return the inverse of its "true" or "false" result. For example, you want the function to return "true" if the condition results in "false".

Syntax:

NOT (single-condition-expression)

Meaning:

NOT (condition_to_evaluate)

Returns:

"True" or "false"

NOT returns "true" if the condition evaluates to "false" and returns "false" if the condition evaluates to "true".

Examples

- **NOT (Qty::InputFile = 0)**
This example returns "false" if the **Qty** equals 0 (the condition is true) and returns "true" if **Qty** does not equal 0 (the condition is false).
- **IF (NOT (PRESENT (StartDate))), "Unknown", "none")**
This example returns "unknown" if **StartDate** is not present and returns "none" if **StartDate** is present.

Another way to test that an object is not present is to use the **ABSENT** function.

OR

Use the **OR** function to test whether one of a series of conditions is true.

OR evaluates a series of conditions and returns "true" if at least one evaluates to "true"; otherwise returns "false".

Syntax:

OR (series-condition-expression)

Meaning:

OR (conditions_to_evaluate)

Returns:

"True" or "false"

OR evaluates to "true" if *any* member of the argument evaluates to "true" and evaluates to "false" if *all* members of the argument evaluate to "false".

Examples

- Order(s)=IF (**OR** (Store:Table = Store#:Order:Input), Order:Input)
This example produces an **Order** if the **Store#** of an **Order** in Input matches any **Store** in **Table**.

Related function

- **ALL**

WHEN

You can use the **WHEN** function in a component rule to test for the validity of one object based on another object.

WHEN evaluates a condition. Then, based on that evaluation, evaluates another condition and returns "true" or "false".

Syntax:

WHEN (single-condition-expression , single-condition-expression [, single-condition-expression])

Meaning:

WHEN (condition1 , condition2 [, condition3])

Returns:

"True" or "false"

The following statements summarize how the **WHEN** function works:

- Returns "true" if *condition1* and *condition2* both evaluate to "true".
- Returns "true" if *condition1* evaluates to "false" and there is no *condition3* or if *condition1* evaluates to "false" and *condition3* evaluates to "true".
- Returns "false" if *condition1* evaluates to "true" and *condition2* evaluates to "false".
- Returns "false" if both *condition1* and *condition3* evaluate to "false".

Evaluation of the three arguments:

Condition 1	Condition 2	Condition 3	WHEN returns:
True	True		True
False			True
False		True	True
True	False		False
False		False	False

Examples

- **WHEN (ABSENT(CatalogueField), ABSENT (QuantityField))**
Returns "true" when **CatalogueField** and **QuantityField** are both absent
- **WHEN (PRESENT(ShipDate), PRESENT(InStock), PRESENT(BackOrderDate))**
Returns "false" when **ShipDate** and **BackOrderDate** are both absent

Lookup and reference functions

- **CHOOSE**
- **DBLOOKUP**
- **DBQUERY**
- **DDEQUERY**
- **EXTRACT**
- **GETANDSET**

- **[GETDIRECTORY](#)**
- **[GETITXUID](#)**
The **GETITXUID** function returns a universally unique identifier (UUID).
- **[GETLOCALE](#)**
The **GETLOCALE** function returns the locale setting of the computer.
- **[GETFILENAME](#)**
- **[GETPARTITIONNAME](#)**
- **[GETRESOURCEALIAS](#)**
The **GETRESOURCEALIAS** function returns the resource alias value specified in a Resource Registry resource name file (.mrn).
- **[GETRESOURCENAME](#)**
- **[GETTXINSTALLDIRECTORY](#)**
The **GETTXINSTALLDIRECTORY** function returns the IBM Transformation Extender product installation directory.
- **[GETDATADIRECTORY](#)**
The **GETDATADIRECTORY** function returns the data directory chosen by the user during installation.
- **[INDEX](#)**
- **[INDEXABS](#)**
- **[LASTERRORCODE](#)**
- **[LASTERRORMSG](#)**
- **[LOOKDOWN](#)**
- **[LOOKUP](#)**
- **[MEMBER](#)**
- **[SEARCHDOWN](#)**
- **[SEARCHUP](#)**
- **[SORTDOWN](#)**
- **[SORTDOWNBY](#)**
- **[SORTUP](#)**
- **[SORTUPBY](#)**
- **[UNIQUE](#)**
- **[GETENV](#)**
The **GETENV** function returns the value of the environment variable. If no variable exists, an empty string is returned.
- **[GETOSNAME](#)**
The **GETOSNAME** function returns the name of the operating system.

CHOOSE

The **CHOOSE** function returns the object within a series whose position in the series corresponds to a specified number.

Syntax:

CHOOSE (series-object-name , single-integer-expression)

Meaning:

CHOOSE (from_these_objects , pick_the_nth_one)

Returns:

A single object whose index within the from_these_objects series matches the number specified by pick_the_nth_one. If that member of the series does not exist, **CHOOSE** returns "none".

You can use **CHOOSE** to use a variable value to specify the index for a particular object from a series. The CHOOSE function does not operate on a series object that was returned by another function, such as EXTRACT.

Examples

- **CHOOSE** (Row:DBSelect , 2)
Returns the second **Row**
- **CHOOSE** (Set:Claim , **INDEX** (Row:Header))
Returns the **Set** within the **Claim** that corresponds to the index of the **Row** of **Header**.

Related function

- **LOOKUP**

DBLOOKUP

The **DBLOOKUP** function executes an SQL statement against a database. The SQL statement can be any permitted by your database management system or ODBC driver.

When the **DBLOOKUP** function is used in a map, the default **OnSuccess** action is adapter specific. The default **OnFailure** action is to rollback any changes made during map processing. The default **Scope** will be integral unless the map is defined to run in bursts (which is the case when one or more inputs have the **FetchAs** property set to **Burst**).

There are two ways to specify arguments for **DBLOOKUP**.

You can use **DBLOOKUP** to execute an SQL statement when you want to execute a **SELECT** statement to retrieve a specific column value in a large table in a database using the value of another input, rather than defining the entire table as an input card and using the **LOOKUP**, **SEARCHDOWN**, or **SEARCHUP** functions.

You can use DBLOOKUP to execute an SQL statement when you want to execute a SELECT statement to retrieve a specific column value from a table or database that might vary based on a parameter file. Using Meaning 2 of the **DBLOOKUP** function allows these parameters to be dynamically specified at run time.

Syntax:

DBLOOKUP (single-text-expression , single-text-expression , [single-text-literal])

Meaning:

1. **DBLOOKUP** (SQL_statement , mdq_filename , database_name)
2. **DBLOOKUP** (SQL_statement , parameters)

Returns:

A single text item

The **DBLOOKUP** function returns the results of the query in the same format as a query specified for a map input card, except that it does not include the last carriage return/linefeed. Because this information is removed, it is easier to make use of a single value extracted from a database.

Arguments for meaning 1

DBLOOKUP (SQL_statement , mdq_filename , database_name)

- **SQL_statement**

The first argument is an SQL statement as a text string. This can be any valid SQL statement permitted by your database management system and supported by your database-specific driver. In addition to a fixed SQL statement, this argument can be a concatenation of text literals and data objects, enabling the concatenation of data values into your SQL statement.

- **mdq_filename**

The second argument is the name of a database query file (**.mdq**) produced by the Database Interface Designer. It contains the definition of the database that the SQL statement is to be executed against. If the **.mdq** file is in a directory other than the directory of the map, the path must be specified.

Note: The **.mdq** file is accessed at map build time and is not needed at run time.

- **database_name**

The third argument is the name of a database in the database query file (**.mdq**) as defined in the Database Interface Designer.

If used in this way, both the **.mdq** filename and database name must be literals.

Arguments for meaning 2

DBLOOKUP (SQL_statement , parameters)

- **SQL_statement**

The first argument is an SQL statement as a text string. This can be any valid SQL statement permitted by your database management system and supported by your database-specific driver. In addition to a fixed SQL statement, this argument can be a concatenation of text literals and data objects, enabling the concatenation of data values into your SQL statement.

- **parameters**

The second argument is a set of parameters, either:

- -MDQ mdqfilename -DBNAME dbname
 - or-

- -DBTYPE database_type [database specific parameters]

The keyword -MDQ is followed by the name of the database query file (**.mdq**) produced by the Database Interface Designer. This **.mdq** file contains the definition of the database. If the **.mdq** file is in a directory other than the directory of the map, the path must be specified. The **.mdq** filename is followed by the keyword -DBNAME and the database name as specified in the Database Interface Designer.

Using this syntax, the **.mdq** file is accessed at run time and must be present.

The keyword -DBTYPE is followed by a keyword specifying the database type (for example, ODBC or ORACLE) followed, optionally, by database-specific parameters.

This syntax does not use an **.mdq** file, because the database-specific parameters provide the information required to connect to the database. See the Resource Adapters documentation for detailed information about the database-specific parameters that can be specified.

When used with Meaning 2, **DBLOOKUP** must conform to these rules:

- All keywords (for example, -DBTYPE) can be upper or lowercase, but not mixed.
- A space is required between the keyword and its value (for example, -DBTYPE ODBC).
- The order of the keywords is not important.

All database-specific parameters are optional.

Examples

Assume that you have a table named "PARTS" that contains the following data:

PART_NUMBER	PART_NAME
1	1/4" x 3" Bolt
2	1/4" x 4" Bolt

Assume that this database has been defined using the Database Interface Designer in a file named **mytest.mdq** and that the name of the database, as specified in the **.mdq** file, is **PartsDB**.

- **DBLOOKUP** ("SELECT PART_NAME from PARTS where PART_NUMBER =1", "mytest.mdq", "PartsDB")
Returns: 1/4" x 3" Bolt

Using Meaning 2, you can specify the **DBLOOKUP** this way:

- **DBLOOKUP**("SELECT PART_NAME from PARTS where PART_NUMBER =1",
"-MDQ mytest.mdq -DBNAME PartsDB")
where both the **.mdq** file name and database name is specified.

Using Meaning 2, you can also specify the database type and the appropriate database-specific parameters:

- **DBLOOKUP**("SELECT PART_NAME from PARTS where PART_NUMBER =1",
"-DBTYPE ORACLE -CONNECT MyDB -USER janes")

Related functions

- **DBQUERY**
- **EXTRACT**
- **FAIL**
- **LASTERRORCODE**
- **LASTERRORMSG**
- **LOOKUP**
- **SEARCHDOWN**
- **SEARCHUP**
- **VALID**

For more examples using the **DBLOOKUP** function, see the Database Interface Designer documentation.

DBQUERY

The **DBQUERY** function executes an SQL statement against a database. The SQL statement can be any permitted by your database management system or ODBC driver.

When the **DBQUERY** function is used in a map, the default **OnSuccess** action is adapter specific. The default **OnFailure** action is to rollback any changes made during map processing. The default **Scope** will be integral unless the map is defined to run in bursts (which is the case when one or more inputs have the **FetchAs** property set to **Burst**).

There are two ways to specify the arguments for **DBQUERY**. You can use **DBQUERY** [Meaning 1] to execute an SQL statement when you want to look up information in a database using a parameterized query that is based on another value in your data. If your SQL statement is a SELECT statement, the **DBQUERY** function might be used in conjunction with the **RUN** function to issue dynamic SELECT statements whose results can be used as input to another map.

You can also use the **DBQUERY** function [Meaning 2] to execute an SQL statement when the database, table, or other database parameters might vary; perhaps being supplied by a parameter file.

Syntax:

```
DBQUERY (single-text-expression , single-text-expression ,
[ single-text-literal ])
```

Meaning:

1. **DBQUERY** (SQL_statement , mdq_filename , database_name)
2. **DBQUERY** (SQL_statement , parameters)

Returns:

A single text item

If your SQL statement is a SELECT statement, the results of the query in the same format as a query specified as a map input card, including row delimiters and terminators, and so on.

If your SQL statement is anything other than a SELECT statement, "none".

Arguments for meaning 1

DBQUERY (SQL_statement , mdq_filename , database_name)

- **SQL_statement**

The first argument is an SQL statement as a text string. This can be any valid SQL statement that is permitted by your database management system and supported by your database-specific driver. In addition to a fixed SQL statement, this argument can be a concatenation of text literals and data objects, enabling the concatenation of data values into your SQL statement.

- **mdq_filename**

The second argument is the name of a database query file (**.mdq**) produced by the Database Interface Designer. It contains the definition of the database that the SQL statement is to be executed against. If the **.mdq** file is in a directory other than the directory of the map, the path must be specified.

Note: The **.mdq** file is accessed at map build time and is not needed at run time.

- **database_name**

The third argument is the name of a database in the database query file (**.mdq**) as defined in the Database Interface Designer.

If used in this way, both the **.mdq** filename and database name must be literals.

Arguments for meaning 2

DBQUERY (SQL_statement , parameters)

- The first argument is an SQL statement as a text string. This can be any valid SQL statement that is permitted by your database management system and supported by your database-specific driver. In addition to a fixed SQL statement, this argument can be a concatenation of text literals and data objects, enabling the

- concatenation of data values into your SQL statement.
- The second argument is a set of parameters, either:
 - MDQ mdqfilename -DBNAME dbname
 - or-
 - DBTYPE database_type [database specific parameters]
 The keyword -MDQ is followed by the name of the database query file (**.mdq**) produced by the Database Interface Designer. This **.mdq** file contains the definition of the database. If the **.mdq** file is in a directory other than the directory of the map, the path must be specified. The **.mdq** filename is followed by the keyword -DBNAME and the database name as specified in the Database Interface Designer.
- Note: Using this syntax, the **.mdq** file is accessed at run time and must be present.
- The keyword -DBTYPE is followed by a keyword specifying the database type (for example, ODBC or ORACLE) followed, optionally, by database-specific parameters.
- Note: This syntax does not use an **.mdq** file, because the database-specific parameters provide the information required to connect to the database. See the appropriate database adapter documentation for detailed information about database-specific parameters.

When used with Meaning 2, **DBQUERY** must conform to these rules:

- All keywords (for example, -DBTYPE) can be upper or lower case, but not mixed.
 - A space is required between the keyword and its value (for example, -DBTYPE ODBC).
 - The order of the keywords is not important.
- All database-specific parameters are optional.

Examples

Assume that you have a table named "PARTS" that contains the following data:

PART_NUMBER	PART_NAME
1	1/4" x 3" Bolt
2	1/4" x 4" Bolt

Also assume that this database has been defined using the Database Interface Designer in a file named **mytest.mdq** and that the name of the database, as specified in the **.mdq** file, is **PartsDB**.

```
DBQUERY ( "SELECT * from
PARTS" , "mytest.mdq" , "PartsDB" )
Returns 1|¼" x 3" Bolt<cr><lf>2|¼" x 4" Bolt<cr><lf>
```

where <cr><lf> is a carriage return followed by a line feed.

Using Meaning 2, you can also specify the DBQUERY this way:

```
DBQUERY ( "SELECT * from PARTS" , "-MDQ mytest.mdq -DBNAME PartsDB"
)
where both the .mdq file name and database name are specified.
```

Or, specify it this way, using Meaning 2 by specifying the database type and the appropriate database-specific parameters:

```
DBQUERY ( "SELECT * from PARTS" , "-DBTYPE ORACLE -CONNECT MyDB -USER janes"
)
```

Assume that you have an input file containing one order record. To map that order to another proprietary format, you also have a parts table with pricing information for every part for every customer, a very large table. Rather than using the entire parts table as the input to your map, you might use the **RUN** function with a **DBQUERY** to dynamically select only those rows from the parts table corresponding to the customer in the order file, as follows:

```
RUN ( "MapOrder.MMC" ,
"IE2" + DBQUERY ( "SELECT * FROM Parts WHERE CustID = "
+ CustomerNo:OrderRecord:OrderFile + " ORDER BY PartNo" ,
"PartsDB.MDQ" , "PartsDatabase" ) )
```

Related functions

- DBLOOKUP
- EXTRACT
- FAIL
- LASTERRORCODE
- LASTERRORMSG
- LOOKUP
- SEARCHUP
- SEARCHDOWN
- VALID

DDEQUERY

The **DDEQUERY** function allows you to interface to other Windows applications such as Trading Partner PC, Excel, and so forth, provided that certain criteria are met. For example, if you receive an Excel spreadsheet file, you must have the appropriate version of the Excel application installed (that is compatible with the file received) and the application must be open.

Syntax:

```
DDEQUERY (single-text-expression , single-text-expression , single-text-expression)
```

Meaning:

```
DDEQUERY (application_name , topic , text)
```

Returns:

A single text item from an application

Examples

- **DDEQUERY** ("excel", "[MKTPRICE.XLS]Sheet1", "R8C1:R14C3")

In this example, **DDEQUERY** is used to get data from an Excel spreadsheet. The third argument, **R8C1:R14C3**, specifies the location of the data in the spreadsheet. (In Excel, the 8th row, 1st column to the 14th row, 3rd column is A8:C14.) The content of this spreadsheet range is returned as a single text item.

This example assumes that the application, the map, and the spreadsheet all reside in the same directory. If they are not in the same directory you must add the path. For example:

```
DDEQUERY ("excel", "c:\spreadsheet[MKTPRICE.XLS]Sheet1", "R8C1:R14C3")
```

- **DDEQYUERY** ("tppc","PartnerX","BGyourEDIode")

In this example, **DDEQUERY** is used as a request to Trading Partner PC.

Related functions

- **EXIT**
- **FAIL**
- **GET**
- **LASTERRORCODE**
- **LASTERRORMSG**
- **PUT**
- **RUN**
- **VALID**

EXTRACT

Use **EXTRACT** whenever you need only particular members of a series returned-those that meet a certain condition. An example might be only POs that contain back-ordered items.

The **EXTRACT** function can only be used in a map rule. It cannot be used in a component rule.

The **EXTRACT** function returns all members of a series for which a specified condition is true.

Syntax:

```
EXTRACT (series-object-expression, single-condition-expression)
```

Meaning:

```
EXTRACT (objects_to_extract, condition_to_evaluate)
```

Returns:

A series object.

The result is each member of *series_to_search* for which the condition specified by *condition_to_evaluate* evaluates to "true". **EXTRACT** returns "none", if no member of *series_to_search* has a corresponding *condition_to_evaluate* that evaluates to "true".

Examples

- **EXTRACT** (PO:Transaction , Store# = Location:PO:Transaction)
This example returns all **POs**, individually, whose **Location** is a particular **Store#**.
- **EXTRACT** (Row:DBSelect , ProcessFlag Column:Row:DBSelect = "Y")
This example returns all **Rows** that have a **ProcessFlag Column** value of "Y".

Related Functions

- **CHOOSE**
- **LOOKUP**
- **SEARCHDOWN**
- **SEARCHUP**

GETANDSET

You can use **GETANDSET** when you have an input file that keeps track of control information that serves as an input to your map and the control information in the file needs to be updated based on processing that occurs in your map.

The **GETANDSET** function gets a fixed length value from the input data stream, updates that value in the input data stream, and returns either the original or updated value.

Syntax:

```
GETANDSET (single-fixed-size-item-object-name, single-item-expression,  
           single-integer-expression)
```

Meaning:

GETANDSET (original_value ,new_value, integer_that_determines_which_value_to_return)

Returns:

A single-fixed-size-item and replaces the original value in the input data stream.

This function does not support decrementing integers past 0 into negative numbers.

GETANDSET updates an input by finding the object represented by *original_value* and replacing it with the value represented by the *new_value*. The function returns either the *original_value* or the *new_value*, depending on the value of the *integer_that_determines_which_value_to_return*. For example,

- If *integer_that_determines_which_value_to_return* has the value 1, *original_value* is returned.
- If *integer_that_determines_which_value_to_return* has the value 2, *new_value* is returned.
- If *integer_that_determines_which_value_to_return* has any other value, *original_value* is returned.
- If one of the input arguments evaluates to "none", **GETANDSET** returns "none".

The original value specified for *new_value* must be a fixed size item. During map execution, the original value is updated to reflect the evaluation of **GETANDSET**.

When the source is a file, that file will be updated after map completion. However, if the source is a database, message, or application, the content of the source is in memory and the source, itself, is not updated.

Examples

- New Tracking# = **GETANDSET** (Tracking#:Card, Tracking#:Card + 3, 1)

This rule finds the object **Tracking#:Card** (assume that it has the value 6), adds 3 to it, giving 9, and replaces the 6 with the 9 in the data location of **Tracking#:Card**. Subsequent references to **Tracking#:Card** will find its value to be 9. If **Card** is an input card whose source is a file, the file is rewritten with the new value.

Because the third input argument (*integer_that_determines_which_value_to_return*) is 1, the returned value is the original value of 6.

GETDIRECTORY

The **GETDIRECTORY** function returns the full path (directory) for the compiled map file or the source or destination associated with a specified card object.You can use **GETDIRECTORY** in a map rule or component rule when you need the full path (directory) of either the compiled map or of a data source or destination.

Syntax:

GETDIRECTORY ([single-simple-object-name])

Meaning:

GETDIRECTORY ([card_or_object_for_which_directory_is_needed])

Returns:

A single text item

Without an argument, **GETDIRECTORY** returns the full path associated with the compiled map.

With an argument, the following occurs:

- From a map rule, the function returns the full path of the source or destination that is associated with the card. In a map rule, the argument must be the name of the card for which to get the directory.
 - From a component rule, the function returns the full path of the source or destination that is associated with the active source or destination.
- The **GETDIRECTORY** command without arguments will return the directory of the compiled map, regardless of whether it is used in a map rule or in a component rule.

Examples

- **GETDIRECTORY** (OrderFile)

If the card **OrderFile** is associated with the data file, *install_dir\order.txt*, **GETDIRECTORY** returns *\install_dir*.

- **GETDIRECTORY** ()

If the compiled map on the HP-UX is */maps/prod/mymap.mmc*, **GETDIRECTORY** returns */maps/prod/*.

Suppose you want to run the map, **MyMap**, from a component rule on **Record** in your input to determine whether input customer names are valid. **MyMap** has two inputs and one output. The first input is the customer name to look up. The second input is a lookup file. The output is a text item whose value is "valid" or "error".

The name of the lookup file is constant; it is always **XREF_TBL.TXT**. However, its location might vary; it will always be in the same directory as the data file used as the source you are trying to validate. For example, if the name of the data file used as the source is **C:\SHR\ABC\INPUT.TXT**, the lookup file name is **C:\SHR\ABC\XREF_TBL.TXT**. If the data file name is **/local/data/somefile**, the lookup file name is **/local/data/XREF_TBL.TXT**, and so forth.

You could use this component rule on **Record** to determine whether the customer name is valid:

```
RUN ("MyMap.mmc", "-IE1S10" + CustomerName:$ + "-IF2" + GETDIRECTORY () + "XREF_TBL.TXT" + "-OE1") = "VALID"
```

Related functions

- **GETFILENAME**
- **GETRESOURCENAME**

GETITXUID

The **GETITXUID** function returns a universally unique identifier (UUID).

Syntax:
resourceLib->**GETITXUID** ()

Meaning:
GETITXUID ()

Returns:
A unique string of 36 single-digit hexadecimal numbers

This function has no arguments but requires parentheses (). This function is valid on all Transformation Extender platforms.

GETLOCALE

The GETLOCALE function returns the locale setting of the computer.

Syntax:
resourceLib->**GETLOCALE** ()

Meaning:
GETLOCALE ()

Returns:
A single text item

The GETLOCALE function returns the locale setting of the computer where the map runs. The locale is returned in the format specified by the operating system. Generally, the return format is an ISO Language Code (as defined by ISO-639) in combination with an ISO Country Code (as defined by ISO-3166) when applicable. The language codes are two lowercase letters and the country codes are two uppercase letters. For example, **en_US** is the code for English (United States).

This function has no arguments but requires parentheses.

Table 1. Examples

System locale	Returns
French	fr
Korean	ko
Brazilian Portuguese	pt_BR
Taiwanese Chinese	zh-TW

GETFILENAME

The **GETFILENAME** function returns the file name for a file adapter source or target of a specified card object. Without an argument, it returns the adapter command associated with the active source or destination.

You can use **GETFILENAME** in a map rule or component rule when you need the data file name.

Syntax:
GETFILENAME ([single-simple-object-name])

Meaning:
GETFILENAME ([card_for_which_resource_info_is_needed])

Returns:
A single text item

With an argument, **GETFILENAME** returns the file name for a file source or target of a specified card object. Without an argument, the function returns the source or destination name associated with the active source or destination.

Examples

- **GETFILENAME** (OrderFile)

If the card **OrderFile** is associated with the data file, *install_dir\order.txt*, **GETFILENAME** returns *install_dir\order.txt*.

Suppose you want to run the map **MyMap** from a component rule on **Record** in your input to determine if input customer names are valid. **MyMap** has two inputs and one output. The first input is the customer name to be looked up. The second input is a lookup file. The output is a text item whose value is "valid" or "error".

The name of the lookup file can vary; it must correspond to the name of the data file used as the source you are trying to validate. For example, if the data file name is **c:\DATA.AAA**, the lookup file name is **c:\LOOKUP.AAA**. If the data file name is **c:\DATA.XYZ**, the lookup file name is **c:\LOOKUP.XYZ**, and so on.

You could use this component rule on **Record** to determine whether the customer name is valid:

```
RUN ( "MyMap.mmc" , "-IE1S10" + CustomerName:$ + " -IF2 LOOKUP." + RIGHT ( GETFILENAME () , 3 ) + " -OE1" ) = "VALID"
```

Related functions

- **GETDIRECTORY**
- **GETRESOURCENAME**

GETPARTITIONNAME

You can use **GETPARTITIONNAME** when you need to know the name of the partition to which the data for the given object belongs.

Syntax:

GETPARTITIONNAME (single-partitioned-object-expression)

Meaning:

GETPARTITIONNAME (partitioned_object)

Returns:

A single simple object name

GETPARTITIONNAME returns a text item that represents the name of the partition to which the data for *partitioned_object* belongs.

Examples

- **GETPARTITIONNAME** (Transaction:File)

The type defined by **Transaction** is a partitioned object that has three partitions: **Add**, **Delete**, and **Modify**. In this example, if the input data for **Transaction** belongs to the **Delete** partition, the **GETPARTITIONNAME** function returns "delete".

Related functions

- **PARTITION**

GETRESOURCEALIAS

The **GETRESOURCEALIAS** function returns the resource alias value specified in a Resource Registry resource name file (.mrn).

Syntax:

resourcelib->GETRESOURCEALIAS (single-text-expression, single-text-expression)

Meaning:

GETRESOURCEALIAS (single-text-expression, single-text-expression)

Returns:

A single text item

The **GETRESOURCEALIAS** function loads a Resource Registry resource configuration file and retrieves the specified alias value. The values are defined as part of the Global object in the .mrc file.

In the below example the resource configuration file is defined as:

```
<?xml version="1.0" encoding="UTF-8"?>
<ResourceCfg>

    <Global>
        <ResourceFile ActiveVirtualServer="test">Company.mrn</ResourceFile>
    </Global>
</ResourceCfg>
```

The resource name file is defined as:

```
<?xml version="1.0" encoding="UTF-8"?>
<MRN>
    <VirtualServerSet>
        <VirtualServer>test</VirtualServer>
    </VirtualServerSet>
    <Resource>
        <Name>company</Name>
        <Value Server="test" encrypt="OFF">ABC</Value>
    </Resource>
</MRN>
```

..when used as a part of this rule:

```
=resourcelib->GETRESOURCEALIAS ("company.mrc", "%company%")
```

In this scenario, the return value is "ABC".

When an absolute path is not defined in the location, the default location is the map directory.

GETRESOURCENAME

The **GETRESOURCENAME** function returns the adapter source or target command of a specified card object. Without an argument, it returns the adapter command associated with the active source or destination.

You can use **GETRESOURCENAME** in a component or map rule when you need to know the name of a source or destination.

Syntax:

GETRESOURCENAME ([single-simple-object-name])

Meaning:

GETRESOURCENAME ([card_for_which_resource_info_is_needed])

Returns:

A single text item

With an argument, the source or destination name that is associated with the card. Without an argument, returns the source or destination name associated with the active source or destination.

Examples

- **GETRESOURCENAME** (OrderFile)

If the card **OrderFile** is associated with the data file, *install_dir\order.txt*, the **GETRESOURCENAME** function returns *install_dir\order.txt*.

Related functions

- **GETDIRECTORY**
 - **GETFILENAME**
-

GETTXINSTALLDIRECTORY

The **GETTXINSTALLDIRECTORY** function returns the IBM Transformation Extender product installation directory.

Syntax:

```
resourceLib->GETTXINSTALLDIRECTORY ()
```

Meaning:

```
GETTXINSTALLDIRECTORY ()
```

Returns:

A single text item

GETTXINSTALLDIRECTORY returns the product installation directory. For example, a Windows operating system might return C:\Program Files\IBM\TransformationExtender_11.0.0, or a UNIX operating system would return the DTX_HOME_DIR value in the setup script.

This function has no arguments but requires parentheses.

This function cannot be used in a z/OS environment.

GETDATADIRECTORY

The **GETDATADIRECTORY** function returns the data directory chosen by the user during installation.

INDEX

You can use the **INDEX** function when you need to select or test particular objects based on their occurrence, or to add a sequence number to output objects.

INDEX returns an integer that represents the index of an object relative to its nearest contained object, counting only valid objects.

INDEX cannot be used in a component rule.

Syntax:

```
INDEX (single-object-name)
```

Meaning:

```
INDEX (object_for_which_to_get_index)
```

Returns:

A single integer

The *object_for_which_to_get_index* variable must be a type name. The result is the index of *object_for_which_to_get_index*.

- If *object_for_which_to_get_index* is an input, this will be the index within all valid objects.
- If *object_for_which_to_get_index* is an output, this will be the index within all objects (valid and invalid).
- Returns 0 if the input argument is "none".

The difference between **INDEXABS** and **INDEX** is that **INDEXABS** counts both valid and invalid instances, whereas **INDEX** counts only valid instances.

Examples

- Message (s) = IF (**INDEX** (Message:Input) > 3, Message:Input, "none")

For example, there are five **Messages** in **Input**. The first three evaluations of this rule return "none". The fourth evaluation returns **Message[4]**. The fifth evaluation returns **Message[5]**.

- Invoice (s) = MyMap (Invoice Segment:Input, INDEX (\$))

For the first evaluation of **MyMap**, **INDEX(\$)** is 1. For the second evaluation of **MyMap**, **INDEX(\$)** is 2.

Related functions

- **CHOOSE**
- **COUNT**
- **COUNTABS**

- **INDEXABS**

INDEXABS

You can use **INDEXABS** when you need the absolute occurrence of a particular object across all occurrences, rather than across only valid occurrences.

The **INDEXABS** function returns an integer that represents the index of an object relative to its nearest contained object, counting both valid and invalid instances of the object.

Syntax:

INDEXABS (single-object-name)

Meaning:

INDEXABS (object_for_which_to_get_index)

Returns:

A single integer

INDEXABS returns an integer that represents the absolute index of *object_for_which_to_get_index*. The integer indicates the instance this object that is in the set of *all* instances of the object, including both valid and invalid occurrences. Returns 0 if the input argument is "none".

- If *object_for_which_to_get_index* is an input, this will be the index within all members of the series, including valid objects, invalid objects, and existing "none"s.
 - If *object_for_which_to_get_index* is an output, this will be the index within all existing members of the series, including existing "none"s.
- The difference between **INDEXABS** and **INDEX** is that **INDEXABS** counts both valid and invalid instances, as well as existing "none"s, whereas **INDEX** counts only valid instances.

Examples

- **INDEXABS** (Message Record:Order:PO_File)

For this example, that **Order** contains the following **Messages**:

Message Record[1] Valid

Message Record[2] Error

Message Record[3] Valid

In a map rule, **INDEXABS** (MessageRecord[3]:Order:PO_File) would evaluate to 3.

If the **INDEX** function was used, **INDEX** (Message Record[3]:Order:PO) would evaluate to 2.

Related function

- **INDEX**

LASTERRORCODE

The **LASTERRORCODE** function returns a text item whose value is the last error code returned by one of a specified set of functions during map execution.

You can use **LASTERRORCODE** to interrogate or report the error code returned by one of the external interface functions.

Syntax:

LASTERRORCODE ()

Meaning:

LASTERRORCODE ()

Returns:

A single text item

LASTERRORCODE has no arguments but it requires parentheses.

LASTERRORCODE returns a text item whose value is the last error code returned by one of a specified set of functions during map execution.

The following functions can fail:

- **DBLOOKUP**
- **DBQUERY**
- **DDEQUERY**
- **EXIT**
- **GET**
- **PUT**
- **RUN**

Examples

- Message = VALID (RUN ("Map1Msg.mmc" , "-AE -OMMSMQ1B ` -QN .\aqeue -CID 2001"), FAIL ("Failure on RUN (" + TEXT (LASTERRORCODE ()) + ")" + LASTERRORMSG ()))

In this example, the **LASTERRORCODE** and **LASTERRORMSG** functions are being used in conjunction with the **FAIL** and **VALID** functions to fail (abort) the map if the map executed by the **RUN** function (**Map1Msg.mmc**) fails. In this example, the map fails and returns the error code and error message reported by the **RUN**

function using the **LASTERRORCODE** and **LASTERRORMSG** functions.

If **Map1Msg** fails because one or more of its inputs was invalid, **Message** is assigned a value of "none". The map aborts and the following message is reported in the execution audit log:

Failure on RUN (8): One or more inputs was invalid.

Related functions

- **DBLOOKUP**
 - **DBQUERY**
 - **DDEQUERY**
-

LASTERRORMSG

The **LASTERRORMSG** function returns a text item whose value is the message corresponding to the last error code returned by one of a specified set of functions during map execution.

Syntax:

LASTERRORMSG ()

Meaning:

LASTERRORMSG ()

Returns:

A single text item

Although **LASTERRORMSG** has no arguments, it does require parentheses.

LASTERRORMSG returns a text item whose value is the message corresponding to the last error code returned by one of a specified set of functions during map execution.

The following is the list of functions that can fail:

• DBLOOKUP	• GET
• DBQUERY	• PUT
• DDEQUERY	• RUN
• EXIT	

Examples

```
• Message = VALID ( RUN (  
"Map1Msg.mmc" , "-AE -OMMSMQ1B ` -QN .\aqueue -CID 2001'" ) , FAIL ( "Failure on RUN (" + TEXT  
(LASTERRORCODE ( ) ) + ")" + LASTERRORMSG ( ) ) )
```

In this example, the **LASTERRORCODE** and **LASTERRORMSG** functions are being used in conjunction with the FAIL and VALID functions to fail (abort) the map if the map executed by the RUN function (**Map1Msg.mmc**) fails. In this example, the map fails and returns the error code and error message reported by the RUN function using the **LASTERRORCODE** and **LASTERRORMSG** functions.

If **Map1Msg** fails because one or more of its inputs was invalid, **Message** is assigned a value of "none". The map aborts and the following message is reported in the execution audit log:

Failure on RUN (8): One or more inputs was invalid.

Related functions

- **DBLOOKUP**
 - **DBQUERY**
 - **DDEQUERY**
-

LOOKDOWN

The **LOOKDOWN** function sequentially searches a series beginning at the end of the series, returning the last member of the series that meets a specified condition.

Syntax:

LOOKDOWN (series-object-expression , single-condition-expression)

Meaning:

LOOKDOWN (series_to_search , condition_to_evaluate)

Returns:

A single object

LOOKDOWN returns the last member of *series_to_search* for which *condition_to_evaluate* evaluates to "true"; it returns "none" if no member of *series_to_search* meets the condition specified by *condition_to_evaluate*.

Examples

- **LOOKDOWN** (Account#:Customer , Company Name:Customer = "ACME")
This example returns the **Account#** of **Customer** whose **Company Name** is ACME.
- **LOOKDOWN** (Part#:Row:DBSelect , Model#:Row:DBSelect = ModelCode:Legacy & Serial#:Row:DBSelect > "123")
This example returns the **Part#** of **DBSelect** where the **Model#** in that row matches the **ModelCode** of **Legacy** and the **Serial#** is greater than 123.

Related functions

- CHOOSE
- EXTRACT
- LOOKUP

Note: **LOOKDOWN** differs from **LOOKUP** in that **LOOKDOWN** returns the last member of *series_to_search* that meets the *condition_to_evaluate*, while **LOOKUP** returns the first member of *series_to_search* that meets the *condition_to_evaluate*.

- SEARCHUP
- SEARCHDOWN

The **LOOKDOWN** function performs a sequential search on an unsorted series, starting from the end of the series. The **SEARCHUP** and **SEARCHDOWN** functions perform a binary search on a series that is sorted in ascending or descending code-page byte-order. The **SEARCHUP** and **SEARCHDOWN** functions are efficient for performing multiple searches on a large series. The **LOOKDOWN** function can be more efficient for performing fewer searches. Note that **LOOKDOWN** returns a different instance from the one returned by **SEARCHUP** and **SEARCHDOWN** if multiple instances in the series match the condition.

LOOKUP

The **LOOKUP** function sequentially searches a series, returning the first member of the series that meets a specified condition.

Syntax:

LOOKUP (series-object-expression , single-condition-expression)

Meaning:

LOOKUP (*series_to_search* , *condition_to_evaluate*)

Returns:

A single object

LOOKUP returns the first member of *series_to_search* for which *condition_to_evaluate* evaluates to "true"; it returns "none" if no member of *series_to_search* meets the condition specified by *condition_to_evaluate*.

Examples

- **LOOKUP** (Account#:Customer , Company Name:Customer = "ACME")
This example returns the **Account#** of **Customer** whose **Company Name** is ACME.
- **LOOKUP** (Part#:Row:DBSelect , Model#:Row:DBSelect = ModelCode:Legacy & Serial#:Row:DBSelect > "123")
This example returns the **Part#** of **DBSelect** where the **Model#** in that row matches the **ModelCode** of **Legacy** and the **Serial#** is greater than 123.

Related functions

- CHOOSE
- EXTRACT

Note: **LOOKUP** differs from **EXTRACT** in that **LOOKUP** returns the first member of *series_to_search* that meets the *condition_to_evaluate*, while **EXTRACT** returns all members (one at a time) of *series_to_search* that meet the *condition_to_evaluate*.

- LOOKDOWN

Note: **LOOKUP** differs from **LOOKDOWN** in that **LOOKUP** returns the first member of *series_to_search* that meets the *condition_to_evaluate*, while **LOOKDOWN** returns the last member of *series_to_search* that meets the *condition_to_evaluate*.

- SEARCHUP
- SEARCHDOWN

The **LOOKUP** function performs a sequential search on an unsorted series, starting from the beginning of the series. The **SEARCHUP** and **SEARCHDOWN** functions perform a binary search on a series that is sorted in ascending or descending code-page byte-order. The **SEARCHUP** and **SEARCHDOWN** functions are efficient for performing multiple searches on a large series. The **LOOKUP** function can be more efficient for performing fewer searches.

MEMBER

Use **MEMBER** when you need to know whether an object occurs within a series.

The **MEMBER** function searches a series, looking for a single specified object in the series. If any object in the series matches the specified object, **MEMBER** returns "true". If there is no match, **MEMBER** returns "false".

Syntax:

MEMBER (single-object-expression , series-object-expression)

MEMBER (single-object-expression , { literal, literal ... })

Meaning:

MEMBER (*object_to_look_for* , *series_of_objects_to_look_at*)

Returns:

"True" or "false"

MEMBER returns "true" if *object_to_look_for* matches one of the values in *series_of_objects_to_look_at*.

It returns "false" if *object_to_look_for* does not match at least one of the values in *series_of_objects_to_look_at*.

The two arguments, *object_to_look_for* and *series_of_objects_to_look_at*, must be objects of the same item interpretation or the same group type. For example, if *object_to_look_for* is a date/time item, *series_of_objects_to_look_at* must be a series of date/time items.

Examples

- **MEMBER** (EntityIDCode:Name, {"BT", "ST"})

This example tests whether **EntityIDCode** has one of a particular set of literal values.

- **MEMBER** (Store#, EntityIDCode:Name)

This example tests whether **Store#** has the same value as any **EntityIDCode:Name**.

Related functions

- **EXTRACT**
 - **LOOKUP**
-

SEARCHDOWN

Use the **SEARCHDOWN** function to look up data within a series of data that is sorted in descending code-page byte-order. **SEARCHDOWN** performs a binary search on the sorted series and returns a related object that corresponds to the item found.

The byte-order of characters varies by code page. For example, because the ASCII **A** character (0x41) has a higher numeric value than the ASCII **O** character (0x30), the **SEARCHDOWN** function requires the **O** character to be sorted after the ASCII **A** character in the series. The EBCDIC **A** character (0xC1) has a lower numeric value than the EBCDIC **O** character, so the **SEARCHDOWN** function requires the EBCDIC **A** character to be sorted be after the EBCDIC **O** character in the series.

The IBM Transformation Extender Data Language property values change the collation order of some characters. For example, the German National Language property value is arranged in byte order, with the exception of certain characters: æ is ordered after a, e is ordered before ä, and so forth. There are particular collation rules for other National Language property values. However, there is no collation order for "Western" and "Japanese" property values, and the deprecated Data Language property values are always in byte order.

Syntax:

SEARCHDOWN (series-object-expression , series-item-object-expression , single-item-expression)

Meaning:

SEARCHDOWN (corresponding_object_to_return , descending_items_to_search , item_to_match)

Returns:

A single object

SEARCHDOWN performs a binary search on the item series of *descending_items_to_search*. The *descending_items_to_search* must be sorted in descending code page byte-order. The value to search for is specified as the *item_to_match*. The object returned (*corresponding_object_to_return*) must be related to *descending_items_to_search* by a common object name.

If no match is found, **SEARCHDOWN** returns "none".

Examples

- **SEARCHDOWN** (Age Column:Row:DBSelect ,
SSN Column:Row:DBSelect ,
SSN_Value:Message)

If there are ten rows in **DBSelect**, the search starts by comparing the first **SSN Column** of the fifth row with the **SSN_Value** in **Message**. If the result matches, **SEARCHDOWN** returns the first **Age Column** of that **Row**. If the value of **SSN Column** is less than the **SSN_Value** in **Message**, the search continues with the third **Row**. If the value of **SSN Column** is greater than the **SSN_Value** in **Message**, the search continues with the seventh Row in **DBSelect**. The search continues in this fashion until either a match is found or until one **Row** is selected. If there is more than one **SSN Column** for the selected **Row**, a similar search is initiated for all **SSN Column**'s for the selected **Row** in **DBSelect**.

SEARCHDOWN returns the first **Age Column** for the selected **Row** of **DBSelect**.

Related functions

- **EXTRACT**
 - **LOOKUP**
 - **SEARCHUP**
-

SEARCHUP

Use the **SEARCHUP** function to look up data within a series of data that is sorted in ascending code-page byte-order. The **SEARCHUP** function performs a binary search on the sorted series and returns a related object that corresponds to the item found.

The byte-order of characters varies by code page. For example, because the ASCII **A** character (0x41) has a higher numeric value than the ASCII **0** character (0x30), the **SEARCHUP** function requires the **0** character to be sorted before the ASCII **A** character in the series. The EBCDIC **A** character (0xC1) has a lower numeric value than the EBCDIC **0** character, so the **SEARCHUP** function requires the EBCDIC **A** character to be sorted before the EBCDIC **0** character in the series.

The IBM Transformation Extender Data Language property values change the collation order of some characters. For example, the German National Language property value is arranged in byte order, with the exception of certain characters: æ is ordered after a, e is ordered before ä, and so forth. There are particular collation rules for other National Language property values. However, there is no collation order for "Western" and "Japanese" property values, and the deprecated Data Language property values are always in byte order.

Syntax:

SEARCHUP (series-object-expression, series-item-object-expression, single-item-expression)

Meaning:

SEARCHUP (corresponding_object_to_return, ascending_items_to_search, item_to_match)

Returns:

A single object

SEARCHUP performs a binary search on the item series of *ascending_items_to_search*. The *ascending_items_to_search* must be sorted in ascending code page byte-order. The value to search for is specified as the *item_to_match* and must be of the same type as the *ascending_item_to_search*.

The object returned (*corresponding_object_to_return*) must be related to *ascending_items_to_search* by a common object name. If no match is found, **SEARCHUP** returns "none".

Examples

- **SEARCHUP** (Age Column:Row:DBSelect, SSN Column:Row:DBSelect, SSN_Value:Message)

If there are ten rows in **DBSelect**, the search starts by comparing the first **SSN Column** of the fifth **Row** with the **SSN_Value** in **Message**. If the result matches, **SEARCHUP** returns the first **Age Column** of that **Row**. If the value of **SSN Column** is greater than the **SSN_Value** in **Message**, the search continues with the third **Row**. If the value of **SSN Column** is less than the **SSN_Value** in **Message**, the search continues with the seventh **Row** in **DBSelect**. The search continues in this fashion until either a match is found or until one **Row** is selected. If there is more than one **SSN Column** for the selected **Row**, a similar search is initiated for all **SSN Column's** for the selected **Row** in **DBSelect**.

SEARCHUP returns the first **Age Column** for the selected **Row** of **DBSelect**.

Related functions

- **EXTRACT**
- **LOOKUP**
- **SEARCHDOWN**

SORTDOWN

Use the **SORTDOWN** function to sort objects in a series in descending sequence. The function returns a series containing the values from the input series in descending order.

- When the character set of the data is ASCII, the objects are sorted in ASCII descending order.
- For all other character sets, the objects are sorted in binary descending order.

Syntax:

SORTDOWN (series-item-expression)

Meaning:

SORTDOWN (item_series_to_sort)

Returns:

A series object

Returns the values in *item_series_to_sort* in descending order.

Examples

- **SORTDOWN** (Abbr:File)

In this example, if **Abbr** has these ASCII values:

ABC, GHI, DEF

SORTDOWN returns these values as a series as:

GHI, DEF, ABC

- The following table displays the results of using the **SORTDOWN** and **SORTUP** functions for a typical series of ASCII text items.

Original Input Series	Result using SORTDOWN	Result using SORTUP
Clams Casino	shrimps	1 Shrimp
Grouper	raw oysters	22 Shrimp
groupers	oysters	A
Shrimp	lobster tails	A 1 A shrimp
shrimps	lobster	A1A Shrimp
lobster	groupers	AA
lobster tails	clams	AAAAA

Original Input Series	Result using SORTDOWN	Result using SORTUP
oysters	aaaaa	Clams Casino
raw oysters	aa	Grouper
clams	a	Rock Lobster
SHRIMP	Snapper	SHRIMP
1 Shrimp	Snapper	Shark Fin Soup
22 Shrimp	Shrimp	Shrimp
A 1 A shrimp	Shark Fin Soup	Snapper
A1A Shrimp	SHRIMP	Snapper
AAAAAA	Rock Lobster	a
aaaaa	Grouper	aa
aa	Clams Casino	aaaaa
AA	AAAAA	clams
A	AA	groupers
a	A1A Shrimp	lobster
Rock Lobster	A 1 A shrimp	lobster tails
Snapper	A	oysters
Snapper	22 Shrimp	raw oysters
Shark Fin Soup	1 Shrimp	shrimps

Related functions

- [SEARCHDOWN](#)
- [SEARCHUP](#)
- [SORTUP](#)
- [UNIQUE](#)

SORTDOWNBY

Use **SORTDOWNBY** to sort the objects of a series based on a component that the objects contain. The **SORTDOWNBY** function returns the input series, sorted in descending order by the value of the specified component.

- When the character set of the data is ASCII, the objects are sorted in ASCII descending order.
- For all other character sets, the objects are sorted in binary descending order.

The **SORTDOWNBY** function does not sort the output of the **EXTRACT**, **REJECT**, or **UNIQUE** functions.

Syntax:

SORTDOWNBY (series-item-expression, single-item-expression)

Meaning:

SORTDOWNBY (group_series_to_sort, group_component_to_sort_by)

Returns:

A series object

SORTDOWNBY returns a series of instances of a group, sorted in descending order by the value of an item that the group contains.

Example

In this example, the group **Soldier** contains the items:

- Name
- Rank
- Serial number

The following command returns the instances of Soldier, sorted in descending order by Rank:

```
SORTDOWNBY (Soldier:Army, Rank:Soldier:Army)
```

Related functions

- [SEARCHDOWN](#)
- [SEARCHUP](#)
- [SORTDOWN](#)
- [SORTUP](#)
- [SORTUPBY](#)

SORTUP

Use **SORTUP** to sort the objects of a series in ascending sequence. The **SORTUP** function returns a series containing the values from an input series in ascending order.

- When the character set of the data is ASCII, the objects are sorted in ASCII ascending order.

- For all other character sets, the objects are sorted in binary ascending order.

Syntax:

SORTUP (series-item-expression)

Meaning:

SORTUP (item_series_to_sort)

Returns:

A series object

SORTUP returns the values in *item_series_to_sort* in ascending order.

Examples

- SORTUP** (Abbr:File)

In this example, if **Abbr** had the ASCII values ABC, GHI, DEF, the **SORTUP** function would return these values as a series: ABC, DEF, GHI.

Related functions

- SEARCHDOWN**
- SEARCHUP**
- SORTDOWN**
- UNIQUE**

SORTUPBY

Use **SORTUPBY** to sort the objects of a series based on a component that the objects contain. The **SORTUPBY** function returns the input series, sorted in ascending order by the value of the specified component.

- When the character set of the data is ASCII, the objects are sorted in ASCII ascending order.
- For all other character sets, the objects are sorted in binary ascending order.

The **SORTUPBY** function does not sort the output of the **EXTRACT**, **REJECT**, or **UNIQUE** functions.

Syntax:

SORTUPBY (series-item-expression, single-item-expression)

Meaning:

SORTUPBY (group_series_to_sort, group_component_to_sort_by)

Returns:

A series object

SORTUPBY returns a series of instances of a group, sorted in ascending order by the value of an item that the group contains.

Example

In this example, the group **Soldier** contains the items:

- Name
- Rank
- Serial number

The following command returns the instances of Soldier, sorted in ascending order by Rank:

```
SORTUPBY (Soldier:Army, Rank:Soldier:Army)
```

Related functions

- SEARCHDOWN**
- SEARCHUP**
- SORTUP**
- SORTDOWN**
- SORTDOWNBY**

UNIQUE

The **UNIQUE** function returns a series containing all "unique" members of a series.

UNIQUE can only be used in a map rule, not in a component rule.

Syntax:

UNIQUE (series-object-expression)

Meaning:

UNIQUE (series_to_evaluate)

Returns:

A series object

UNIQUE evaluates to a series containing all unique members of *series_to_evaluate* and evaluates to "none" if *series_to_evaluate* evaluates to "none".

Examples

- **UNIQUE** (PartNumber:Inventory:File)
Returns the unique **PartNumbers** in **Inventory:File**
- **COUNT** (**UNIQUE** (Customer:Order:File))
Returns the number of unique **Customers** in **Order:File**

Related functions

- **SORTDOWN**
- **SORTUP**

GETENV

The **GETENV** function returns the value of the environment variable. If no variable exists, an empty string is returned.

Syntax:

GETENV(*single-text-expression*)

Meaning:

GETENV (the environment variable name)

Returns:

 A single-text-item

Related reference

- [GETOSNAME](#)

GETOSNAME

The **GETOSNAME** function returns the name of the operating system.

The returned values can be:

- Windows for any version of Microsoft Windows operating system
- zLinux for Linux operating system on z/OS hardware
- Linux for Linux operating system on Intel hardware
- AIX for AIX operating system on PowerPC hardware
- OS/390 for native z/OS operating system

Syntax:

GETOSNAME()

Meaning:

GETOSNAME ()

Returns:

 A single-text-item

Related reference

- [GETENV](#)
- [GETFILENAME](#)

Math and statistics functions

Precede all MATHLIB functions with

mathlib->

- [**ABS**](#)
- [**ACOSINE**](#)
 Use the ACOSINE function to calculate the arccosine of a value.
- [**ASIN**](#)
 Use the ASIN function to calculate the arcsine of a value.
- [**ATAN**](#)
 The ATAN function calculates the arctangent of a value.
- [**ATAN2**](#)
 The ATAN2 function calculates the arctangent of y,x.
- [**COSINE**](#)
 The COSINE function calculates the cosine of a value.

- [**COSINEH**](#)
The COSINEH function calculates the hyperbolic cosine of a value.
 - [**COUNT**](#)
 - [**COUNTABS**](#)
 - [**EXP**](#)
The EXP function calculates the exponential of a value.
 - [**FACTORIAL**](#)
The FACTORIAL function calculates the factorial of a value.
 - [**FROMBASETEN**](#)
 - [**INT**](#)
 - [**LOG**](#)
The LOG function calculates the logarithms of a value.
 - [**LOG10**](#)
The LOG10 function calculates the logarithms for base 10 of a value.
 - [**MAX**](#)
 - [**MIN**](#)
 - [**MOD**](#)
 - [**POWER**](#)
The POWER function calculates x raised to the power of y .
 - [**RAND**](#)
The RAND function returns a pseudorandom number.
 - [**ROUND**](#)
 - [**SEED**](#)
The SEED function primes the RAND and RANDDATA functions so they produce repeated results.
 - [**SIN**](#)
The SIN function calculates the sine of a value.
 - [**SINH**](#)
The SINH function calculates the hyperbolic sine of a value.
 - [**SQRT**](#)
 - [**SUM**](#)
 - [**TAN**](#)
The TAN function calculates the tangent of a value.
 - [**TANH**](#)
The TANH function calculates the hyperbolic tangent of a value.
 - [**TOBASETEN**](#)
 - [**TRUNCATE**](#)
-

ABS

The **ABS** function returns the absolute value of a number.

Syntax:

ABS (single-number-expression)

Meaning:

ABS (number)

Returns:

A single number; the absolute value of a number

Examples

- **ABS** (-3)
Returns 3
 - **ABS** (3)
Returns 3
 - AvailableCredit has a value of -69.42
ABS (AvailableCredit) returns 69.42
 - FlexDollars has a value of 50
ABS ((100 - FlexDollars)/2) returns 25
-

ACOSINE

Use the ACOSINE function to calculate the arccosine of a value.

Syntax:

`mathlib->ACOSINE (single-number-expression)`

Meaning:

`mathlib->ACOSINE (number_to_convert)`

Returns:

A single number item

The result is the arccosine of the converted value.

ASIN

Use the ASIN function to calculate the arcsine of a value.

Syntax:

```
mathlib->ASIN (single-number-expression)
```

Meaning:

```
mathlib->ASIN (number_to_convert)
```

Returns:

A single number item

The result is the arcsine of the converted value.

ATAN

The ATAN function calculates the arctangent of a value.

Syntax:

```
mathlib->ATAN (single-number-expression)
```

Meaning:

```
mathlib->ATAN (number_to_convert)
```

Returns:

A single number item

ATAN2

The ATAN2 function calculates the arctangent of y,x.

Syntax:

```
mathlib->ATAN2 (single-number-expression, single-number-expression)
```

Meaning:

```
mathlib->ATAN2 (number_to_convert)
```

Returns:

A single number item

COSINE

The COSINE function calculates the cosine of a value.

Syntax:

```
mathlib->COSINE (single-number-expression)
```

Meaning:

```
mathlib->COSINE (number_to_convert)
```

Returns:

A single number item

COSINEH

The COSINEH function calculates the hyperbolic cosine of a value.

Syntax:

```
mathlib->COSINEH (single-number-expression)
```

Meaning:

```
mathlib->COSINEH (number_to_convert)
```

Returns:

A single number item

COUNT

You can use the COUNT function to return an integer representing the number of valid input or output objects in a series.

Syntax:

```
COUNT (series-object-expression)
```

Meaning:

```
COUNT (valid_objects_to_count)
```

Returns:

A single integer

The result is the number of *valid_objects_to_count*. If the input argument evaluates to "none", **COUNT** returns 0.

COUNT does not count existing "none's unless its group was defined as an explicit format with a **Track** setting of **Places**.

Examples

- **COUNT** (Claim Record:Patient File)
This example returns the number of valid **Claim Record** objects in **Patient File**.
- **COUNT** (Class IN Transcript)
This example returns the number of valid **Class** objects in **Transcript**.
- **COUNT (UNIQUE** (Class IN Transcript))
This example returns the number of valid **Unique Class** objects in **Transcript**.

Related functions

- **COUNTABS**

COUNTABS

You can use **COUNTABS** to count the input or output objects in a series, regardless of the validity of the object.

Unlike **COUNT**, **COUNTABS** includes both valid and invalid objects in a series.

Syntax:

COUNTABS (series-object-expression)

Meaning:

COUNTABS (objects_to_count)

Returns:

A single-integer

The result is the number of *objects_to_count*. If the input argument evaluates to "none", **COUNTABS** returns 0.

COUNTABS does not count existing "none's unless its group was defined as an explicit format with a **Track** setting of **Places**.

Examples

- **COUNTABS** (Claim Record:Patient File)
This example returns the number of **Claim Record** objects in **Patient File**.
- **COUNTABS** (Class IN Transcript)
This example returns the number of **Class** objects in **Transcript**.

Related functions

- **COUNT**

EXP

The EXP function calculates the exponential of a value.

Syntax:

`mathlib->EXP (single-number-expression)`

Meaning:

`mathlib->EXP (number_to_convert)`

Returns:

A single number item

FACTORIAL

The FACTORIAL function calculates the factorial of a value.

Syntax:

`mathlib->FACTORIAL (single-number-expression)`

Meaning:

`mathlib->FACTORIAL (number_to_convert)`

Returns:

A single number item

FROMBaseten

You can use **FROMBaseten** when you need to convert numbers to a base other than 10.

The **FROMBaseten** function converts an integer to a text item that can be interpreted as a number, using positional notation of the base specified.

Syntax:

FROMBaseten (single-integer-expression, single-integer-expression)

Meaning:

FROMBaseten (positive_integer_to_convert, base_to_convert_to)

Returns:

 A single text item

FROMBaseten returns a text item that results from converting *positive_integer_to_convert* to a text item that can be interpreted as a number using positional notation of the base specified by *base_to_convert_to*.

If *base_to_convert_to* is less than 2 or greater than 36, **FROMBaseten** evaluates to "none". Resulting text item characters A-Z are interpreted as digits having decimal values from 10-35, respectively. The characters returned are uppercase.

Example

- **FROMBaseten** (18, 2)
Returns the value 10010
- **FROMBaseten** (123, 8)
Returns the value 173

Related function

- **TOBaseten**

INT

You can use the **INT** function when you need only the integer portion of a number.

Syntax:

INT (single-number-expression)

Meaning:

INT (number_to_convert)

Returns:

 A single integer

INT returns the integer portion of a number. The result is the integer part of *number_to_convert*. Any fractional part after the decimal point is dropped.

Examples

- **INT** (1.45)
Returns 1
- **INT** (3.6)
Returns 3
- **INT** (Purchase:Amt - Discount:Amt)
Subtracts **Discount:Amt** from **Purchase:Amt** and returns the result as a whole number.

Related functions

- **MOD**
- **ROUND**
- **TRUNCATE**

LOG

The **LOG** function calculates the logarithms of a value.

Syntax:

 mathlib->**LOG** (single-number-expression)

Meaning:

 mathlib->**LOG** (number_to_convert)

Returns:

 A single number item

LOG10

The LOG10 function calculates the logarithms for base 10 of a value.

Syntax:

```
mathlib->LOG10 (single-number-expression)
```

Meaning:

```
mathlib->LOG10 (number_to_convert)
```

Returns:

A single number item

MAX

The **MAX** function returns the maximum value from a series of number, date, time, or text values.

Syntax:

```
MAX (series-item-expression)
```

Meaning:

```
MAX (series_of_which_to_find_max)
```

Returns:

A single number

The result is the maximum value in the input argument series: number, text, or date/time.

Examples

- **MAX (UnitPrice:Input)**
If the values for **UnitPrice** are {20, 10, 100}, MAX returns 100.
- **MAX(EXTRACT(DueDate:Book:Library, CheckedOut:Book:Library = "Y"))**
Returns the maximum (latest) **DueDate** for a book that is checked out from the library.

Related functions

- MIN

MIN

Use MIN when you need the minimum value from a series of number, date, time, or text values.

The MIN function returns the minimum value from a series.

Syntax:

```
MIN (series-item-expression)
```

Meaning:

```
MIN (series_of_which_to_find_min)
```

Returns:

A single number

The result is the minimum value of the input series: number, text, or date/time.

Examples

- **MIN (UnitPrice:Input)**
If the values for **UnitPrice** are {20,10,100}, MIN returns 10.
- **MIN (StartTime:::Schedule)**
Returns the minimum (earliest) **StartTime** in **Schedule**.

Related functions

- MAX

MOD

Use MOD when you need the modulus of an integer and a number.

The MOD function returns the modulus that remains after a number is divided by an integer.

Syntax:
MOD (single-number-expression, single-integer-expression)

Meaning:
MOD (dividend , divisor)

Returns:
A single integer

The result is the remainder (modulus) after *dividend* is divided by *divisor*. The result has the same sign as *divisor*.

The *dividend* is first divided by the integer *divisor*, resulting in a quotient. The modulus is calculated by multiplying the integer portion of the quotient by *divisor* and then subtracting that product from *dividend*.

If *divisor* is 0, MOD returns "none".

Examples

- MOD (3, 2) or MOD (-3, 2)
Returns 1
 - MOD (3, -2) or MOD (-3, -2)
Returns -1
 - MOD (-3, 2) or MOD (-3, -2)
Returns 1
 - MOD (-3, -2) or MOD (-3, -2)
Returns -1
-

POWER

The POWER function calculates x raised to the power of y.

Syntax:
mathlib->POWER (single-number-expression, single-number-expression)

Meaning:
mathlib->POWER (base_number, exponent_number)

Returns:
A single number item

RAND

The RAND function returns a pseudorandom number.

Syntax:
RAND ()
Meaning:
RAND ()
Returns:
A single number item

RAND takes no parameters. If the SEED function is called prior to RAND, RAND will return the same value based on the value to SEED.

Related functions

- RANDDATA
 - SEED
-

ROUND

The **ROUND** function rounds a number to a specified number of decimal places. If the number of decimal places is not specified, the number is rounded to a whole number. The result is in character number format.

Syntax:
ROUND (single-number-expression [, single-integer-expression])
Meaning:
ROUND (number_to_round [, number_of_decimal_places])
Returns:
A single number

ROUND converts *number_to_round* to character format, if necessary, and then produces the value of *number_to_round* rounded to the number of decimal places specified by *number_of_decimal_places*. If *number_of_decimal_places* is not specified, *number_to_round* is rounded to the nearest whole number.

Examples

- **ROUND** (1.46, 1)
Returns 1.5
- **ROUND** (1.46)
Returns 1

Related functions

- TRUNCATE
-

SEED

The SEED function primes the RAND and RANDDATA functions so they produce repeated results.

Syntax:

SEED(*single-number-expression*)

Meaning:

SEED(*number_for_seed*)

Returns:

NONE

Related functions

- **RAND**
 - **RANDDATA**
-

SIN

The SIN function calculates the sine of a value.

Syntax:

mathlib->SIN (*single-number-expression*)

Meaning:

mathlib->SIN (*number_to_convert*)

Returns:

The value of a number

SINH

The SINH function calculates the hyperbolic sine of a value.

Syntax:

mathlib->SINH (*single-number-expression*)

Meaning:

mathlib->SINH (*number_to_convert*)

Returns:

A single number item

SQRT

The **SQRT** function returns the square root of a number.

Syntax:

SQRT (*single-number-expression*)

Meaning:

SQRT (*number*)

Returns:

A single number

SQRT returns the square root of *number*.

Examples

- **SQRT** (4)
Returns 2

SUM

The **SUM** function calculates the sum of a series of numbers.

Syntax:

SUM (series-number-expression)

Meaning:

SUM (series_to_sum)

Returns:

A single number

SUM returns the sum of all members in *series_to_sum*.

Examples

- **SUM** (Quantity:LineItem)

This example calculates the sum of all the **Quantity** objects of **LineItem**.

TAN

The TAN function calculates the tangent of a value.

Syntax:

mathlib->TAN (single-number-expression)

Meaning:

mathlib->TAN (number_to_convert)

Returns:

A single number item

TANH

The TANH function calculates the hyperbolic tangent of a value.

Syntax:

mathlib->TANH (single-number-expression)

Meaning:

mathlib->TANH (number_to_convert)

Returns:

A single number item

TOBASETEN

The **TOBASETEN** function converts a text item that can be interpreted as a number, using positional notation of the base specified, to a base 10 number.

Syntax:

TOBASETEN (single-text-expression, single-integer-expression)

Meaning:

TOBASETEN (text_to_convert, base_to_convert_from)

Returns:

A single integer

TOBASETEN returns a number that results from converting *text_to_convert* that can be interpreted as a number, using positional notation of the base specified by *base_to_convert_from*, to its base 10 representation. Text item characters A-Z are interpreted as digits having decimal values from 10-35, respectively.

If *base_to_convert_from* is less than 2 or greater than 36, **TOBASETEN** evaluates to "none". If *text_to_convert* contains a character that is not alphanumeric or is not in the range specified by *base_to_convert_from*, **TOBASETEN** returns "none".

Examples

- **TOBASETEN** ("A", 16)

Returns the value 10

- **TOBASETEN** ("10", 36)

Returns the value 36

- **TOBASETEN** ("A0", 15)

Returns the value 150

- **TOBASETEN** ("A0", 5)

Returns the value "none"

Related functions

- [FROMBASESETEN](#)

TRUNCATE

The **TRUNCATE** function removes decimal places from a number, leaving a specified number of decimal places.

You can use **TRUNCATE** with a second argument to truncate a number to a specified number of decimal places or without a second argument to reduce a number to an integer by removing all decimal places.

Syntax:

TRUNCATE (single-number-expression[, single-integer-expression])

Meaning:

TRUNCATE (number_to_truncate[, number_of_decimal_places])

Returns:

A single number

TRUNCATE first converts *number_to_truncate* to character format, if necessary. It then truncates that number by removing decimal places to the right of *number_of_decimal_places*. If *number_of_decimal_places* is not used, the number is truncated to an integer.

Examples

- **TRUNCATE** (3.9292, 2)
Returns 3.92
- **TRUNCATE** (3.9292)
Returns 3

Related functions

- [INT](#)
- [ROUND](#)

Text functions

- [BCDTOTEXT](#)
- [COUNTSTRING](#)
- [CPACKAGE](#)
- [CSERIESTOTEXT](#)
- [CSIZE](#)
- [CTEXT](#)
- [DATETOTEXT](#)
- [FILLEFT](#)
- [FILLRIGHT](#)
- [FIND](#)
- [HEXTEXTTOSTREAM](#)
- [LEAVEALPHA](#)
- [LEAVEALPHANUM](#)
- [LEAVENUM](#)
- [LEAVEPRINT](#)
- [LEFT](#)
- [LOWERCASE](#)
- [MAX](#)
- [MID](#)
- [MIN](#)
- [NORMXML](#)
- [NUMBERTOTEXT](#)
- [PACKAGE](#)
- [PROPERCASE](#)
- [REGEXMATCH](#)

The **REGEXMATCH** function tests whether a piece of text matches the regular expression.

- [REGEXPARSE](#)

The **REGEXPARSE** function breaks a text based on regular expression into its capture groups, delimiting them using the provided delimiter.

- [REGEXPOSITION](#)

The **REGEXPOSITION** function returns the first position of a match of the regular expression in the text.

- [REGEXREPLACE](#)

The **REGEXREPLACE** function replaces the first occurrence of the regular expression with the replacement text in the input text.

- [REGEXREPLACEALL](#)

The **REGEXREPLACEALL** function replaces all occurrences of the regular expression with the replacement text in the input text.

- [RANDDATA](#)

The **RANDDATA** function returns random text based on the properties of the output type. The **RAND** function returns a pseudo-random integer between 0 and RAND_MAX (guaranteed to be at least 32,767). The **RANDDATA** function generates a valid random value of the item type that is being built when it is invoked. In

other words, if the rule were "A:Card = RandData()", the random value would be valid data for type "A"—where that type "A" is a component of the specified "Card". The type "A" can be text, a date/time object, or a number.

- [REVERSEBYTE](#)
 - [RIGHT](#)
 - [SEED](#)
The SEED function primes the RAND and RANDDATA functions so they produce repeated results.
 - [SERIESTOTEXT](#)
 - [SQUEEZE](#)
 - [SUBSTITUTE](#)
 - [TEXT](#)
 - [TEXTTOBCD](#)
 - [TEXTTONUMBER](#)
 - [TEXTTOTIME](#)
 - [TIMETOTEXT](#)
 - [TODATETIME](#)
 - [TONUMBER](#)
 - [TRIMLEFT](#)
 - [TRIMRIGHT](#)
 - [UPPERCASE](#)
 - [WORD](#)
-

BCDTOTEXT

The **BCDTOTEXT** function converts the digits in a BCD (Binary Coded Decimal) item to a text item containing the digits of the BCD-encoded item as a string of characters.

Syntax:

BCDTOTEXT (single-text-expression)

Meaning:

BCDTOTEXT (BCD_item_to_convert)

Returns:

A single text item

BCD_item_to_convert is converted from BCD format to a text string containing the digits of the BCD-encoded value as a string of characters.

Numbers in BCD format have two decimal digits in each byte. Each half-byte, therefore, can contain a binary value from 0000 (which represents the digit 0) through 1001, which represents the digit 9). Based on this definition, the following behavior applies:

- If any half-byte of the BCD number contains the binary value 1101 or 1111, that half-byte is ignored.
- If the BCD item contains the binary value 1010, 1011, 1100, or 1110, the output of the function is "none".

Examples

- **BCDTOTEXT** (Qty:Item)
If **Qty** is x'1234', the result is 1234.
- **BCDTOTEXT** (DiscountAmt)
If **DiscountAmt** is x'0123', the result is 0123.
- **BCDTOTEXT** (TotalDollars)
If **Total** is x'F123', the result is 123.

Related functions

- [BCDTOHEX](#)
 - [BCDToInt](#)
 - [TEXTTOBCD](#)
-

COUNTSTRING

You can use the **COUNTSTRING** function when you need to know the number of times a specific text string appears within another text string. The function begins to look for the character string from the first position of the first string, and proceeds forward one byte at a time.

Syntax:

COUNTSTRING (single-text-expression , single-text-expression)

Meaning:

COUNTSTRING (text_to_search , text_to_find_and_count)

Returns:

A single-integer

COUNTSTRING returns an integer that represents the number of times that a specified character string appears in another character string.

The result is a number representing the number of times *text_to_find_and_count* appears within *text_to_search*. If either *text_to_search* or *text_to_find_and_count* evaluates to "none", COUNTSTRING returns 0.

Examples

- **COUNTSTRING** ("banana" , "a")
Returns a value of 3.
- **COUNTSTRING** ("aaaa" , "aa")
Returns a value of 3.

Related functions

- FIND
- LEFT
- MID
- RIGHT

CPACKAGE

CPACKAGE specifies the character set of the output of the function. From that point onward, the data is treated as if it were in that character set. If the data is not in the specified character set, you get the wrong answer.

The character set is required to be specified in this function. If you choose not to specify a character set, you should use the original version of the PACKAGE function.

Syntax:

CPACKAGE (single-object-expression , "character-set-of-object-content")

Meaning:

CPACKAGE (object_to_convert, object_character_set)

Returns:

A single text item

The second argument, *object_character_set*, represents the character set of the resulting object. Character set codes are listed in [Character set codes](#).

Examples

In this example, the group **Record** has an initiator of "#", a terminator of "@" and a delimiter of ";" with the following data:

"#1339X10A,491.38,Green,42x54@"

- **CPACKAGE** (Record:Card, "ASCII")
Returns: #1339X10A,491.38,Green,42x54@

CSERIESTOTEXT

CSERIESTOTEXT specifies the character set of the output of the function. From that point onward, the data is treated as if it were in that character set. If the data is not in the specified character set, you get the wrong answer.

The character set is required to be specified in this function. If you choose not to specify a character set, you should use the original version of the SERIESTOTEXT function.

Syntax:

CSERIESTOTEXT (series-object-expression , "character-set-of-object-content")

Meaning:

CSERIESTOTEXT (series_to_convert, object_character_set)

Returns:

A single text item

The *series_to_convert* argument concatenates the series of the input argument, including nested delimiters but excluding initiators and terminators.

The second argument, *object_character_set*, represents the character set of the resulting object.

Examples

In this example, you have the following data that represents bowler information for a bowling league:

Andrews, Jessica:980206:JBC:145:138:177:159

Little, Randy:980116:BBK:175:168

Wayne, Richard:980102:JBC:185:204:179:164:212

Each record consists of the bowler's name, the date of their last game played, a team code and one or more bowling scores. **Record** is defined as a group that is infix delimited by a colon.

Using the following rule produces results of the concatenation of all scores for all of the bowlers, even though the scores are not all contiguous within the data.

- = **CSERIESTOTEXT** (Score Field:Bowler:Input, "ASCII")
Returns: 145138177159175168185204179164212

You can change the rule to concatenate the list of scores to the bowler's name using the following rule:

- = BowlerName Field:Bowler:Input + " ->" +
CSERIESTOTEXT (Score Field:Bowler:Input, "ASCII")

Returns:

Andrews, Jessica -> 145138177159

Little, Randy -> 175168

Wayne, Richard -> 185204179164212

In this example, you have an input number that is of variable size, followed by a name. There is no syntax that separates the number from the name. You can define the number as a group with **Byte(s)** as a component and provide a component rule for **Byte(s)**, such as:

ISNUMBER (\$)

CSIZE

The **CSIZE** function returns an integer representing the size of a specified object in characters, exclusive of any pad characters.

Syntax:

CSIZE (single object expression)

Meaning:

CSIZE (object whose size is needed)

Returns:

A single integer

Example

This example shows how the CSIZE function differs from the SIZE function.

The SIZE function returns the number of bytes used to represent those symbolic characters in a particular code page.

The CSIZE function returns the number of symbolic characters.

Symbolic text: Mañana

UTF-8 hexadecimal representation : 0x4D 0x61 0xC3 0xA3 0x61 0x6E 0x61

SIZE (Mañana)

Returns 7 bytes

CSIZE (Mañana)

Returns 6 characters

CTEXT

CTEXT specifies the character set of the output of the function. From that point onward, the data is treated as if it were in that character set. If the data is not in the specified character set, you get the wrong answer.

The character set is required to be specified in this function. If you choose not to specify a character set, you should use the original version of the TEXT function.

Syntax:

CTEXT (single-object-expression , "character-set-of-object-content")

Meaning:

CTEXT (object_to_convert , object_character_set)

Returns:

A single text item

The first argument, *object_to_convert*, represents the object that is converted to a text item, excluding the initiator and terminator of the input object.

The second argument, *object_character_set*, represents the character set of the resulting object.

Example

In this example, the group **Record** has an initiator of the pound sign (#), a terminator of the at sign (@), and a delimiter of a comma (,), and uses the following data:

#1339X10A,491.38,Green,42x54@

- **CTEXT** (Record:card, "ASCII")
Returns: 1339X10A,491.38,Green,42x54

The initiator and terminator are not included because only the content of the object is converted to text.

DATETOTEXT

The **DATETOTEXT** function converts a date object or expression to a text item.

Syntax:

DATETOTEXT (single-date-expression)

Meaning:

DATETOTEXT (date_to_convert)

Returns:

A single text item

If *date_to_convert* is a date object name, this returns the date as a text item formatted according to the presentation of the date object.

If *date_to_convert* is a date expression produced by a function, this returns the date as a text item formatted according to the presentation of the output argument of that function.

Examples

- **DATETOTEXT** (ShipDate)

In this example, **ShipDate** is converted from a date to text. If **ShipDate** has a CCYYMMDD presentation, the resulting text item will have that presentation, as well.

- **DATETOTEXT** (**CURRENTDATETIME** ("{MM/DD/CCYY}"))

In this example, **CURRENTDATETIME** evaluates and returns a date in MM/DD/CCYY format. Then **DATETOTEXT** evaluates and returns a text string that is that date in MM/DD/CCYY format.

For example, use **DATETOTEXT**, to do text concatenation. The **FROMDATETIME** function provides greater flexibility in specifying the format of the resulting text item.

Related Functions

- **FROMDATETIME**
 - **NUMBERTOTEXT**
 - **TEXT**
 - **TEXTTODATE**
 - **TEXTTONUMBER**
 - **TEXTTOTIME**
 - **TIMETOTEXT**
 - **TODATETIME**
-

FILLELEFT

The **FILLELEFT** function returns a text item of the length specified. In the output, the text item is preceded with the specified pad value.

You can use **FILLELEFT** when you have a value that needs to be of a fixed size with a variable number of leading characters with a specified value.

Syntax:

FILLELEFT (single-text-expression , single-text-expression , single-integer-expression)

Meaning:

FILLELEFT (text_to_fill , pad_character , pad_to_length)

Returns:

A single text item

The **FILLELEFT** function returns the text string that results from padding out *text_to_fill* by preceding it with the *pad_character* up to *pad_to_length* bytes.

If the *pad-length* argument is less than the number of bytes in the text to fill, no padding will appear.

Examples

- **FILLELEFT** (AcctID:Transaction , "0" , 5)

If **AcctID** has the value 14, **FILLELEFT** returns 00014

- **FILLELEFT** (**NUMBERTOTEXT** (InvoiceAmt) , "*" , 10)

If **InvoiceAmt** has the value 24.75, **FILLELEFT** returns ****24.75

Related functions

- **FILLRIGHT**
 - **LEAVEALPHA**
 - **LEAVEALPHANUM**
 - **LEAVENUM**
 - **LEAVEPRINT**
 - **SQUEEZE**
 - **SUBSTITUTE**
 - **TRIMLEFT**
 - **TRIMRIGHT**
-

FILLRIGHT

The **FILLRIGHT** function returns a text item of the length specified. In the output, the text item is appended with the specified pad value.

You can use **FILLRIGHT** when you have a value that needs to be of a fixed size with a variable number of trailing characters of a specified value.

Syntax:

FILLRIGHT (single-text-expression , single-text-expression , single-integer-expression)

Meaning:

FILLRIGHT (text_to_fill , pad_character , pad_to_length)

Returns:

A single text item

FILLRIGHT returns the text string that results from padding out *text_to_fill* by appending the *pad_character* up to *pad_to_length* bytes.

If the *pad-length* argument is less than the number of bytes in the text to fill, no padding will appear.

Examples

- **FILLRIGHT** (LastName>Contact, " ", 25)

If *Lastname* has the value Peterson, **FILLRIGHT** returns Peterson followed by 17 spaces.

Related functions

- **FILLEFT**
- **LEAVEALPHA**
- **LEAVEALPHANUM**
- **LEAVENUM**
- **LEAVEPRINT**
- **SQUEEZE**
- **SUBSTITUTE**
- **TRIMLEFT**
- **TRIMRIGHT**

FIND

The **FIND** function looks for one text string within another text string and returns to its starting position, if found.

Syntax:

FIND (single-text-expression , single-text-expression
[, single-number-expression])

Meaning:

FIND (text_to_find, where_to_look[, position_to_start_the_search])

Returns:

A single integer

FIND returns the starting position of the text item specified by *text_to_find* within the text item specified by *where_to_look*. A third argument (*position_to_start_the_search*) can be used to specify the location in *where_to_look* for the **FIND** to begin. Bytes in the text are numbered from left to right, with the leftmost byte being position 1.

If *text_to_find* is "none", **FIND** evaluates to "none".

If a third argument is not used or *position_to_start_the_search* evaluates to a negative number, it is assumed to be 1. If *position_to_start_the_search* evaluates to a number greater than the size of *where_to_look*, **FIND** evaluates to "none".

If *text_to_find* is not found in the *where_to_look* string, **FIND** evaluates to 0.

Examples

- **FIND** ("id", "Florida")
Returns the value 5
- **FIND** ("id", "Florida", 8)
Returns 0 because the 8 (*position_to_start_the_search*) is greater than the size of *where_to_look*
- **FIND** ("\", "mypath",2)
Returns 0 because the string "\" was not found in argument 2

Related functions

- **LEFT**
- **MID**
- **RIGHT**

HEXTXTTOSTREAM

HEXTXTTOSTREAM is the reverse of **STREAMTOHEXTXT**. You can use the **HEXTXTTOSTREAM** function to assign a binary text value to a character text item represented by hexadecimal pairs.

HEXTXTTOSTREAM returns a binary text stream whose value is the evaluation of input character text represented by hexadecimal pairs.

Syntax:

HEXTEXTTOSTREAM (single-text-expression)

Meaning:

HEXTEXTTOSTREAM (series_of_hex_pairs)

Returns:

A single byte stream item

This function returns a binary text stream item whose value is the evaluation of input character text in *series_of_hex_pairs*, ignoring <WSP> characters between the hexadecimal pairs. White space characters include space, horizontal tab, carriage return, and line feed characters.

Input formats

The following table shows an example of input in its character text representation as viewed through the character editor, and in its ASCII code representation (binary text stream) as viewed through the hex editor. Each pair of binary text in the hex view represents one character in the character view of the character text.

Input ("41 42 43 44")	Editor View	Value
Character text (hex pairs)	Character	"41 42 43 44"
ASCII code representation (binary text stream)	Hex	0x3431203432203433203434

Examples

- **HEXTEXTTOSTREAM** ("41 42 43 44")
Returns the evaluated value of the input (ASCII) character text string "41 42 43 44" as the output (ASCII) character text string "ABCD" as viewed in the character editor. (The hex view of the input is 0x3431203432203433203434. The hex view of the output is 0x41424344.)
- **HEXTEXTTOSTREAM** ("0D 0A 00")
Returns the evaluated value of the input (ASCII) character text string "0D 0A 00" as the output (ASCII) character text string "<CR><LF><NULL>" as viewed in the character editor. (The hex view of the input is 0x3044203041203030. The hex view of the output is 0xD0A00.)

See Design Studio Introduction documentation for a list of special symbols.

Related functions

- **SYMBOL**
- **STREAMTOHEXTEXT**

LEAVEALPHA

The **LEAVEALPHA** function removes all non-alphabetic characters from a specified text item.

You can use **LEAVEALPHA** to remove non-alphabetic characters such as symbols or numbers from a text item.

Syntax:

LEAVEALPHA (single-text-expression)

Meaning:

LEAVEALPHA (*text_to_change*)

Returns:

A single text item

LEAVEALPHA returns a string containing only the alphabetic characters (for example, A-Z and a-z) in *text_to_change*.

Examples

- **LEAVEALPHA** ("A-b-C-1\$3")
Returns: AbC

Related functions

- **ISALPHA**
- **ISLOWER**
- **ISNUMBER**
- **ISUPPER**
- **LEAVEALPHANUM**
- **LEAVENUM**
- **LEAVEPRINT**

LEAVEALPHANUM

The **LEAVEALPHANUM** function removes all non-alphanumeric characters (such as symbols) from a specified text item.

Syntax:

LEAVEALPHANUM (single-text-expression)

Meaning:

LEAVEALPHANUM (*text_to_change*)

Returns:

A single text item

LEAVEALPHANUM returns a string containing only the alphanumeric characters (for example, A-Z, a-z and 0-9) in *text_to_change*.

Examples

- **LEAVEALPHANUM** ("A-b-C-1\$3")

Returns: AbC13

Related functions

• ISALPHA	• LEAVEALPHA
• ISLOWER	• LEAVENUM
• ISNUMBER	• LEAVEPRINT
• ISUPPER	

LEAVENUM

The **LEAVENUM** function removes all non-numeric characters from a text item. For example, you can use **LEAVENUM** when you want to remove all alphabetic characters and symbols from a text string.

Syntax:

LEAVENUM (*single-text-expression*)

Meaning:

LEAVENUM (*text_to_change*)

Returns:

A single text item

LEAVENUM returns a string containing only the numeric characters (for example, 0-9) in *text_to_change*.

Examples

- **LEAVENUM** ("A-b-C-1\$3")

Returns: 13

Related functions

• ISALPHA	• LEAVEALPHA
• ISLOWER	• LEAVEALPHANUM
• ISNUMBER	• LEAVEPRINT
• ISUPPER	

LEAVEPRINT

The **LEAVEPRINT** function removes all non-printable characters from a text item.

Syntax:

LEAVEPRINT (*single-text-expression*)

Meaning:

LEAVEPRINT (*text_to_change*)

Returns:

A single text item

LEAVEPRINT returns a string containing only printable characters in *text_to_change*.

Examples

- **LEAVEPRINT** ("A-b<SP>C-1\$3<CR><LF>")

Returns: A-b C-1\$3

Related functions

• ISALPHA	• LEAVEALPHA
• ISLOWER	• LEAVEALPHANUM
• ISNUMBER	• LEAVENUM
• ISUPPER	

LEFT

The **LEFT** function returns a specified number of characters from a text expression beginning with the leftmost byte of a text item.

You can use **LEFT** when you need a specific part of a text item. For example, a customer number might have several uses and sometimes only the first 10 characters are needed. Therefore, **LEFT** can be used to return only the leftmost 10 characters.

Syntax:

LEFT (single-text-expression , single-integer-expression)

Meaning:

LEFT (text_to_extract_from , number_of_characters_to_extract)

Returns:

A single text item

LEFT returns the leftmost *number_of_characters_to_extract* characters from *text_to_extract_from* starting at the first (the leftmost) character in *text_to_extract_from*.

If *number_of_characters_to_extract* evaluates to an integer whose value is less than 1, **LEFT** evaluates to "none". If *number_of_characters_to_extract* evaluates to an integer whose value is greater than the size of *text_to_extract_from*, **LEFT** evaluates to the entire value of *text_to_extract_from*.

Examples

- **LEFT** ("Abcd", 2)
Returns Ab
- **LEFT** ("Abcd", 6)
Returns Abcd
- **LEFT** (LastName + " " + FirstName, 25)
Returns the leftmost 25 characters of the text string resulting from the concatenation of **LastName** and **FirstName** (separated by a comma and a space).

Related functions

- **RIGHT**
- **MID**
- **FIND**

LOWERCASE

The **LOWERCASE** function converts an alphabetic text item to all lowercase characters.

Syntax:

LOWERCASE (single-text-expression)

Meaning:

LOWERCASE (text_to_convert)

Returns:

A single text item

LOWERCASE produces a text item in which each byte from the input has been converted to lowercase. Any numeric or symbol characters in the text item remain unchanged.

Examples

- **LOWERCASE** ("A1b2C!")
Returns: a1b2c!

Related functions

- **ISLOWER**
- **ISUPPER**
- **UPPERCASE**

MAX

The **MAX** function returns the maximum value from a series of number, date, time, or text values.

Syntax:

MAX (series-item-expression)

Meaning:

MAX (series_of_which_to_find_max)

Returns:

A single number

The result is the maximum value in the input argument series: number, text, or date/time.

Examples

- **MAX** (UnitPrice:Input)
If the values for **UnitPrice** are {20, 10, 100}, MAX returns 100.
- **MAX(EXTRACT**(DueDate:Book:Library, CheckedOut:Book:Library = "Y"))
Returns the maximum (latest) **DueDate** for a book that is checked out from the library.

Related functions

- **MIN**

MID

You can use the **MID** function when you need specific characters from a text item. **MID** returns one or more characters from a text item.

Syntax:

MID (single-text-expression, single-number-expression,
single-number-expression)

Meaning:

MID (source_text, position_to_start_the_search, number_of_characters)

Returns:

A single text item

MID extracts one or more characters from *source_text* where *position_to_start_the_search* is the position of the first character to extract and *number_of_characters* specifies the number of characters to extract. The first character (leftmost) of *source_text* has a starting position of 1.

If *position_to_start_the_search* is greater than the length of *source_text*, **MID** returns "none". If *position_to_start_the_search* or *number_of_characters* is less than one, **MID** returns "none". If the rightmost number of characters of *source_text*, starting at *position_to_start_the_search*, is less than *number_of_characters*, **MID** returns the rightmost characters starting at the position specified in *position_to_start_the_search*.

Examples

- **MID** ("abc123", 5, 3)
Returns 23
- **MID** ("abc123", 7, 1)
Returns "none" because argument2 is larger than the number of characters in argument1
- **MID** ("abc123", -1, 3)
Returns "none" because argument2 is a negative number
- **MID** ("abc123", 2, -2)
Returns "none" because argument3 is a negative number

Related functions

- **FIND**
- **LEFT**
- **RIGHT**

MIN

Use MIN when you need the minimum value from a series of number, date, time, or text values.

The MIN function returns the minimum value from a series.

Syntax:

MIN (series-item-expression)

Meaning:

MIN (series_of_which_to_find_min)

Returns:

A single number

The result is the minimum value of the input series: number, text, or date/time.

Examples

- MIN (UnitPrice:Input)
If the values for **UnitPrice** are {20,10,100}, MIN returns 10.
- MIN (StartTime::Schedule)
Returns the minimum (earliest) **StartTime** in **Schedule**.

Related functions

- MAX

NORMXML

Use this function to remove XML formatting from an input XML fragment. The function accepts text items only.

Syntax:

NORMXML (single-text-expression)

Meaning:

NORMXML (XML fragment)

Returns:

A single text item

This function removes any XML whitespaces from between a (parent) start tag and the start tag of the first child element; and from between the end tag of the child element and the (parent) end tag. *XML whitespace* pertains to carriage returns, line feeds, tabs, and spaces.

For example, all carriage returns, line feeds, tabs, and spaces in between parent tags `<A>` and `` are removed, excluding any that are inside of child element tags `<a>` and ``.

In the following example, there is a carriage return after `<A>`, ``, and ``. There are also three spaces before `<a>` and three spaces before ``.

```
<A>
  <a>This is sample text</a>
  <b>More   sample text</b>
</A>
```

The **NORMXML** function removes the three carriage returns and the six spaces. As a result, there is one line instead of four lines:

```
<A><a>This is sample text</a><b>More   sample text</b></A>
```

Limitation

When a type has mixed content, the function removes any XML whitespaces from the character data sections. Here is the previous example, but with mixed content added:

```
<A>My first example
  <a>This is sample text</a>
  <b>More   sample text</b>
</A>
```

This is the result of using the **NORMXML** function:

```
<A>Myfirstexample<a>This is sample text</a><b>More   sample text</b></A>
```

The XML whitespaces in **My first example** are removed because they are not enclosed within child element tags.

Using SIZE with NORMXML

The **NORMXML** function converts the input data to Unicode, and then removes the XML formatting from the input XML fragment. The conversion to Unicode is transparent because the normal processing automatically converts the data from Unicode to the character set that was assigned to the output object.

If the **SIZE** function is used with the **NORMXML** function to determine the size of the specified object after **NORMXML** removes the XML formatting from the input XML fragment, the **SIZE** operation calculates the size of the data as it exists at that time, in the Unicode character set. The returned size does not match the size that would have been calculated if the data remained in its original character set, unless the original character set was Unicode.

Using CSIZE with NORMXML

If the **CSIZE** function is used with the **NORMXML** function, the **CSIZE** operation returns the size in characters, regardless of the character set.

NUMBERTOTEXT

The **NUMBERTOTEXT** function converts a character number to a text item that looks like the original object.

You can use **NUMBERTOTEXT** when you need an object that is defined as a number converted to an object defined as text. This is useful when you need to concatenate text, however, the **FROMNUMBER** function provides greater flexibility in specifying the format of the resulting text item.

Syntax:

NUMBERTOTEXT (single-number-expression)

Meaning:

NUMBERTOTEXT (number_to_convert)

Returns:

A single text item

The resulting text looks like the input argument. The result is truncated, if necessary.

Examples

- **NUMBERTOTEXT** (ROUND (1000 - 24.75, 3))

This example converts the result of the calculation (rounded to 3 decimal places) to text, resulting in 975.250.

- **NUMBERTOTEXT** (PurchaseNumber)

This example converts **PurchaseNumber** from a number to text.

Related functions

- **FROMNUMBER**
 - **TEXTTONUMBER**
 - **TODATETIME**
 - **TONUMBER**
-

PACKAGE

The **PACKAGE** function converts a group or item object to a text item, including its initiator, terminator, and any delimiters it contains.

Syntax:

PACKAGE (single-object-expression)

Meaning:

PACKAGE (object_to_convert)

Returns:

A single text item

The **PACKAGE** function converts *object_to_convert*, which must be a type reference to a text item, including the type reference's initiator, terminator, and all delimiters. **PACKAGE** differs from **TEXT** in that it includes the initiator and terminator of the specified type reference.

Examples

- **PACKAGE** (Record:Card)

Returns: #1339X10A,491.38,Green,42x54@

For this example, the group **Record** has an initiator of "#", a terminator of "@" and a delimiter of ",". The data looks like this: "#1339X10A,491.38,Green,42x54@".

Related functions

- **DATETOTEXT**
- **NUMBERTOTEXT**
- **SERIESTOTEXT**
- **TIMETOTEXT**
- **TEXT**

PACKAGE differs from **TEXT** because it includes the initiator and terminator of the input object.

JSON native schema package

JSON schema fields with _V following the name and fields starting with oneOf, allOf and anyOf are the virtual fields.

Virtual JSON fields do not appear in the data and have no syntax. Syntax is controlled by real fields above these virtual fields.

Real fields are fields that appear in the JSON data.

Using a virtual field in a package command will give incomplete syntax. Use a real field either above or below the virtual fields for complete syntax.

PROPERCASE

Use **PROPERCASE** to convert the first alphabetic character in each word to uppercase and all remaining characters to lowercase.

Syntax:

PROPERCASE (single-text-item-expression)

Meaning:

PROPERCASE (text_item_to_convert)

Returns:

A single text item

The **PROPERCASE** function views the text string as containing a series of "words" where the delimiter between words is the space character. For each "word" in *text_item_to_convert*, **PROPERCASE** converts the first alphabetic character (for example, A-Z and a-z) found to uppercase and all other characters are converted to lowercase.

Example

PROPERCASE ("sally jo BRADLEY")

Returns: Sally Jo Bradley

Related functions

- **ISALPHA**
 - **ISLOWER**
 - **ISUPPER**
 - **LOWERCASE**
 - **UPPERCASE**
-

REGEXMATCH

The **REGEXMATCH** function tests whether a piece of text matches the regular expression.

Syntax:

REGEXMATCH (single-text-expression, single-text-expression)

Meaning:

REGEXMATCH (input_text, regular_expression)

Returns:

A single boolean

Examples

- **REGEXMATCH** ("abcdXYZ", "([a-zA-Z]+)")
Returns: True
 - **REGEXMATCH** ("ABCDXYZ", "([0-9]+)")
Returns: False
-

REGEXPARSE

The **REGEXPARSE** function breaks a text based on regular expression into its capture groups, delimiting them using the provided delimiter.

Syntax:

REGEXPARSE (single-text-expression, single-text-expression, single-text-expression)

Meaning:

REGEXPARSE (input_text, regular_expression, delimiter)

Returns:

A single text item

Examples

- **REGEXPARSE** ("44034AAA",
"^(4)([0-9] | [aAbBcC])\s*([0-9]{2})\s?(?:[A-Z]{3})\$", " | ")
Returns: 44|0|34
 - **REGEXPARSE** ("1011XYZ",
"^(10)([0-9] | [aAbBcC])\s*([0-9]{1})\s?(?:[A-Z]{3})\$", ".")
Returns: 10.1.1
-

REGEXPOSITION

The **REGEXPOSITION** function returns the first position of a match of the regular expression in the text.

Syntax:

REGEXPOSITION (single-text-expression, single-text-expression)

Meaning:

REGEXPOSITION (input_text, regular_expression)

Returns:
A single integer

Examples

- **REGEXPOSITION** ("ABC6EF", "(\d+)")
Returns: 4
 - **REGEXPOSITION** ("1234A3456", "(A-Z)+")
Returns: 5
-

REGEXREPLACE

The **REGEXREPLACE** function replaces the first occurrence of the regular expression with the replacement text in the input text.

Syntax:
REGEXREPLACE (single-text-expression, single-text-expression, single-text-expression)
Meaning:
REGEXREPLACE (input_text, regular_expression, replacement_text)
Returns:
A single text item

Examples

- **REGEXREPLACE** ("ABC 7 DEF 8 GHI", "\d+", "###")
Returns: ABC ### DEF 8 GHI
 - **REGEXREPLACE** ("ABC_DEF_GHI", "_", " ")
Returns: ABC DEF_GHI
-

REGEXREPLACEALL

The **REGEXREPLACEALL** function replaces all occurrences of the regular expression with the replacement text in the input text.

Syntax:
REGEXREPLACEALL (single-text-expression, single-text-expression, single-text-expression)
Meaning:
REGEXREPLACEALL (input_text, regular_expression, replacement_text)
Returns:
A single text item

Examples

- **REGEXREPLACEALL** ("ABC 7 DEF 8 GHI", "\d+", "###")
Returns: ABC ### DEF ### GHI
 - **REGEXREPLACEALL** ("10 1 1 1", " ", ".")
Returns: 10.1.1.1
-

RANDDATA

The **RANDDATA** function returns random text based on the properties of the output type. The **RAND** function returns a pseudo-random integer between 0 and RAND_MAX (guaranteed to be at least 32,767). The **RANDDATA** function generates a valid random value of the item type that is being built when it is invoked. In other words, if the rule were "A:Card = RandData()", the random value would be valid data for type "A"—where that type "A" is a component of the specified "Card". The type "A" can be text, a date/time object, or a number.

Syntax:
RANDDATA ()
Meaning:
RANDDATA ()
Returns:
A single object (text, a date/time object, or a number)

RANDDATA takes no parameters. If the SEED function is called prior to RANDDATA, RANDDATA will return the same value for the same item properties it is returning to, based on the value to SEED.

Related functions

- **RAND**
- **SEED**

REVERSEBYTE

Use the **REVERSEBYTE** function when you need the bytes in the opposite sequence. **REVERSEBYTE** reverses the byte order of an item.

Syntax:

REVERSEBYTE (single-item-expression)

Meaning:

REVERSEBYTE (item_to_reverse)

Returns:

A single item

This function reverses the byte order of *item_to_reverse*.

Examples

- **REVERSEBYTE** ("HI MOM!")

Returns "IMOM IH"

- **RIGHT** (FullName, FIND(" ", **REVERSEBYTE** (FullName)) - 1)

If **FullName** is "Alyce N. Wunderland", the above example uses **REVERSEBYTE** to reverse the characters in **FullName** (resulting in "dnalrednuW .N ecylA"). Then, the **FIND** function is evaluated to locate the first space in the resultant string (between the "W" and the ".") that would result in a value of 11. Finally, the **RIGHT** function is evaluated to take the rightmost 10 (11-1) characters of **FullName**; providing the final result of "Wunderland".

Note: The examples are used for illustration only. Do not use this function with string literals.

RIGHT

You can use **RIGHT** when you need a specific part of a text item. For example, a customer number might have several uses and sometimes only the last three characters are needed. **RIGHT** can be used to return only the rightmost three characters.

The **RIGHT** function returns a specified number of characters from a text expression beginning with the rightmost byte of a text item.

Syntax:

RIGHT (single-text-expression , single-integer-expression)

Meaning:

RIGHT (text_to_extract_from , number_of_characters_to_extract)

Returns:

A single text item

RIGHT returns the rightmost *number_of_characters_to_extract* characters from *text_to_extract_from* starting at the last (the rightmost) character in *text_to_extract_from*.

If *number_of_characters_to_extract* evaluates to an integer whose value is less than 1, **RIGHT** evaluates to "none". If *number_of_characters_to_extract* evaluates to an integer whose value is greater than the size of *text_to_extract_from*, **RIGHT** evaluates to the entire value of *text_to_extract_from*.

Examples

- **RIGHT** ("Abcd" , 2)

Returns cd

- **RIGHT** ("Abcd" , 6)

Returns Abcd

- **RIGHT** ("000000" + **NUMBERTOTEXT** (TransactionNum) , 6)

If **TransactionNum** contains 123, this example returns 000123. If **TransactionNum** contains 123456789, this example returns 456789.

Related functions

- **FIND**
- **LEFT**
- **MID**

SEED

The **SEED** function primes the **RAND** and **RANDDATA** functions so they produce repeated results.

Syntax:

SEED(single-number-expression)

Meaning:

SEED(number_for_seed)

Returns:

NONE

Related functions

- **RAND**
- **RANDDATA**

SERIESTOTEXT

You can use the **SERIESTOTEXT** function to project your input data as a series and to interpret it as a text item for output.

SERIESTOTEXT converts a contiguous or non-contiguous series to a text item.

Syntax:

SERIESTOTEXT (series-object-expression)

Meaning:

SERIESTOTEXT (series_to_convert)

Returns:

A single text item

SERIESTOTEXT returns a text item containing the concatenation of the series of the input argument, including nested delimiters but excluding initiators and terminators.

Examples

In this example, you have the following data that represents bowler information for a bowling league:

Andrews, Jessica:980206:JBC:145:138:177:159

Little, Randy:980116:BBK:175:168

Wayne, Richard:980102:JBC:185:204:179:164:212

Each record consists of the bowler's name, the date of their last game played, a team code and one or more bowling scores. **Record** is defined as a group that is infix-delimited by a colon.

Using the rule:

= **SERIESTOTEXT** (Score Field:Bowler:Input)

the following results are produced, which is the concatenation of all of the scores for all of the bowlers, even though the scores are not all contiguous within the data:

145138177159175168185204179164212

However, if the rule was changed, for instance, to concatenate the list of scores to the bowler's name:

= BowlerName Field:Bowler:Input + "-" +
SERIESTOTEXT (Score Field:Bowler:Input)

the following output would be produced:

Andrews, Jessica -> 145138177159

Little, Randy -> 175168

Wayne, Richard -> 185204179164212

In this example, you have an input number that is of variable size, followed by a name. There is no syntax that separates the number from the name. You can define the number as a group with **Byte(s)** as a component and provide a component rule for **Byte(s)**, such as:

ISNUMBER (\$)

Related functions

- **PACKAGE**
- **TEXT**

SQUEEZE

The **SQUEEZE** function removes consecutive duplicate occurrences of a specified character or characters from a text item.

Syntax:

SQUEEZE (single-text-item , single-text-item)

Meaning:

SQUEEZE (text_to_squeeze , duplicate_characters_to_remove)

Returns:

A single text item

SQUEEZE returns a text item with all the consecutive duplicates of *duplicate_characters_to_remove* removed from *text_to_squeeze*.

Examples

- **SQUEEZE** ("AB CDE F", " ")
Returns: AB CDE F
- **SQUEEZE** ("Connolly", "n")
Returns: Conolly

Related functions

- LEAVEALPHA
- LEAVEALPHANUM
- LEAVENUM
- LEAVEPRINT
- SUBSTITUTE

SUBSTITUTE

You can use the **SUBSTITUTE** function to replace or remove a character. You can also use this function for multiple pairs.

Syntax:

SUBSTITUTE (single-text-expression { , single-text-expression , single-text-expression })

Meaning:

SUBSTITUTE (item_to_convert, one-or-more-text-substitution-pairs)

Where each one-or-more-text substitution-pair is

text_to_change , substitute_text

Returns:

A single text item

SUBSTITUTE returns the text string that results from replacing all instances of the first *text_to_change* with *substitute_text* in *item_to_convert*, then replaces all instances of the second *text_to_change* with the *substitute_text* in the result of the first substitution, and so forth.

Examples

- **SUBSTITUTE** ("123*456*7" , "*" , "/")
Finds 123*456*7 and returns 123/456/7
- **SUBSTITUTE** ("120-45-6789" , "-" , "")
Finds 120-45-6789 and returns 120456789
- **=SUBSTITUTE** ("ABBA" , "B" , "A" , "A" , "B")
This example illustrates multiple searches for the **SUBSTITUTE** function.

The first search-and-replace finds all "B"s and returns "A"s: AAAA

The next search-and-replace finds all "A"s and returns "B"s: BBBB

The end result is a return of: BBBB

Related functions

- LEAVEALPHA
- LEAVEALPHANUM
- LEAVENUM
- LEAVEPRINT
- SQUEEZE

TEXT

You can use the **TEXT** function to convert an object to a text item or when echoing entire data objects to another map.

TEXT converts the content of a group or item object to a text item.

Syntax:

TEXT (single-object-expression)

Meaning:

TEXT (object_to_convert)

Returns:

A single text item

The **TEXT** function converts *object_to_convert* to a text-item, excluding the initiator and terminator of the input object.

Examples

- **TEXT** (Record:card)

Data: #1339X10A,491.38,Green,42x54@

Returns: 1339X10A,491.38,Green,42x54

In this example, the group **Record** has an initiator of the pound sign (#), a terminator of the at-sign (@), and a delimiter of a comma (,).

The initiator and terminator are not included because only the content of the object is converted to text.

Related functions

- **DATETOTEXT**
 - **FROMDATETIME**
 - **FROMNUMBER**
 - **NUMBERTOTEXT**
 - **PACKAGE**
Note: **TEXT** differs from **PACKAGE** in that it does not include the initiator and terminator of the input object.
 - **SERIESTOTEXT**
 - **TIMETOTEXT**
-

TEXTTOBCD

The **TEXTTOBCD** function converts a text item from decimal digits to BCD (Binary Coded Decimal) format.

Syntax:

TEXTTOBCD (single-integer-text-expression)

Meaning:

TEXTTOBCD (text_to_be_converted)

Returns:

A single BCD-formatted text item

TEXTTOBCD converts *text_to_be_converted* (which consists of decimal digits) to BCD format. In this format, each byte contains two decimal digits represented as binary numbers. If there is an odd number of decimal digits in the input, the high-order half-byte of the leftmost output byte will contain the decimal value 15 (hex "F").

If anything other than a decimal digit is encountered in the input, **TEXTTOBCD** returns "none".

Examples

- **TEXTTOBCD** ("1234")
Returns the hexadecimal value x'1234'
- **TEXTTOBCD** ("123A")
Returns "none"
- **TEXTTOBCD** ("123")
Returns the hexadecimal value x'F123'

In this example, the values shown as input ("123") are meant to represent character items in the native character set to the machine on which the map is running. On a personal computer, "123" would contain the ASCII characters for the digits that have the hexadecimal values "31", "32", and "33". The output, described as "the hexadecimal value 'F123'", consists of the two binary bytes "F1" and "23".

On an IBM mainframe the input string would contain EBCDIC characters for the digits that have the hexadecimal values "F1", "F2", "F3", Âº, but the output would be the same as the personal computer output.

Related functions

- **BCDTOHEX**
 - **BCDToint**
 - **BCDTOTEXT**
-

TEXTTONUMBER

Use **TEXTTONUMBER** to convert text to a number.

The **TONUMBER** function provides greater flexibility for specifying the format of the text item that is to be converted to a number.

Syntax:

TEXTTONUMBER (single-text-expression)

Meaning:

TEXTTONUMBER (text_to_convert_to_number)

Returns:

A single character number

The *text_to_convert_to_number* must be in integer or ANSI-formatted (floating point) presentation. The resulting number looks like the input argument, however, nonsignificant zeroes to the right of the decimal separator will be truncated. If the input argument is in error (for example, it is not a recognizable as a valid number), the

result is "none".

When specified as in ANSI-formatted presentation, the text string must meet the following requirements:

- The decimal point can be a period, a comma, or "none".
- The leading sign can be a plus sign, a minus sign, or "none".
- No thousands separator is allowed.

Examples

- **TEXTT tonumber** (OrderQty)
Returns **OrderQty** as a character number item

Related functions

- DATETOTEXT
- FROMNUMBER
- NUMBERTOTEXT
- TEXTTODATE
- TEXTTOTIME
- TIMETOTEXT

TEXTTOTIME

Use **TEXTTOTIME** when you want to convert an object defined as text that is in HHMM or HHMMSS presentation, to an item defined as time. For greater flexibility, use the **TODATETIME** function for specifying the format of the text item that is to be converted to a date/time.

Syntax:

TEXTTOTIME (single-text-expression)

Meaning:

TEXTTOTIME (text_to_convert_to_time)

Returns:

A single time

The *text_to_convert_to_time* must be in HHMM or HHMMSS presentation. HH is a two-digit hour in a 24-hour format. If the result is being assigned to a time object, the resulting time looks like the output object. Otherwise, the resulting time looks like the input argument. If the input argument is in error (for example, it is not a valid time), the result is "none".

Examples

- **TEXTTOTIME** (CallTime)
Returns **CallTime** as a time item

Related functions

• DATETOTEXT	• TEXTT tonumber
• FROMDATETIME	• TIMETOTEXT
• NUMBERTOTEXT	• TODATETIME
• TEXTTODATE	

TIMETOTEXT

You can use the **TIMETOTEXT** function to perform text concatenation. For greater flexibility, use the **FROMDATETIME** function for specifying the format of the resulting text item.

TIMETOTEX converts a time object or expression to a text item.

Syntax:

TIMETOTEXT (single-time-expression)

Meaning:

TIMETOTEXT (time_to_convert_to_text)

Returns:

A single text item

If *time_to_convert_to_text* is a time object name, this returns the time as a text item formatted according to the presentation of the input date object.

If *time_to_convert_to_text* is a time expression produced by a function, this returns the time as a text item formatted according to the presentation of the output argument of that function.

Examples

- **TIMETOTEXT** (LeadTime)
In this example, **LeadTime** is converted from a time to text. If **LeadTime** has an HH:MM presentation, the resulting text item will be of that presentation.
- **TIMETOTEXT (CURRENTDATETIME ("{HH:MM:SS}"))**
Here, **CURRENTDATETIME** evaluates and returns a time in HH:MM:SS format. Then, **TIMETOTEXT** evaluates and returns a text string that is that time in HH:MM:SS format.

Related functions

- **DATETOTEXT**
- **FROMDATETIME**
- **NUMBERTOTEXT**
- **TEXTTODATE**
- **TEXTTONUMBER**
- **TEXTTOTIME**
- **TODATETIME**

TODATETIME

The **TODATETIME** function converts a text string of a specified format to a date-time item.

Syntax:

```
TODATETIME ( single-character-text-expression  
[ , single-text-expression ] )
```

Meaning:

```
TODATETIME ( text_to_convert [ , date_time_format_string ] )
```

Returns:

A single character date item

TODATETIME returns the date-time that corresponds to the value specified by *text_to_convert*, which is in the format specified by *date_time_format_string*. If *date_time_format_string* is not specified, it will be assumed that *text_to_convert* is in [CCYYMMDDHH24MMSS] format.

The *date_time_format_string* must conform to the date-time format strings as described in "Format strings".

Examples

- **TODATETIME ("05/14/1999@10:14pm", "[MM/DD/CCYY]@{HH12:MMAM/PM}")**
In this example, a text string containing a date and time is converted to a date-time item.
- **RptDate = TODATETIME (RIGHT (GETRESOURCENAME() , 8) , "CCYYMMDD")**
Assume that you receive a file that contains historical data. The name of the file identifies the date of the historical data. For example, a filename of **19960424** indicates that the data was produced on April 24, 1996. To map this date to **RptDate**, the **TODATETIME** function could be used with the **RIGHT** and **GETRESOURCENAME** functions.

Related functions

- **CURRENTDATE**
- **CURRENTDATETIME**
- **CURRENTTIME**
- **TEXTTODATE**
- **TEXTTOTIME**

TONUMBER

The **TONUMBER** function converts a text string of a specified format to a number.

Syntax:

```
TONUMBER ( single-character-text-expression  
[ , single-text-expression ] )
```

Meaning:

```
TONUMBER ( text_to_convert [ , number_format_string ] )
```

Returns:

A single character number item

TONUMBER returns the number that corresponds to the value specified by *text_to_convert*, which is in the format specified by *number_format_string*. If *number_format_string* is not specified, it will be assumed that *text_to_convert* is in ANSI decimal format (for example, "[L-#####[.'##]")).

The *number_format_string* must conform to the number format strings as described in the "Format strings" topic.

Examples

- **TONUMBER(text_to_convert, "[L+'\$'##]##")**
L+\$ indicates the leading dollar sign is positive. That leading sign and the comma separators are removed when the text is converted to a number.

Input String: \$123,000,000

Output: 123000000

- **TONUMBER(text_to_convert, "####T-")**
Four number signs are required for each whole number, regardless of the actual number of digits in the number.

Input string:	Output:	Note:
12345-	-12345	The output becomes a negative number.
67890	67890	No change occurs.
345-	-345	The output becomes a negative number.

- **TONUMBER(text_to_convert, "####T+'K'-")**
If an invalid character, such as an X, is encountered, nothing is returned.

If a K is encountered, it is treated as a positive indicator.

Input string:	Output:	Note:
11212-	-11212	The output becomes a negative number.
67890X		The X is an invalid character. No number is returned.
54354	54354	No change occurs.
34567K	34567	The K is recognized as a positive sign. The character is removed and the number is returned as a positive.
345-	-345	The output becomes a negative number.

- **TONUMBER(text_to_convert, "[L-'#','##]T')")**
The parentheses indicating a negative number are removed and replaced with a negative sign.

Comma separators are removed when the text is converted to a number.

Input string:	Output:	Note:
(12,345)	-12345	The output becomes a negative number. The comma separator is removed.
67,890	67890	The comma separator is removed.
(345)	-345	The output becomes a negative number.

- **TONUMBER(text_to_convert, "#['']##[.'##5]T+'K'-")**
The optional comma separators are removed, but the decimal points and decimal values are retained.

Input string:	Output:	Note:
54,345.098	54354.098	The comma separator is removed.
67890.0X		The X is an invalid character. No number is returned.
11213-	-11213	The output becomes a negative number.
34567K	34567	The K is recognized as a positive sign. The character is removed and the number is returned as a positive.
345.1-	-345.1	The output becomes a negative number.

Related functions

- **DATETONUMBER**
- **FROMNUMBER**
- **NUMBERTODATE**
- **NUMBERTOTEXT**

TRIMLEFT

You can use the **TRIMLEFT** function to remove spaces or a text string at the beginning of some text.

TRIMLEFT removes leading characters from a text item.

Syntax:

TRIMLEFT (single-text-expression [, single-text-expression])

Meaning:

TRIMLEFT (item_to_trim [, text_to_trim])

Returns:

A single text item

TRIMLEFT removes leading characters that match *text_to_trim* from *item_to_trim* and returns the result as a single text item. If *text_to_trim* is not specified, leading spaces are removed from *item_to_trim*.

Examples

- **TRIMLEFT (" abc")**
Returns: abc
- **TRIMLEFT ("000345" , "0")**
Returns: 345
- **TRIMLEFT (LastName Column:Row , "<WSP>")**

Returns the content of **LastColumn** after removing all leading whitespace characters (space, horizontal tab, carriage return, or line feed characters).

Related functions

• FILLEFTP	• LEAVEPRINT
• FILLRIGHT	• SQUEEZE
• LEAVEALPHA	• SUBSTITUTE
• LEAVEALPHANUM	• TRIMRIGHT
• LEAVENUM	

TRIMRIGHT

You can use the **TRIMRIGHT** function to remove spaces or a text string at the end of text.

TRIMRIGHT removes trailing characters from a text item.

Syntax:

TRIMRIGHT (single-text-expression [, single-text-expression])

Meaning:

TRIMRIGHT (item_to_trim [, text_to_trim])

Returns:

A single text item

TRIMRIGHT removes trailing characters that match *text_to_trim* from *item_to_trim* and returns the result as a single text item. If *text_to_trim* is not specified, trailing spaces are removed from *item_to_trim*.

Example

- **TRIMRIGHT** ("abc ")
Returns: abc
- **TRIMRIGHT** ("Cat in the Hat!?!?!" , "!")
Returns: Cat in the Hat
- **TRIMRIGHT** (**LastColumn:Row**)
Returns the content of **LastColumn** after removing all trailing spaces.

Related functions

• FILLEFTP	• LEAVEPRINT
• FILLRIGHT	• SQUEEZE
• LEAVEALPHA	• SUBSTITUTE
• LEAVEALPHANUM	• TRIMLEFT
• LEAVENUM	

UPPERCASE

The **UPPERCASE** function converts text to all uppercase characters.

Syntax:

UPPERCASE (single-text-expression)

Meaning:

UPPERCASE (text_to_convert)

Returns:

A single text item

UPPERCASE produces a text item in which each byte in *text_to_convert* has been converted to uppercase. Any numeric or symbolic characters in *text_to_convert* remain unchanged.

Examples

- **UPPERCASE** ("abC123!")
Returns ABC123!

Related functions

- [ISLOWER](#)
- [ISUPPER](#)
- [LOWERCASE](#)

WORD

You can use the **WORD** function to parse a text item that is delimited by some character, such as a space or a comma.

WORD returns the characters between two user-defined separators within a text item. The separators are counted from left to right when the third argument is positive, and from right to left when the third argument is negative, enabling the function to search from either end of the item.

Syntax:

WORD (single-text-expression , single-text-expression ,
single-integer-expression)

Meaning:

WORD (text_to_search , word_separator , number_of_word_to_get)

Returns:

A single text item

WORD returns the characters (word) between the *n*th-1 and *n*th *word_separator*, where *n* corresponds to *number_of_word_to_get*.

Define the separator (*word_separator*) and specify the number of the occurrence of that separator. The **WORD** function returns the characters between the *n*th-1 and *n*th separators in the delimited text item.

The separator is case sensitive.

Examples

The following examples assume that a file exists named **Letter** with two text objects named **Line1** and **Line2**:

```
Line1:Congratulations, Mr Brown! You're a winner!  
Line2:You may have already won 1 million dollars!;
```

- **WORD** (Line1:Letter , " ", 3)
Returns: **Brown!**

The exclamation point is returned because it is read as a character within the word before the separator (a space).

- **WORD** (Line1:Letter , " ", 6)
Returns: **winner!**

If the *n*th separator is missing and the *n*th-1 separator exists, the function returns the characters between the last separator and the end of the delimited text item. The separator is a space; "**winner!**" (including the exclamation point) is the sixth word.

- **WORD** (Line2:Letter , "!", 3)
Returns "none"

Both *n*th and *n*th-1 separators are missing. In this example, there is only one separator, located at the end of the text object. As a result, there is no third word because the function sees everything before "!" as the first word.

- **WORD** (Line1:Letter , " ", 1)
Returns: **Congratulations,**

If *n*-1 = 0, the function returns the characters between the beginning of the delimited text item and the first separator.

- **WORD** (Line1:Letter , " ", -1)
Returns: **winner!**

If *n*+1 = 0, then the function returns the characters between the end of the text item and the last separator.

Related functions

- [FIND](#)
- [MID](#)

XML functions

There are several XML functions that you can call from component rules and map rules to process input XML data. They fall under two different categories. The XML functions in one category are called through the XMLLIB library. They are the XPATH (used with XMLLIB), VALIDATEEX, VALIDATE, XPATHEX, XSLT (used with XMLLIB), and XSLTEX functions. The XML functions in the other category are called directly, without using the XMLLIB library. They are the XPATH, XVALIDATE, and XSLT functions.

- [**VALIDATE**](#)
The VALIDATE function validates the XML input by using the XMLLIB library.

- **VALIDATEEX**
The VALIDATEEX function validates the XML input, and logs the function processing in the directory where the map runs by using the XMLLIB library.
 - **XVALIDATE**
The XVALIDATE function validates the XML input.
 - **XPATH functions**
 - **XPATHEX**
The XPATHEX function queries the XML input by using the XMLLIB library, and logs function processing in the directory where the map runs.
 - **XSLT functions**
 - **XSLTEX**
The XSLTEX function applies an XSLT transformation to the XML input, and logs the function processing in the directory where the map runs by using the XMLLIB library.
-

VALIDATE

The VALIDATE function validates the XML input by using the XMLLIB library.

This function validates XML input, which is provided as a text stream or URL, against the provided XML Schema, returning 0 if the validation succeeded or -1 otherwise.

Syntax:

```
VALIDATE (single-text-expression, single-text-expression)
```

Meaning:

```
VALIDATE (xml_url_or_xml_fragment, target_namespace XML_schemaname)
```

Returns:

A single number

Examples

- NUMBERTOTEXT(xmllib->VALIDATE("ipo.in.xml", "urn:h17-org:v3 http://www.example.com/IPO ipo.xsd"))
Returns 0 when validation of the ipo.in.xml file succeeds and returns -1 otherwise.
 - VALID(NUMBERTOTEXT(xmllib->VALIDATE(PACKAGE(source), "ipo.xsd")), LASTERRORMSG())
Returns 0 when validation of the input XML fragment succeeds or returns a validation error message otherwise.
-

VALIDATEEX

The VALIDATEEX function validates the XML input, and logs the function processing in the directory where the map runs by using the XMLLIB library.

This function validates XML input, which is provided as a text stream or URL, against the provided XML Schema. The VALIDATEEX function returns 0 if the validation succeeded or -1 if validation failed, and generates the specified trace log file in the directory where the IBM Transformation Extender map runs.

Syntax:

```
VALIDATEEX (single-text-expression, single-text-expression, tracelog)
```

Meaning:

```
VALIDATEEX (xml_url_or_xml_fragment, target_namespace XML_schemaname, trace_file_name.extension)
```

Returns:

A single number

Example

- NUMBERTOTEXT(xmllib->VALIDATEEX("ipo.in.xml", "urn:h17-org:v3 http://www.example.com/IPO ipo.xsd", "numtotext.log"))
Returns 0 when validation of the ipo.in.xml file succeeds or returns -1 when validation fails, and generates a trace log of function processing.
-

XVALIDATE

The XVALIDATE function validates the XML input.

This function validates XML input, provided as a text stream or URL, against the provided XML Schema.

The default encoding type for a literal string in the Map rule is UTF-16LE.

Syntax:

```
XVALIDATE (single-text-expression, single-text-expression)
```

Meaning:

```
XVALIDATE (xml_url_or_xml_fragment, target_namespace XML_schemaname)
```

Returns:

This function evaluates to a Boolean "true" or "false".

Examples

- IF(XVALIDATE("ipo.in.xml", "http://www.example.com/IP ipo.xsd"), "VALIDATION SUCCESS", LASTERRORMSG())

Returns the message VALIDATION SUCCESS when the validation of the ipo.in.xml file succeeds and returns a message that explains the validation error otherwise.

XPATH functions

- **XPATH**
The XPATH function queries the XML input.
- **XPATH used with XMLLIB**
The XPATH function that is used with XMLLIB queries the XML input.

XPATH

The XPATH function queries the XML input.

This function queries the XML input, which is provided as a text stream or URL, by using the specified XPATH expression and context, returning the result of the evaluation.

Syntax:

XPATH (single-text-expression, single-text-expression, single-text-expression)

Meaning:

XPATH (xml_url_or_xml_fragment, xpath_expression, context_expression)

Returns:

A single text item

Examples

- XPATH("current://ipo.in.xml", "/ipo:purchaseOrders/order/items/item[1]/shipDate", "ipo http://www.example.com/IPO")
Returns the content of the shipDate element from the ipo.in.xml file.
- XPATH(PACKAGE(source), "/ipo:purchaseOrders/order/items/item[1]/shipDate", "ipo http://www.example.com/IPO")
Returns the content of the shipDate element from the input XML fragment.

XPATH used with XMLLIB

The XPATH function that is used with XMLLIB queries the XML input.

This function queries the XML input, which is provided as a text stream or URL, by using the specified XPATH expression and context, returning the result of the evaluation.

Syntax:

XPATH (single-text-expression, single-text-expression, single-text-expression)

Meaning:

XPATH (xml_url_or_xml_fragment, xpath_expression, context_expression)

Returns:

A single text item

Examples

- xmllib->XPATH("ipo.in.xml", "./order//item[1]/shipDate", "/ipo:purchaseOrders")
Returns the content of the shipDate element from the ipo.in.xml file.
- xmllib->XPATH(PACKAGE(source), "/ipo:purchaseOrders/order/items/item[1]/shipDate", "/")
Returns the content of the shipDate element from the input XML fragment.

XPATHEX

The XPATHEX function queries the XML input by using the XMLLIB library, and logs function processing in the directory where the map runs.

This function queries the XML input, provided as a text stream or URL, by using the specified XPath expression and context. The XPATHEX function returns the result of the evaluation and generates the specified trace log file in the directory where the map runs.

Syntax:

XPATHEX (single-text-expression, single-text-expression, single-text-expression, tracelog)

Meaning:

XPATHEX (xml_url_or_xml_fragment, xpath_expression, context_expression, trace_file_name.extension)

Returns:

A single text item

Example

- xmllib->XPATHEX("ipo.in.xml", "./order//item[1]/shipDate", "/ipo:purchaseOrders", "xpathlog.txt")
Returns the content of the shipDate element from the ipo.in.xml file and generates a trace log of function processing.

XSLT functions

- **XSLT**
The XSLT function applies an XSLT transformation.
- **XSLT used with XMLLIB**
The XSLT function that is used with XMLLIB applies an XSLT transformation.

XSLT

The XSLT function applies an XSLT transformation.

This function applies an XSLT transformation, expressed as a text stream or URL, to the provided XML input, returning the result of the transformation.

Syntax:

XSLT (single-text-expression, single-text-expression)

Meaning:

XSLT (xml_url_or_xml_fragment, xslt_url_or_xslt_fragment)

Returns:

A single text item

Examples

- XSLT("current://ipo.in.xml", "current://ipo.xsl")
Applies the XSLT transformation that is defined in the ipo.xsl file to the ipo.in.xml file.
- XSLT(PACKAGE(source), "current://ipo.xsl")
Applies the XSLT transformation that is defined in the ipo.xsl file to the input XML fragment.

XSLT used with XMLLIB

The XSLT function that is used with XMLLIB applies an XSLT transformation.

This function applies an XSLT transformation, expressed as a text stream or URL, to the provided XML input, returning the result of the transformation.

Syntax:

XSLT (single-text-expression, single-text-expression)

Meaning:

XSLT (xml_url_or_xml_fragment, xslt_url_or_xslt_fragment)

Returns:

A single text item

Examples

- xmllib->XMLLIB XSLT("ipo.in.xml", "ipo.xsl")
Applies the XSLT transformation that is defined in the ipo.xsl file to the ipo.in.xml file.
- xmllib->XMLLIB XSLT(PACKAGE(source), "ipo.xsl")
Applies the XSLT transformation that is defined in the ipo.xsl file to the input XML fragment.

XSLTEX

The XSLTEX function applies an XSLT transformation to the XML input, and logs the function processing in the directory where the map runs by using the XMLLIB library.

This function applies an XSLT transformation, expressed as a text stream or URL, to the provided XML input. The XSLTEX function returns the result of the transformation and generates the specified trace log file in the directory where the IBM Transformation Extender map runs.

Syntax:

XSLTEX (single-text-expression, single-text-expression, tracelog)

Meaning:

XSLTEX (xml_url_or_xml_fragment, xslt_url_or_xslt_fragment, trace_file_name.extension)

Returns:

A single text item

Example

- xmllib->XSLTEX("ipo.in.xml", "ipo.xsl", "transform.log")
Applies the XSLT transformation that is defined in the ipo.xsl file to the ipo.in.xml file, and generates a trace log of function processing

Custom functions

From the Type Designer or Map Designer, you can create custom functions to call external libraries.

While the EXIT function is often used to call external libraries, the EXIT function is limited to support only text objects and is operating system dependent. When you create a custom function to use in a map rule, the function is operating system independent and supports text, number, and date-time objects.

Similar to the regular functions that are shipped with the product, a maximum of four parameters are supported per argument.

- [Creating a custom function](#)

To create your own functions, a C or C++ development tool is required, or the Java Native Interface (JNI) if you are working in a Java environment.

Creating a custom function

To create your own functions, a C or C++ development tool is required, or the Java Native Interface (JNI) if you are working in a Java environment.

1. From a component rule in the Type Designer or from a map rule in the Map Designer, right-click and select the Insert Function option.
(Additionally in the Map Designer, if you are not in a map rule, you can create a new function by selecting Rules>New Function.)
2. From the Insert Function window, select Design.
The Custom Function Modules window is displayed.
3. In the Path field, enter the path of where to create the external library.
This should always be in the *install_dir/function_libs* directory.
4. In the Name field, enter a name for the library.
5. Click Add to add a function to the library.
The Function Specifics window is displayed.
6. In the Name field, enter a name for the function.
7. For Return Type, select the function return type from the drop-down list.
You can select up to four function parameters from the respective drop-down list. Choices include Boolean, date, number, text, time, and byte stream.
8. In the space provided, enter a description for your function.
This information will be displayed in the Insert Function window of both Designers.
9. Click OK to validate the selected parameters and close the Function Specifics window.
10. Click Generate.
The Designer generates a collection of operating system specific makefiles and definition files that provide the framework for the function you are creating.
11. Go to the *install_dir/function_libs* directory to view the results.
As a result of the generation process, framework files are created for each operating system that IBM Transformation Extender supports. You can find the operating system-specific makefiles and definition files in the *install_dir/function_libs/your_new_lib* directory.
12. Now you must modify the framework that was generated by the Designer.
The following functions with the parameter information that you selected were exported to the .c file:
 - GetFunctionCount
 - GetFunctionName
 - GetInputParameter
 - GetReturnType
 - GetParameterCount
 - GetFunctionDesc

To complete the new function, open the .c file and add your programming code for the applicable function or functions provided.

Use the .c file to build your dynamic link library (DLL) and then place the DLL in the *install_dir/function_libs/your_new_lib* directory. (All custom designed libraries and functions must be placed in the *install_dir/function_libs* directory.)

The new library name is listed under Category in the Insert Function window, and the new function that you created is placed in the list of functions and will remain available for future use from both the Type Designer and Map Designer applications.

The following example displays how to implement a custom function called SIN.

```
void ConvertToBytes(double returnValue, LPEXITPARAM lpep)
{
    double value = 0.0;
    int decimal = 2, sign = 0, j = 0, k = 0;
    char byString[100];
    char* lpbyData = fcvt(returnValue, 7, &decimal, &sign );
    memset(byString, 0, sizeof(byString));

    if (sign)
        byString[j++] = '-';

    if (decimal <= 0)
    {
        byString[j++] = '0';
        byString[j++] = '.';
        while (decimal != 0)
        {
            byString[j++] = '0'; decimal++;
        }
    }
    else if (decimal > 0)
    {
        while (decimal != 0)
        {
            byString[j++] = lpbyData[k++]; decimal--;
        }
        byString[j++] = '.';
    }
}
```

```

    }
    while (lpbyData[k]) byString[j++] = lpbyData[k++];
    byString[j] = '\0';

    if (NULL == (lpep->lpDataFromApp = GlobalAllocPtr
        (GHND, j + 1)))
    {
        lpep->nReturn = -1;
        lstrcpy(lpep->szErrMsg, "Memory allocation failed in Alternate");
        return;
    }

    memcpy(lpep->lpDataFromApp, byString, j);
    lpep->dwFromLen = j;
    lpep->lpDataFromApp[j++] = '\0';
}

```

```

void CALLBACK EXPORT SIN(LPEXITPARAM lpep)
{
    double value = 0.0;
    double returnValue = 0.0;
    LPEXITPARAMEXTENDED lpExtended = NULL;

    if (lpep->dwSize != sizeof(EXITPARAM))
    {
        return;
    }

    lpExtended = (LPEXITPARAMEXTENDED)lpep->lpv;
    value = atof(lpExtended->lpFirstInputParameter);

    returnValue = sin(value);
    ConvertToBytes(returnValue, lpep);
    lpep->wCleanupAction = GetReturnType("SIN");
    lstrcpy(lpep->szErrMsg, "SIN function was successful");

    return;
}

```

At run time, all custom designed libraries and functions that your maps use must be in the *install_dir/function_libs* directory. When you deploy a map that uses a custom function to a remote host, the library is not transferred. Therefore you must manually copy the custom function library to the *install_dir/function_libs* directory on the remote host.

Date and time format strings

You can use the listed format strings for numbers, dates and times in functions such as the CURRENTDATETIME, FROMNUMBER, TONUMBER, FROMDATETIME, and TODATETIME.

Create custom date and time formats by using the symbols based on the given format strings.

- [Time units](#)
- [Date units](#)
- [Binary date and time format strings](#)
- [Japanese date and time format strings](#)
- [Western date and time format strings](#)

Time units

Symbol	Description
HH24	Hour (based on a 24-hour clock) in two digits (00 to 24)
H24	Hour (based on a 24-hour clock) in one or two digits as needed (0 to 24)
HH12	Hour (based on a 12-hour clock) in two digits (1-12)
H12	Hour (based on a 12-hour clock) in one or two digits as needed (1 to 12)
MM	Minute in two digits (00 to 59)
M	Minute in one or two digits as needed (0 to 59)
SS	Seconds in two digits (00 to 59)
S	Second in one or two digits as needed (0 to 59)
AM/PM	Meridian (AM/PM)
ZZZ	Three character abbreviation for time zone (EST, and so on)

+/-ZZZ
 Hours and minutes before or after Greenwich Mean Time (GMT), also known as Coordinated Universal Time (UTC)

+/-ZZ:ZZ
 4-digit time where the format is a 2-digit hour and 2-digit minute, separated by a colon.

+/-ZZ[ZZ]
 4-digit time where the format is a 2-digit hour and an optional 2-digit minute, separated by colon.

+/-ZZ[ZZ]
 4-digit time where the format is a 2-digit hour and an optional 2-digit minute.

TZD
 4-digit time where the format is a 2-digit hour and 2-digit minute, separated by a colon. Z on output, if value is +/−00:00.

Date units

Symbol	Description
CCYY	Full year including century (2001)
YY	Last two digits of the year (00-99)
MM	Month of the year in two numeric digits (01-12)
M	Month of the year in one or two numeric digits as needed (1 to 12)
MON	First three letters of the month (Jan to Dec)
MONTH	Full name of the month (January to December)
DDD	Day of the year in three numeric digits (001 to 366)
DD	Day of the month in two numeric digits (01 to 31)
D	Day of the month in one or two numeric digits as needed (1 to 31)
DY	First three letters of the weekday (Sun to Sat)
DAY	Full name of the weekday (Sunday to Saturday)
WW	Week of year (1-52)
Qn	Quarter of the year (Q1-Q4)
EEYY	Emperor's year, long form
EY	Emperor's year, short form

Binary date and time format strings

Sub-String Name	Sub-String Value
Binary DateTime	["{" + Date + "}"] + ["{" + Time + "}"]
Date	CCYYMMDD YYMMDD
	CCYYDDD
	YYDDD
Time	HH24MMSS HH24MM

Japanese date and time format strings

Sub-String Name	Sub-String Value
Japanese DateTime	"{" + DateTime + "}"
	"{" + DateTime + "}" + Separator(1) + "{" + DateTime + "}"

```

"{" + DateTime + "}" + "[" + Separator(1) + "{" + DateTime + "}" + "]"

```

DateTime
 Date
 Time

Separator
 Separator is optional. If specified, it can be up to 120 bytes, composed of non-alphabetic characters. Symbol table values, such as <CR>, can be used to indicate non-printable characters.
 The Separator is required if the second DateTime option is used and the format string of the first DateTime ends in a variable sized specification - EY, M, or D for Date or H24, H12, M, or S for Time.

- [Japanese date format strings](#)
- [Japanese time format strings](#)

Japanese date format strings

Sub-String Name	Sub-String Value
Japanese Date	Year + MonthSet(1)
Year	CCYY YY EEYY EY
MonthSet	Separator(1) + "MM" + DayOfMonth(1) "[" + Separator(1) + "MM" + DayOfMonth(1) +] Separator(1) + M + DayOfMonth(1) [+ Separator(1) + M + DayOfMonth(1) +]
DayOfMonth	Separator(1) + "DD" + WeekDay(1) "[" + Separator(1) + "DD" + WeekDay(1) + "]" Separator(1) + "D" + WeekDay(1) "[" + Separator(1) + "D" + WeekDay(1) + "]"
WeekDay	Separator(1) + "DY" "[" + Separator(1) + "DY" + "]" Separator(1) + "DAY" "[" + Separator(1) + "DAY" + "]"
Separator	Separator is optional. If specified, it can be up to 120 bytes composed of non-alphabetic characters. Symbol table values, such as <CR>, can be used to indicate non-printable characters. MonthSet Separator required if Year is EY DayOfMonth Separator required if MonthSet is M WeekDay Separator required if DayOfMonth is D

Japanese time format strings

Sub-String Name	Sub-String Value
Japanese Time	Meridian(1) + HMS
Meridian	"AM/PM"
HMS	"HH24" + MinuteSet(1) "H24" + MinuteSet(1) "HH12" + MinuteSet(1) "H12" + MinuteSet(1)
MinuteSet	

```

Separator(1) + "MM" + SecondSet(1)
"[ " + Separator(1) + "MM" + SecondSet(1) + "]"

Separator(1) + "M" + SecondSet(1)
"[ " + Separator(1) + "M" + SecondSet(1) + "]"

SecondSet
Separator(1) + "SS"
"[ " + Separator(1) + "SS" + "]"

Separator(1) + "S"
"[ " + Separator(1) + "S" + "]"

Separator
Separator is optional. If specified, it can be up to 120 bytes, composed of non-alphabetic characters. Symbol table values, such as <CR> can be used to indicate non-printable characters.
MinuteSet Separator required if HMS is H24 or H12
SecondSet Separator required if MinuteSet is M

```

Western date and time format strings

Substring Name

Substring Value

Western DateTime

```

"{" +DateTime + "}"
"{" +DateTime + "}" + Separator(1) + "{" + DateTime + "}"

"{" +DateTime + "}" + "[" + Separator(1) + "{" + DateTime + "}" + "]"

```

DateTime

Date

Time

Separator

Separator is optional. If specified, it can be up to 120 bytes, composed of non-alphabetic characters. Symbol table values, such as <CR>, can be used to indicate non-printable characters.

The Separator is required if the second DateTime option is used and the format string of the first DateTime ends in a variable sized specification - M, MONTH, D, or DAY for Date or H24, H12, M, or S for Time.

- [Western date format strings](#)
- [Western time format strings](#)

Western date format strings

Sub-String Name

Sub-String Value

Date

```

DateUnit + DatePart2(1)

```

DatePart2

```

Separator(1) + DateUnit + DatePart3(1)
"[ " + Separator(1) + DateUnit + DatePart3(1) + "]"

```

DatePart3

```

Separator(1) + DateUnit + DatePart4(1)
"[ " + Separator(1) + DateUnit + DatePart4(1) + "]"

```

DatePart4

```

Separator(1) + DateUnit
"[ " + Separator(1) + DateUnit + "]"

```

Separator

Separator is optional. If specified, it can be up to 120 bytes, composed of non-alphabetic characters. Symbol table values, such as <CR>, can be used to indicate non-printable characters.

When DateUnit has a variable length (M, MONTH, D, DAY), a Separator must follow that DateUnit if data follows.

Western time format strings

Sub-String Name

Sub-String Value

Western Time

```

Hours + MinutesSet(1) + Meridian(1) + Zone(1)

```

Hours

```

HH24
H24

HH12

H12

MinutesSet
Separator(1) + "MM" + SecondSet(1)
"[ " + Separator(1) + "MM" + SecondSet(1) + "]"

Separator(1) + "M" + SecondSet(1)
"[ " + Separator(1) + "M" + SecondSet(1) + "]"

SecondSet
Separator(1) + "SS" + FractionSet(1)
"[ " + Separator(1) + "SS" + FractionSet(1) + "]"

Separator(1) + "S" + FractionSet(1)
"[ " + Separator(1) + "S" + FractionSet(1) + "]"

FractionSet
Separator(1) + MinPlaces + "-" + MaxPlaces
"[ " + Separator(1) + MinPlaces + "-" + MaxPlaces + "]"

MinPlaces
An integer from 0 to 9

MaxPlaces
An integer from 0 to 9 (must be less than MinPlaces)

Meridian
"AM/PM"
"[AM/PM]"

Zone
"+/-ZZZZ"
"ZZZ"

"+/-ZZ:ZZ"
"+/-ZZ[:ZZ]"
"+/-ZZ[ZZ]"

"TZD"
"[+/-ZZZZ]"
"[ZZZ]"
"[+/-ZZ:ZZ]"
"[+/-ZZ[:ZZ]]"
"[+/-ZZ[ZZ]]"
"[TZD]"

Separator
Separator is optional. If specified, it can be up to 120 bytes, composed of non-alphabetic characters. Symbol table values, such as <CR>, can be used to indicate non-printable characters.
MinuteSet Separator required if Hours is H24 or H12

SecondSet Separator required if MinuteSet is M

FractionSet Separator required if SecondSet is S

```

Number format strings

You can create custom number formats by using the given format strings.

Decimal
"{" + Leading-Sign(1) + Whole# + Fraction + Trailing-Sign(1) + "}"
Integer
"{" + Leading-Sign(1) + Whole# + Trailing-Sign(1) + "}"

- [Leading sign format strings](#)
- [Trailing sign format strings](#)
- [Substring format strings](#)
- [Whole number and fraction format strings](#)

Leading sign format strings

Sub-String Name	Sub-String Value	Positive	Negative
Leading-Sign	"L" + Positive + Negative + Zero(1) -or-	Req	Req
	"L" + "[" + Positive + "]" + Negative + Zero(1) -or-	Opt	Req
	"L" + Positive + "[" + Negative + "]" + Zero(1) -or-	Req	Opt
	"L" + Positive + Zero(1) -or-	Req	-
	"L" + Negative + Zero(1)	-	Req

Trailing sign format strings

Sub-String Name	Sub-String Value	Positive	Negative
Trailing-Sign	"T" + Positive + Negative + Zero(1)	Req	Req
	"T" + "[" + Positive + "]" + Negative + Zero(1)	Opt	Req
	"T" + Positive + "[" + Negative + "]" + Zero(1)	Req	Opt
	"T" + Positive + Zero(1)	Req	-
	"T" + Negative + Zero(1)	-	Req

Substring format strings

Substring Name	Substring Value	Meaning
Positive	"+ Value(1)	
Negative	"-" + Value(1)	
Value	A text-string enclosed in single quotation marks '. Value has a release character of / if the text contains any single quotation mark ' or forward slash / characters.	The sign value of the previous sign-indicator. If Value is not used, the default is + for positive numbers and - for negative numbers.
Zero	"Z" + Value "[" + "Z" + Value + "]"	Specifies the required leading sign value if the number is zero. If Zero is not used, there is no sign associated with a zero. Specifies the optional leading sign value if the number is zero. If Zero is not used, there is no sign associated with a zero.

Whole number and fraction format strings

Whole# substring name

Substring value
Meaning
`MinDigits(1) + "#" + Value(1) + "##" + MaxDigits(1)`
`MinDigits(1) + "#" + "[" + Value(1) + "]" + "##" + MaxDigits(1)`
 Specifies the thousands separator and the range of whole number digits. If min-digits is not used, the default is zero. If max-digits is not used, the default is "S". If a ThousandsItem is specified, a Value must be present as the default for the syntax item.

Fraction substring name

Substring value
Meaning
`"V" + ImpliedPlaces`
 Specifies the decimal to be in an explicit place, and ImpliedPlaces determines where the intended decimal separator is to be placed.
`Value + MinDigits(1) + "##" + MaxDigits(1)`
 Specifies the value of the decimal separator to be required and the range of fraction digits. If min-digits is not used, default is zero. If max-digits is not used, default is "S"
`"[" + Value + MinDigits(1) + "##" + MaxDigits(1) + "]"`
 Specifies the value of the decimal separator to be optional if there is no fractional portion of the number. It also specifies the range of fraction digits. If min-digits is not used, default is zero. If max-digits is not used, default is "S"

RUN function return codes

The **RUN** function return codes and messages might result when using the **RUN** function. Return codes and messages are returned when the particular activity completes. Return codes and messages might also be recorded as specified in the audit logs, trace files, execution summary files, and so forth.

The following table lists the return codes and messages that can result when using the **RUN** function.

Return Code	Message
50	<i>Memory allocation failure</i> Occurs when memory fails.
51	<i>Card override failure</i> Occurs when memory fails.
52	<i>I/O initialization failure</i> Occurs when memory fails.
53	<i>Open audit failure</i> The audit log file is not accessible.
54	<i>No command line</i> There is nothing to process.
55	<i>Recursive command files</i> More than one command file is included in the command line.
56	<i>Invalid command line option -x</i> The option is invalid for the command.
57	<i>Invalid 'W' command line option</i> The Work file option is invalid.
58	<i>Invalid 'B' command line option</i> The Batch (close) file option is invalid.
59	<i>Invalid 'R' command line option</i> The Refresh Rate option is invalid.
60	<i>Invalid 'A' command line option</i> The Audit option is invalid.
61	<i>Invalid 'P' command line option</i> The Paging option is invalid
62	<i>Invalid 'Y' command line option</i> The General I/O Retry option is invalid.
63	<i>Invalid 'T' command line option</i> The Trace option is invalid.
64	<i>Invalid 'G' command line option</i> The Ignore option is invalid
65	<i>Invalid 'I' command line option for input x</i> The Source option is invalid for the identified input.
66	<i>Invalid size in echo command line for input x</i> The size specified using the Size option is greater than memory allowed.
67	<i>Invalid adapter type in command line for input x</i> The adapter is not of a known adapter type. Includes -IMxxx where xxx is an unknown adapter alias.
68	<i>Invalid 'O' command line option for output x</i> The target option is invalid for output x. The number of characters between the single quotation marks that represent the options for an adapter exceed 258 characters in the adapter override.
69	Invalid adapter type in command line for output x

The adapter is not of a known adapter type. Includes -OMxxx where xxx is an unknown adapter alias.

70

Command line memory failure

Occurs when memory is exceeded during echo or override card commands.

71

Invalid 'D' command line option

The Date option is invalid.

72

Invalid 'F' command line option

The Failure option is invalid.

73

Resource manager failure

(Launcher only) The resource manager is not used, possibly a memory failure.

74

Invalid 'Z' command line option

The Ignore option is invalid.

75

Adapter failed to get data on input

Enable the adapter trace to record the adapter activity to discover the cause of the error.

76

Adapter failed to put data on output

Enable the adapter trace to record the adapter activity to discover the cause of the error.

77

Invalid map name

This message can occur in two different cases. First, this message occurs when the map name specified on the command line is more than 32 bytes long in UTF-8 encoding. Also, this message can occur when there is an error in the command line such that text for another execution command is erroneously being interpreted as the map name. For example, in the command line below, the number representing the size of the echoed data is missing.

`mymap mmc -IE1S HereIsMyDataButIForgotToSpecifyTheSize -AED`

Because the size is missing, it is interpreted to be 0, such that there is no echoed data. The next string encountered on the command line

`(HereIsMyData...)`

Because it does not start with a hyphen (-), it is assumed to be the name of the next map to execute. Because the text is longer than 32 bytes in UTF-8 encoding, the *Invalid Map Name* message is returned.

Character set codes for CPACKAGE, CSERIESTOTEXT, and CTEXT

The second argument of the **CPACKAGE**, **CSERIESTOTEXT**, and **CTEXT** functions specifies the character set of the output of the function. The value of the second argument (the character set of the object content), must be a valid character set code.

All actions done that use the text string that results from one of these functions (**CPACKAGE**, **CSERIESTOTEXT**, or **CTEXT**) treat the text string as being of the specified character set—it is not automatically treated as Native.

Character set code	Data language
Native	Native
ASCII	ASCII (deprecated)
EBCDIC	EBCDIC (deprecated)
UNICODE_BE	UNICODE Big Endian (deprecated)
EUC	EUC (deprecated)
SJIS	SJIS (deprecated)
IBMKANJI	IBM Kanji (deprecated)
CIIKANJI	CII Kanji (deprecated)
JIS	JIS (deprecated)
UNICODE_LE	UNICODE Little Endian (deprecated)
UTF-8	UTF-8 (deprecated)
Latin1	Latin1 (deprecated)
UTF-8\ibm-1208	UTF-8
UTF-16	UTF-16
UTF-16BE	UTF-16 Big Endian
UTF-16LE	UTF-16 Little Endian
UTF-32	UTF-32
UTF-32BE	UTF-32 Big Endian
UTF-32LE	UTF-32 Little Endian
UTF16_PlatformEndian	UTF-16 Platform Endian
UTF16_OppositeEndian	UTF-16 Opposite Endian
UTF32_PlatformEndian	UTF-32 Platform Endian
UTF32_OppositeEndian	UTF-32 Opposite Endian

Character set code	Data language
UTF-16BE,version=1	UTF-16BE,version=1
UTF-16LE,version=1	UTF-16LE,version=1
UTF-16,version=1	UTF-16,version=1
UTF-16,version=2	UTF-16,version=2
UTF-7	UTF-7
IMAP-mailbox-name	IMAP-mailbox-name
SCSU	SCSU
BOCU-1	BOCU-1
CESU-8	CESU-8
ISO-8859-1	Latin1
US-ASCII	US-ASCII
gb18030	gb18030
ibm-912_P100-1995	Latin2
ibm-913_P100-2000	Latin3
ibm-914_P100-1995	CESU-8
ibm-915_P100-1995	Cyrillic
ibm-1089_P100-1995	Arabic
ibm-9005_X110-2007	Greek8
ibm-813_P100-1995	ibm-813
ibm-5012_P100-1999	Hebrew
ibm-916_P100-1995	ibm-916
ibm-920_P100-1995	Latin5
iso-8859_10-1998	Latin6
Iso-8859_11-2001	Thai8
ibm-921_P100-1995	ibm-921
iso-8859_14-1998	Latin8
ibm-923_P100-1998	Latin9
ibm-942_P12A-1999	shift_jis78
ibm-943_P15A-2003	MS_Kanji
ibm-943_P130-1999	Shift_JIS
ibm-33722_P12A-1999	EUC-JP
ibm-33722_P120-1999	ibm-33722_P120-1999
ibm-954_P101-2000	ibm-954_P101-2000
ibm-1373_P100-2002	ibm-1373_P100-2002
windows-950-2000	Big5
ibm-950_P110-1999	ibm-950_P110-1999
ibm-1375_P100-2003	Big5-HKSCS (IBM)
ibm-5471_P100-2006	ibm-5471_P100-2006
ibm-1386_P100-2001	ibm-1386_P100-2001
windows-936-2000	GBK
ibm-1383_P110-1999	EUC-CN
ibm-5478_P100-1995	GB_2312-80
ibm-964_P110-1999	EUC-TW
ibm-949_P110-1999	ibm-949_P110-1999
ibm-949_P11A-1999	ibm-949_P11A-1999
ibm-970_P110-1995	ibm-eucKR
ibm-971_P100-1995	ibm-971_P100-1995
ibm-1363_P11B-1998	ibm-1363 (korean)
ibm-1363_P110-1997	ibm-1363_P110-1997
windows-949-2000	Windows 949 (korean)
windows-874-2000	Windows 874
ibm-874_P100-1995	ibm-874
ibm-1162_P100-1999	ibm-1162
ibm-437_P100-1995	ibm-437
ibm-720_P100-1997	ibm-720
ibm-737_P100-1997	ibm-737
ibm-775_P100-1996	ibm-775
ibm-850_P100-1995	ibm-850
ibm-851_P100-1995	ibm-851
ibm-852_P100-1995	ibm-852
ibm-855_P100-1995	ibm-855
ibm-856_P100-1995	ibm-856
ibm-857_P100-1995	ibm-857
ibm-858_P100-1997	ibm-858
ibm-860_P100-1995	ibm-860
ibm-861_P100-1995	ibm-861
ibm-862_P100-1995	ibm-862

Character set code	Data language
ibm-863_P100-1995	ibm-863
ibm-864_X110-1999	ibm-864
ibm-865_P100-1995	ibm-865
ibm-866_P100-1995	ibm-866
ibm-867_P100-1998	ibm-867
ibm-868_P100-1995	ibm-868
ibm-869_P100-1995	ibm-869
ibm-878_P100-1996	KOI8-R
ibm-901_P100-1999	ibm-901
ibm-902_P100-1999	ibm-902
ibm-922_P100-1999	ibm-922
ibm-1168_P100-2002	KOI8-U
ibm-4909_P100-1999	ibm-4909
ibm-5346_P100-1998	ibm-5346
ibm-5347_P100-1998	ibm-5347
ibm-5348_P100-1997	ibm-5348
ibm-5349_P100-1998	ibm-5349
ibm-5350_P100-1998	ibm-5350
ibm-9447_P100-2002	ibm-9447
ibm-9448_X100-2005	ibm-9448
ibm-9449_P100-2002	ibm-9449
ibm-5354_P100-1998	ibm-5354
ibm-1250_P100-1995	ibm-1250
ibm-1251_P100-1995	ibm-1251
ibm-1252_P100-2000	ibm-1252
ibm-1253_P100-1995	ibm-1253
ibm-1254_P100-1995	ibm-1254
ibm-1255_P100-1995	ibm-1255
ibm-5351_P100-1998	ibm-5351
ibm-1256_P110-1997	ibm-1256
ibm-5352_P100-1998	ibm-5352
ibm-1257_P100-1995	ibm-1257
ibm-5353_P100-1998	ibm-5353
ibm-1258_P100-1997	ibm-1258
macos-0_2-10.2	macintosh
macos-6-10.2	x-mac-greek
macos-7_3-10.2	x-mac-cyrillic
macos-29-10.2	x-mac-ce
macos-35-10.2	x-mac-turkish
ibm-1051_P100-1995	hp-roman8
ibm-1276_P100-1995	ibm-1276 (Adobe Standard Encoding)
ibm-1006_P100-1995	ibm-1006
ibm-1098_P100-1995	ibm-1098
ibm-1124_P100-1996	ibm-1124
ibm-1125_P100-1997	ibm-1125
ibm-1129_P100-1997	ibm-1129
ibm-1131_P100-1997	ibm-1131
ibm-1133_P100-1997	ibm-1133
ISO_2022,locale=ja,version=0	ISO_2022,locale=ja,version=0
ISO_2022,locale=ja,version=1	ISO_2022,locale=ja,version=1 (JIS)
ISO_2022,locale=ja,version=2	ISO_2022,locale=ja,version=2
ISO_2022,locale=ja,version=3	ISO_2022,locale=ja,version=3 (JIS7)
ISO_2022,locale=ja,version=4	ISO_2022,locale=ja,version=4 (JIS8)
ISO_2022,locale=ko,version=0	ISO_2022,locale=ko,version=0
ISO_2022,locale=ko,version=1	ISO_2022,locale=ko,version=1
ISO_2022,locale=zh,version=0	ISO_2022,locale=zh,version=0
ISO_2022,locale=zh,version=1	ISO_2022,locale=zh,version=1
ISO_2022,locale=zh,version=2	ISO_2022,locale=zh,version=2
HZ	HZ-GB-2312
x11-compound-text	x11-compound-text
ISCII,version=0	x-iscii-de
ISCII,version=1	x-iscii-be
ISCII,version=2	x-iscii-pa
ISCII,version=3	x-iscii-gu
ISCII,version=4	x-iscii-or
ISCII,version=5	x-iscii-ta
ISCII,version=6	x-iscii-te

Character set code	Data language
ISCII,version=7	x-iscii-ka
ISCII,version=8	x-iscii-ma
LMBCS-1	LMBCS-1
ibm-37_P100-1995	ibm-037 (ebcdic-cp-us/ca/wt/nl)
ibm-273_P100-1995	ebcdic-de
ibm-277_P100-1995	EBCDIC-CP-DK/NO
ibm-278_P100-1995	ebcdic-cp-fi/se/sv
ibm-280_P100-1995	ebcdic-cp-it
ibm-284_P100-1995	ebcdic-cp-es
ibm-285_P100-1995	ebcdic-cp-gb
ibm-290_P100-1995	EBCDIC-JP-kana
ibm-297_P100-1995	ebcdic-cp-fr
ibm-420_X120-1999	ebcdic-cp-ar1
ibm-424_P100-1995	ebcdic-cp-he
ibm-500_P100-1995	ebcdic-cp-be/ch
ibm-803_P100-1999	ibm-803
ibm-838_P100-1995	IBM-Thai
ibm-870_P100-1995	ebcdic-cp-roece/yu
ibm-871_P100-1995	ebcdic-is
ibm-875_P100-1995	ibm-875
ibm-918_P100-1995	ebcdic-cp-ar2
ibm-930_P120-1999	ibm-930
ibm-933_P110-1995	ibm-933
ibm-935_P110-1999	ibm-935
ibm-937_P110-1999	ibm-937
ibm-939_P120-1999	ibm-939
ibm-1025_P100-1995	ibm-1025
ibm-1026_P100-1995	ibm-1026
ibm-1047_P100-1995	ibm-1047
ibm-1097_P100-1995	ibm-1097
ibm-1112_P100-1995	ibm-1112
ibm-1122_P100-1999	ibm-1122
ibm-1123_P100-1995	ibm-1123
ibm-1130_P100-1997	ibm-1130
ibm-1132_P100-1998	ibm-1132
ibm-1137_P100-1999	ibm-1137
ibm-4517_P100-2005	ibm-4517
ibm-1140_P100-1997	ibm-1140 (ebcdic-us-37+euro)
ibm-1141_P100-1997	ibm-1141 (ebcdic-de-273+euro)
ibm-1142_P100-1997	ibm-1142 (ebcdic-dk/no-277+euro)
ibm-1143_P100-1997	ibm-1143 (ebcdic-fi/se-278+euro)
ibm-1144_P100-1997	ibm-1144 (ebcdic-it-280+euro)
ibm-1145_P100-1997	ibm-1145 (ebcdic-es-284+euro)
ibm-1146_P100-1997	ibm-1146 (ebcdic-gb-285+euro)
ibm-1147_P100-1997	ibm-1147 (ebcdic-fr-297+euro)
ibm-1148_P100-1997	ibm-1148 (ebcdic-international+euro)
ibm-1149_P100-1997	ibm-1149 (ebcdic-is-871+euro)
ibm-1153_P100-1999	ibm-1153
ibm-1154_P100-1999	ibm-1154
ibm-1155_P100-1999	ibm-1155
ibm-1156_P100-1999	ibm-1156
ibm-1157_P100-1999	ibm-1157
ibm-1158_P100-1999	ibm-1158
ibm-1160_P100-1999	ibm-1160
ibm-1164_P100-1999	ibm-1164
ibm-1364_P110-1997	ibm-1364
ibm-1371_P100-1999	ibm-1371
ibm-1388_P103-2001	ibm-1388
ibm-1390_P110-2003	ibm-1390
ibm-1399_P110-2003	ibm-1399
ibm-5123_P100-1999	ibm-5123
ibm-8482_P100-1999	ibm-8482
ibm-16684_P110-2003	ibm-16684
ibm-4899_P100-1998	ibm-4899
ibm-4971_P100-1999	ibm-4971
ibm-9067_X100-2005	ibm-9067
ibm-12712_P100-1998	ebcdic-he

Character set code	Data language
ibm-16804_X110-1999	ebcdic-ar
ibm-37_P100-1995,swaplfnl	ibm-37-s390
ibm-1047_P100-1995,swaplfnl	ibm-1047-s390
ibm-1140_P100-1997,swaplfnl	ibm-1140-s390
ibm-1141_P100-1997,swaplfnl	ibm-1141-s390
ibm-1142_P100-1997,swaplfnl	ibm-1142-s390
ibm-1143_P100-1997,swaplfnl	ibm-1143-s390
ibm-1144_P100-1997,swaplfnl	ibm-1144-s390
ibm-1145_P100-1997,swaplfnl	ibm-1145-s390
ibm-1146_P100-1997,swaplfnl	ibm-1146-s390
ibm-1147_P100-1997,swaplfnl	ibm-1147-s390
ibm-1148_P100-1997,swaplfnl	ibm-1148-s390
ibm-1149_P100-1997,swaplfnl	ibm-1149-s390
ibm-1153_P100-1999,swaplfnl	ibm-1153-s390
ibm-12712_P100-1998,swaplfnl	ibm-12712-s390
ibm-16804_X110-1999,swaplfnl	ibm-16804-s390
ebcdic-xml-us	ebcdic-xml-us

Sterling B2B Integrator in Design Server

Sterling B2B Integrator functions are available for you to use from the Design Server.

- [Sterling B2B Integrator in Design Server overview](#)
The Design Server provides the functionality to create and test maps that you can run on a Sterling B2B Integrator server and deploy to a Sterling B2B Integrator server map repository.
- [Supported Transformation Extender map functionality on Sterling B2B Integrator](#)
Sterling B2B Integrator supports all mapping functions in IBM Transformation Extender maps.
- [Differences when running Transformation Extender maps on Sterling B2B Integrator](#)
On Sterling B2B Integrator, IBM Transformation Extender maps run through different services that are called by business processes.
- [Getting started with Sterling B2B Integrator](#)
To get started using the Design Server with the Sterling B2B Integrator runtime, use the following high-level task list in a preproduction phase.
- [Adding a Sterling B2B Integrator server](#)
- [Configuring the Check In Map service options](#)
To run the services on a Sterling B2B Integrator server, you must configure the Check In Map service options in the Design Server.
- [Building Transformation Extender map](#)
Go to Design>Maps tab to use the build and run on process to generate a compiled map to use on a Sterling B2B Integrator server.
- [Deploying Transformation Extender maps](#)
After you tested the running of IBM Transformation Extender maps in the Design Server, you can deploy them from Design Server to a Sterling B2B Integrator server map repository.

Sterling B2B Integrator in Design Server overview

The Design Server provides the functionality to create and test maps that you can run on a Sterling B2B Integrator server and deploy to a Sterling B2B Integrator server map repository.

The Sterling B2B Integrator functions can be used for various processes such as validation, running as a step in a Business Process, translation, or transformation, and for generating reports.

To read about the benefits of using Sterling B2B Integrator, see the product page on the IBM website: <http://www.ibm.com/software/commerce>.

See additional information in the IBM Transformation Extender for Sterling B2B Integrator documentation.

Supported Transformation Extender map functionality on Sterling B2B Integrator

Sterling B2B Integrator supports all mapping functions in IBM Transformation Extender maps.

All adapters are available on the target platform and specified in the map and card rules.

In addition to IBM Transformation Extender mapping functions, there are extensions to IBM Transformation Extender maps with the Sterling adapter so that the map can share information with the Sterling B2B Integrator services and other Sterling B2B Integrator processes. The adapter calls the data harness to provide access to Sterling B2B Integrator tables for maps to read data from, and write data to, these tables. Through calls to the data harness from the Sterling adapter in map rules, maps can store information about the data for tracking purposes and retrieve information from these tables.

If the maps use **GET** and **PUT** calls to the Sterling adapter to get and set certain values that are maintained in the Sterling B2B Integrator database tables, those maps can only run within the Sterling B2B Integrator environment.

Differences when running Transformation Extender maps on Sterling B2B Integrator

On Sterling B2B Integrator, IBM Transformation Extender maps run through different services that are called by business processes.

There is a set of Sterling B2B Integrator services that run IBM Transformation Extender maps with some restrictions. There is also a WTX Map service that can run any IBM Transformation Extender map with no restrictions. Through the Design Server, you can test running maps on a Sterling B2B Integrator server inside this WTX Map service.

Although map functions and all adapters are supported by Sterling B2B Integrator, running IBM Transformation Extender maps on Sterling B2B Integrator is different in the following ways:

- For maps to run inside Sterling B2B Integrator services other than the WTX Map service, these services require that maps meet certain requirements to be considered "[well behaved](#)" maps.
- If the map is not checked into the Sterling B2B Integrator map repository, the map runs in the same way as any other IBM Transformation Extender map runs.
- The map can include the Sterling adapter in **PUT** and **GET** rules in the following locations:
 - On the map
 - Through the command line on an input or output card

The Sterling adapter calls the [data harness](#) to provide access to Sterling B2B Integrator tables. These maps can only run within the Sterling B2B Integrator.

- You must configure the Design Server settings, such as connection properties, the Check In Map service options, and the troubleshooting options, for Sterling B2B Integrator services.
- The map must be built for a platform that the Sterling B2B Integrator server supports.
- For troubleshooting problems, you must configure the Save service request and response, Save service session log messages options in the Design Server.

You can test the map in the Design Server to verify that you configured the map to run correctly on Sterling B2B Integrator. After you completed test running the map, you can then [deploy](#) the map from Design Server to a Sterling B2B Integrator server map repository.

- [Well-behaved map](#)

There are several Sterling B2B Integrator services that can run IBM Transformation Extender maps. When a IBM Transformation Extender map runs from those services, except of the WTX Map service, it must operate according to a certain set of requirements. A map that meets these requirements is well-behaved map.

- [Data harness](#)

IBM Transformation Extender maps share information with the Sterling B2B Integrator services and other Sterling B2B Integrator processes through the data harness interface.

Well-behaved map

There are several Sterling B2B Integrator services that can run IBM Transformation Extender maps. When a IBM Transformation Extender map runs from those services, except of the WTX Map service, it must operate according to a certain set of requirements. A map that meets these requirements is well-behaved map.

A well-behaved map has the following characteristics:

- The map is an executable IBM Transformation Extender map. If at runtime the map is provided with the required input, it can be run.
- The map is designed to be called from within a specific Sterling B2B Integrator service. If the service is set up to process incoming X12 data, for example, the first input card of the map accepts X12 data. If the service is expecting the map to generate an XML message, then the last output card of the map creates an XML message.
- You define the restart and burst settings in the map as the service requires, and as stipulated in the Sterling B2B Integrator exhaust input setting.
- If the map requires table data information or settings from Sterling B2B Integrator, such as delimiter settings for outbound EDI, or extract information while parsing the data that is to be sent back to Sterling B2B Integrator, such as correlations, then the map has been designed to meet these requirements.
- The map design follows published Sterling B2B Integrator and IBM Transformation Extender integration best practices including being a thread-safe map. In Sterling B2B Integrator, Sterling B2B Integrator maps are run in a multi-threaded environment.
- The service passes the Primary Document to the first input card and reads it from the last output card. For more information about the Primary Document, see the IBM Transformation Extender for Sterling B2B Integrator documentation.

Data harness

IBM Transformation Extender maps share information with the Sterling B2B Integrator services and other Sterling B2B Integrator processes through the data harness interface.

The data harness uses the basic service framework, or harness model, that Sterling B2B Integrator uses to view all services in the same way. It is invoked by the IBM Transformation Extender map through the Sterling adapter to provide access to database tables.

IBM Transformation Extender maps can read information, such as trading partner information or code list table data, from Sterling B2B Integrator tables. They can also write information about the data, such as a document ID or other key field, that Sterling B2B Integrator can use to track the data.

Some of the examples that are included in the IBM Transformation Extender for Sterling B2B Integrator use the data harness to set values in Sterling B2B Integrator database tables. These values and any other values that maps set or read are included in the map description.

Access the data harness through maps with **PUT** and **GET** rules in the following locations:

- On the map
- Through the Sterling adapter command line on an input or output card

For details about accessing the data harness from maps, see the IBM Transformation Extender for Sterling B2B Integrator documentation.

For more information about the Sterling B2B Integrator harness model and how it operates with external systems, see the IBM Sterling B2B Integrator documentation that you can access through the IBM Sterling B2B Integrator Support website (<http://www.ibm.com/software/commerce/support>).

Getting started with Sterling B2B Integrator

To get started using the Design Server with the Sterling B2B Integrator runtime, use the following high-level task list in a preproduction phase.

- Add a Sterling B2B Integrator server in the Design Server.
- Test run the IBM Transformation Extender map.
- Deploy the IBM Transformation Extender maps to a Sterling B2B Integrator server map repository.

Adding a Sterling B2B Integrator server

Specify the Design Server settings for the connection properties in the Sterling B2B Integrator services. You must enter values for the Sterling B2B Integrator server name, port number for the Sterling B2B Integrator Dashboard, and the user name and password.

1. Login to Design Server.
2. Go to Deploy and select Servers from the list.
The Servers page opens.
3. Click on Add button.
An Add a Server window opens.
4. In the Add a Server window, do the following:
 - a. In the Name field, type the name of the server.
 - b. In the Description field, add the description of the server.
 - c. From the Type drop down list, select Sterling B2B Integrator.
 - d. From the Platform drop-down list, select the required platform.
 - e. Enter the host name or IP address of the Sterling B2B Integrator in the Login server field.
 - f. Enter the port number of the Dashboard Port field.
 - g. Enter the User name and Password.
5. Click on Test button to test the connection with the Sterling B2B Integrator server.
You will get the **Successfully Connected** message when the server is up and running.
6. Click on Add button to add the Sterling B2B Integrator server in the Design Server.
You can see the added server in the Servers page.

- [Configuring an HTTPS connection to the Sterling B2B Integrator server](#)
You can configure a secure connection to the Sterling B2B Integrator server.

Configuring an HTTPS connection to the Sterling B2B Integrator server

You can configure a secure connection to the Sterling B2B Integrator server.

The Design Server establishes a HyperText Transfer Protocol (HTTP) connection with a Sterling B2B Integrator server, which is the default connection type, using the server and Sterling B2B Integrator Dashboard port number connection properties specified in the options for IBM Transformation Extender maps.

To configure a secure connection with the Sterling B2B Integrator server using HTTPS in the Design Server:

1. Go to Design Server...Deploy...Servers.
2. Select the added Sterling B2B Integrator server from the Servers page and click on Edit button to open the Edit server window.
3. Enter the port number in the Dashboard port field.
If you configured the Map Test HTTP Server Adapter for a secure connection, use that same port number for the Dashboard port field.
4. Select the Secure Transport check box.
5. To specify direct trust, select the Trust all hosts check box.
At run time when the map run or deploy operations are attempting to establish a connection to a Sterling B2B Integrator server, the following results occur depending on how Trust all hosts is set:
 - If Trust all hosts is selected, the connection verification step that checks the host passes even if the machine host name does not match the host name in the certificate.
 - If Trust all hosts is cleared, the connection verification step that checks the host fails if the machine host name does not match the host name in the certificate.
6. Click Save to save changes.

Configuring the Check In Map service options

To run the services on a Sterling B2B Integrator server, you must configure the Check In Map service options in the Design Server.

You can change Design Server settings pertaining to the Check In Map service options for IBM Transformation Extender maps in the Sterling B2B Integrator service, under Check In Map Service.

1. Go to Design Server...Deploy...Servers.
2. Select the added Sterling B2B Integrator server from the Servers page and click on Edit button to open the Edit server window.
3. Select the options under Check In Map service.
4. Select one of the following choices about which version of the map you want to specify as the default in the Sterling B2B Integrator map repository:
 - To make the version of the map that you deploy from the Design Server the default map in the Sterling B2B Integrator map repository, select the Make checked in map the default check box.

- To keep the version of the same map that is already in the Sterling B2B Integrator map repository, as the default map, clear the Make checked in map the default check box.
5. Select one of the following choices about whether you want the service to check if the map is thread-safe to run on the Sterling B2B Integrator server:
 - If you do not want the Check In Map service to check if the map is thread-safe to run on the Sterling B2B Integrator server, select the Ignore map thread-safe check check box.
 - If you do want the Check In Map service to check if the map is thread-safe to run on the Sterling B2B Integrator server, clear the Ignore map thread-safe check check box.
 6. Click Save to save changes.
- **[Configuring troubleshooting options](#)**
- To help you troubleshoot problems when test running IBM Transformation Extender maps on a Sterling B2B Integrator server and deploying the maps to a Sterling B2B Integrator server map repository, you must configure the Service request and response messages option in the Design Server.
-

Configuring troubleshooting options

To help you troubleshoot problems when test running IBM Transformation Extender maps on a Sterling B2B Integrator server and deploying the maps to a Sterling B2B Integrator server map repository, you must configure the Service request and response messages option in the Design Server.

You can change Design Server settings pertaining to the Service request and response messages option for IBM Transformation Extender maps in the Sterling B2B Integrator preferences, under Save Messages.

To configure troubleshooting options:

1. Select the added Sterling B2B Integrator server from the Servers page and click on Edit button to open the Add a server window.
2. In the Add a server window, under Save Messages, select the Service request and response messages check box to save request and response messages. These files are generated when you run a map from the Design Server with the Test Map service, or deploy a map from the Design Server to the Sterling B2B Integrator map repository with the Check In Map service.
3. Select Service session log check box to see the session log. These files are generated when you run or deploy a map.
You can see the files in Design>Files.
4. Click Save to save changes.

Building Transformation Extender map

Go to Design>Maps tab to use the build and run on process to generate a compiled map to use on a Sterling B2B Integrator server.

You can build a map for the Sterling B2B Integrator server using Build button.

You can use Run On button to select the available Sterling B2B Integrator server to run a map.

After a successful build, you can test the map on the Sterling B2B Integrator server. When you run on Sterling B2B Integrator server the IBM Transformation Extender map, the compiled map, and input files are sent to the Sterling B2B Integrator for processing. The map audit, map trace and results are returned to the Design Server.

Deploying Transformation Extender maps

After you tested the running of IBM Transformation Extender maps in the Design Server, you can deploy them from Design Server to a Sterling B2B Integrator server map repository.

To deploy a IBM Transformation Extender map or maps to a Sterling B2B Integrator server map repository:

1. Configure the Design Server connection properties for IBM Transformation Extender maps deployed on a Sterling B2B Integrator server.
 2. Configure the Design Server Check In Map service options for IBM Transformation Extender maps deployed on a Sterling B2B Integrator server.
 3. Make the selections to deploy your map on a Sterling B2B Integrator server in either one of the two following ways:
 - Deploy a single map.
 - Deploy multiple maps or projects by creating a package.
You can select multiple compiled IBM Transformation Extender maps to deploy at the same time by selecting the created packages.
 4. Click Deploy button from the Packages page.
The Deploy window opens.
 5. Select one or more packages to deploy and one or more Sterling B2B Integrator Servers to deploy to.
 6. Click Deploy.
This starts the map deploy operation. Once done, you get the message **Deployed Successfully** in the Deploy window.
 7. Click View Log to see the detailed information of deploy operation.
The **View deploy log** is generated with detailed information.
- **[Making the checked-in map the default](#)**
Select the Make checked in map default check box to make the version of the map that you deploy from the Design Server the default map in the Sterling B2B Integrator map repository.

Making the checked-in map the default

Select the Make checked in map default check box to make the version of the map that you deploy from the Design Server the default map in the Sterling B2B Integrator map repository.

If there already is a version of the same map in the Sterling B2B Integrator map repository that is the default, this version of the map is made the default map instead.

When this version of the map is made the default map in the Sterling B2B Integrator map repository, this is the version that Sterling B2B Integrator uses for various processes such as validation, running as a step in a Business Process, translation or transformation, and for generating reports.

To keep the version of the same map that is already in the Sterling B2B Integrator map repository, as the default map, clear the Make checked in map default check box.

Sterling B2B Integrator in the Design Studio

Sterling B2B Integrator functions are available for you to use from the Design Studio.

- [Sterling B2B Integrator in Design Studio overview](#)

Sterling B2B Integrator functions are available for you to use from the Design Studio.

- [Getting started with Sterling B2B Integrator](#)

To get started using the Design Studio with the Sterling B2B Integrator runtime option, use the following high-level task list in a preproduction phase.

- [Configuring Design Studio for Sterling B2B Integrator](#)

You have to configure the Design Studio settings for Sterling B2B Integrator services.

- [Building Transformation Extender maps](#)

Use the build process in the Map Designer to generate a compiled map to use on a Sterling B2B Integrator server.

- [Testing Transformation Extender maps](#)

You can test running an IBM Transformation Extender map on a Sterling B2B Integrator server.

- [Deploying Transformation Extender maps](#)

After you tested the running of IBM Transformation Extender maps in the Design Studio, you can deploy them from Design Studio to a Sterling B2B Integrator server map repository.

- [Troubleshooting Transformation Extender maps](#)

Sterling B2B Integrator in Design Studio overview

Sterling B2B Integrator functions are available for you to use from the Design Studio.

The Design Studio provides the functionality to create and test maps that you can run on a Sterling B2B Integrator server, and deploy to a Sterling B2B Integrator server map repository.

The Sterling B2B Integrator functions can be used for various processes such as validation, running as a step in a Business Process, translation or transformation, and for generating reports.

You need to install IBM Transformation Extender for Integration Servers on both the machine where the Design Studio is installed, and on the machine where the Sterling B2B Integrator server is installed. For specific product installation and configuration information, see the [release notes](#).

To read about the benefits of using Sterling B2B Integrator, see the product page on the IBM website: <http://www.ibm.com/software/commerce>.

See additional information in the IBM Transformation Extender for Sterling B2B Integrator documentation.

- [Supported Transformation Extender map functionality on Sterling B2B Integrator](#)

Sterling B2B Integrator supports all mapping functions in IBM Transformation Extender maps.

- [Differences when running Transformation Extender maps on Sterling B2B Integrator](#)

On Sterling B2B Integrator, IBM Transformation Extender maps run through different services that are called by business processes.

- [Well-behaved map](#)

There are several Sterling B2B Integrator services that can run IBM Transformation Extender maps. When a IBM Transformation Extender map runs from those services, with the exception of the WTX Map service, it must operate according to a certain set of requirements. A map that meets these requirements is considered to be well-behaved.

- [Data harness](#)

IBM Transformation Extender maps share information with the Sterling B2B Integrator services and other Sterling B2B Integrator processes through the data harness interface.

Supported Transformation Extender map functionality on Sterling B2B Integrator

Sterling B2B Integrator supports all mapping functions in IBM Transformation Extender maps.

All adapters are available on the target platform and specified in the map and card rules.

In addition to IBM Transformation Extender mapping functions, there are extensions to IBM Transformation Extender maps with the Sterling adapter so that the map can share information with the Sterling B2B Integrator services and other Sterling B2B Integrator processes. The adapter calls the data harness to provide access to Sterling B2B Integrator tables for maps to read data from, and write data to, these tables. Through calls to the data harness from the Sterling adapter in map rules, maps can store information about the data for tracking purposes and retrieve information from these tables.

If the maps use **GET** and **PUT** calls to the Sterling adapter to get and set certain values that are maintained in the Sterling B2B Integrator database tables, those maps can only run within the Sterling B2B Integrator environment.

Differences when running Transformation Extender maps on Sterling B2B Integrator

On Sterling B2B Integrator, IBM Transformation Extender maps run through different services that are called by business processes.

There is a set of Sterling B2B Integrator services that run IBM Transformation Extender maps with some restrictions. There is also a WTX Map service that can run any IBM Transformation Extender map with no restrictions. Through the Design Studio, you can test running maps on a Sterling B2B Integrator server inside this WTX Map service.

Although map functions and all adapters are supported by Sterling B2B Integrator, running IBM Transformation Extender maps on Sterling B2B Integrator is different in the following ways:

- For maps to run inside Sterling B2B Integrator services other than the WTX Map service, these services require that maps meet certain requirements to be considered "[well behaved](#)" maps.
- If the map is checked into the Sterling B2B Integrator map repository, the IBM Transformation Extender map working directory is *SI_install_dir\logs*.
 - If a map uses the **RUN** function in a rule, the map that is referenced must be located either in the Sterling B2B Integrator map repository, or relative to the parent map location on the server.
 - If the Schema *>_Type_>* Metadata card setting specifies an XML schema, the schema should be checked into the Sterling B2B Integrator schema repository. However, if the schema is not in the repository, it should be placed relative to the *SI_install_dir\logs* directory.

All IBM Transformation Extender maps, including validation and translation maps, and maps run through the WTX Map service, first attempt to load the schema from the repository. If the schema is not in the repository, the maps then attempt to load the schema from the map location on the server.

If the map is not checked into the Sterling B2B Integrator map repository, the map runs in the same way as any other IBM Transformation Extender map runs.

- The map can include the Sterling adapter in **PUT** and **GET** rules in the following locations:
 - on the map
 - on component rules in schemas
 - through the command line on an input or output card

The Sterling adapter calls the [data harness](#) to provide access to Sterling B2B Integrator tables. These maps can only run within the Sterling B2B Integrator environment.

To see how to use the Sterling adapter in a map rule to access Sterling B2B Integrator tables, see the translation example in the *install_dir\examples\integrations\SterlingB2B\Engine* directory, where *install_dir* refers to the path where IBM Transformation Extender is installed.

See [Supported Transformation Extender map functionality on Sterling B2B Integrator](#).

- You must [configure](#) the Design Studio settings, such as connection properties, the Check In Map service options, and the troubleshooting options, for Sterling B2B Integrator services.

To test that IBM Transformation Extender for Sterling B2B Integrator is installed correctly, and that Design Studio and Sterling B2B Integrator are configured and are running properly, run the Installation Verification Test (IVT) example under *install_dir\examples\integrations\SterlingB2B*.

- The map must be built on a platform that the Sterling B2B Integrator server supports.
- When you run maps from the Design Studio on the Sterling B2B Integrator server, the map server location is the *install_dir\logs* directory.
- For troubleshooting problems, you must [configure](#) the Save service request and response messages option in the Design Studio.

You can [test](#) the map in the Design Studio to verify that you configured the map to run correctly on Sterling B2B Integrator. After you completed test running the map, you can then [deploy](#) the map from Design Studio to a Sterling B2B Integrator server map repository.

Well-behaved map

There are several Sterling B2B Integrator services that can run IBM Transformation Extender maps. When a IBM Transformation Extender map runs from those services, with the exception of the WTX Map service, it must operate according to a certain set of requirements. A map that meets these requirements is considered to be well-behaved.

A well-behaved map has the following characteristics:

- The map is an executable IBM Transformation Extender map. If at run time the map is provided with the required input, it can be run.
- The map is designed to be called from within a specific Sterling B2B Integrator service. If the service is set up to process incoming X12 data, for example, the first input card of the map accepts X12 data. If the service is expecting the map to generate an XML message, then the last output card of the map creates an XML message.
- You define the restart and burst settings in the map as the service requires, and as stipulated in the Sterling B2B Integrator exhaust input setting.
- If the map requires table data information or settings from Sterling B2B Integrator, such as delimiter settings for outbound EDI, or extract information while parsing the data that is to be sent back to Sterling B2B Integrator, such as correlations, then the map has been designed to meet these requirements.
- The map design follows published Sterling B2B Integrator and IBM Transformation Extender integration best practices including being a thread-safe map. In Sterling B2B Integrator, Sterling B2B Integrator maps are run in a multi-threaded environment.
- The service passes the Primary Document to the first input card and reads it from the last output card. For more information about the Primary Document, see the IBM Transformation Extender for Sterling B2B Integrator documentation.

Data harness

IBM Transformation Extender maps share information with the Sterling B2B Integrator services and other Sterling B2B Integrator processes through the data harness interface.

The data harness uses the basic service framework, or harness model, that Sterling B2B Integrator uses to view all services in the same way. It is invoked by the IBM Transformation Extender map through the Sterling adapter to provide access to database tables.

IBM Transformation Extender maps can read information, such as trading partner information or code list table data, from Sterling B2B Integrator tables. They can also write information about the data, such as a document ID or other key field, that Sterling B2B Integrator can use to track the data.

Some of the examples that are included in the IBM Transformation Extender for Sterling B2B Integrator use the data harness to set values in Sterling B2B Integrator database tables. These values and any other values that maps set or read are included in the map description.

Access the data harness through maps with **PUT** and **GET** rules in the following locations:

- on the map
- on component rules in schemas
- through the Sterling adapter command line on an input or output card

For details about accessing the data harness from maps, see the IBM Transformation Extender for Sterling B2B Integrator documentation.

For more information about the Sterling B2B Integrator harness model and how it operates with external systems, see the IBM Sterling B2B Integrator documentation that you can access through the IBM Sterling B2B Integrator Support website (<http://www.ibm.com/software/commerce/support>).

Getting started with Sterling B2B Integrator

To get started using the Design Studio with the Sterling B2B Integrator runtime option, use the following high-level task list in a preproduction phase.

- [Configure](#) Design Studio for Sterling B2B Integrator.
- [Build](#) the IBM Transformation Extender maps in the Design Studio.
- [Test](#) run the IBM Transformation Extender maps from the Design Studio to a Sterling B2B Integrator server.
- [Deploy](#) the IBM Transformation Extender maps to a Sterling B2B Integrator server map repository.
When your maps are ready for production, after you deploy the maps to a Sterling B2B Integrator server map repository, there are additional tasks to complete using the Sterling B2B Integrator Graphical Process Modeler (GPM) user interface. For information about running maps in a production environment, see the [IBM Sterling B2B Integrator documentation](#).

Configuring Design Studio for Sterling B2B Integrator

You have to configure the Design Studio settings for Sterling B2B Integrator services.

Configure the Test Map service and the Check In Map service to test run IBM Transformation Extender maps on a Sterling B2B Integrator server, and deploy the maps to a Sterling B2B Integrator map repository. In the Sterling B2B Integrator preferences pane, configure the connection properties that are under Connectivity, and the options that are under Check In Map Service and Save Messages.

- [Configuring the connection properties](#)
To test run a IBM Transformation Extender map on a Sterling B2B Integrator server and deploy it to a Sterling B2B Integrator map repository, you must configure the connectivity properties for the services in the Design Studio.
- [Configuring the Check In Map service options](#)
To run the services on a Sterling B2B Integrator server, you must configure the Check In Map service options in the Design Studio.
- [Configuring troubleshooting options](#)
To help you troubleshoot problems when test running IBM Transformation Extender maps on a Sterling B2B Integrator server, and deploying the maps to a Sterling B2B Integrator server map repository, you must configure the Save service request and response messages option in the Design Studio.

Configuring the connection properties

To test run a IBM Transformation Extender map on a Sterling B2B Integrator server and deploy it to a Sterling B2B Integrator map repository, you must configure the connectivity properties for the services in the Design Studio.

Specify the Design Studio settings for the connection properties in the Sterling B2B Integrator preferences under Connectivity. You must enter values for the Sterling B2B Integrator server name, port number for the Sterling B2B Integrator Dashboard, and the user name and password Dashboard authentication information; these are required fields. The connection is established using HyperText Transport Protocol (HTTP), which is the default connection type, or HyperText Transport Protocol Secure (HTTPS).

To configure a connection to the server in the Design Studio:

1. In the Design Studio, click Window > Preferences > Transformation Extender > Map > Sterling B2B Integrator.
The Sterling B2B Integrator preferences pane opens. Enter values and select options under the Connectivity section.
 2. Select one of the following options to specify how you want to set the connection properties for the services:
 - To specify that the same set of connection properties be used for the Sterling B2B Integrator services, select the Use the same server connection properties for 'Test Map' and 'Check In Map' services check box. The preferences pane displays the Connection Properties tab.
 - To specify that a different set of connection properties be used for the Sterling B2B Integrator services, clear the Use the same server connection properties for 'Test Map' and 'Check In Map' services check box. The preferences pane displays the Test Map Service and the Check In Map Service tabs.
- Under the tab that the preference pane displays based on your selection, enter values in the fields and select the required options for the connection properties.
3. Enter the host name or IP address of the Sterling B2B Integrator server in the Login server field.
 4. Enter a number for the Dashboard port field.
If you are using HTTPS, see the topic about [configuring an HTTPS connection](#).
 5. Enter the name of the user in the User name field.

6. Enter the password in the Password field.
7. Click OK to save changes.

- **Configuring an HTTPS connection to the Sterling B2B Integrator server**

You can configure a secure connection to the Sterling B2B Integrator server.

You can configure a secure connection to the Sterling B2B Integrator server.

The Design Studio establishes a HyperText Transfer Protocol (HTTP) connection with a Sterling B2B Integrator server, which is the default connection type, using the server and Sterling B2B Integrator Dashboard port number connection properties specified in the options for IBM Transformation Extender maps.

However, to have a secure connection to the Sterling B2B Integrator server, you can use HyperText Transfer Protocol Secure (HTTPS) access through the Secure Socket Layer (SSL) with the Test Map and Check In Map services. You must configure the Map Test HTTP Server Adapter using SSL in the Sterling B2B Integrator Dashboard. To read about how to do this, see the topic about configuring the HTTP Server adapter in the IBM Sterling B2B Integrator documentation that you can access through the IBM Sterling B2B Integrator Support website (<http://www.ibm.com/software/commerce/support>).

There are additional options under the Connectivity section of the Sterling B2B Integrator server preferences pane in the Design Studio where you specify a secure connection.

To configure a secure connection with the Sterling B2B Integrator server using HTTPS in the Design Studio:

1. Enter the port number in the Dashboard port field.

If you configured the Map Test HTTP Server Adapter for a secure connection, use that same port number for the Dashboard port field.

2. Select the Secure Transport check box.

3. To specify direct trust, select the Trust all hosts check box.

At run time when the map run or deploy operations are attempting to establish a connection to a Sterling B2B Integrator server, the following results occur depending on how Trust all hosts is set:

- If Trust all hosts is selected, the connection verification step that checks the host passes even if the machine host name does not match the host name in the certificate.
- If Trust all hosts is cleared, the connection verification step that checks the host fails if the machine host name does not match the host name in the certificate.

4. Click OK to save changes.

Configuring the Check In Map service options

To run the services on a Sterling B2B Integrator server, you must configure the Check In Map service options in the Design Studio.

You can change Map Designer settings pertaining to the Check In Map service options for IBM Transformation Extender maps in the Sterling B2B Integrator preferences, under Check In Map Service.

To configure the options for running the services:

1. In the Design Studio, click Window > Preferences > Transformation Extender > Map > Sterling B2B Integrator.

The Sterling B2B Integrator preferences pane opens. Select the options under Check In Map Service.

2. Select one of the following choices about which version of the map you want to specify as the default in the Sterling B2B Integrator map repository:

- To make the version of the map that you deploy from the Design Studio the default map in the Sterling B2B Integrator map repository, select the Make checked in map the default check box.
- To keep the version of the same map that is already in the Sterling B2B Integrator map repository, as the default map, clear the Make checked in map the default check box.

3. Select one of the following choices about whether or not you want the service to check if the map is thread-safe to run on the Sterling B2B Integrator server:

- If you do not want the Check In Map service to check if the map is thread-safe to run on the Sterling B2B Integrator server, select the Ignore map thread-safe check check box.
- If you do want the Check In Map service to check if the map is thread-safe to run on the Sterling B2B Integrator server, clear the Ignore map thread-safe check check box.

4. Click OK to save changes.

Configuring troubleshooting options

To help you troubleshoot problems when test running IBM Transformation Extender maps on a Sterling B2B Integrator server, and deploying the maps to a Sterling B2B Integrator server map repository, you must configure the Save service request and response messages option in the Design Studio.

You can change Design Studio settings pertaining to the Save service request and response messages option for IBM Transformation Extender maps in the Sterling B2B Integrator preferences, under Save Messages.

To configure troubleshooting options:

1. In the Design Studio, click Window > Preferences > Transformation Extender > Map > Sterling B2B Integrator.

The Sterling B2B Integrator preferences pane opens.

2. Under Save Messages, select the Save service request and response messages check box to save request and response messages. These files are generated when you run a map from the Design Studio with the Test Map service, or deploy a map from the Design Studio to the Sterling B2B Integrator map repository with the Check In Map service.

3. Click OK to save changes.

Building Transformation Extender maps

Use the build process in the Map Designer to generate a compiled map to use on a Sterling B2B Integrator server.

The build process generates a compiled map with a file name extension that is specific to the target platform of the server, which you specified in the Sterling B2B Integrator preferences window.

You can test run a map from the Design Studio or deploy a map from the Design Studio to the Sterling B2B Integrator server in either the Outline view or Composition view. When you select the map in either of these views, the test run map and deploy map functions automatically build the IBM Transformation Extender map for the target platform of the server, which you specified in the Sterling B2B Integrator preferences window.

You can also deploy a map from the Design Studio to the Sterling B2B Integrator server in either the Navigator view or the Extender Navigator view. When you select an executable map in either of these views, you must first manually build the IBM Transformation Extender map for the target platform of the server by selecting one of the build map options.

After a successful build, you can test the map on the Sterling B2B Integrator server. When you run a IBM Transformation Extender map, the compiled map, and input files are sent to the Sterling B2B Integrator for processing. The results are returned to the Design Studio.

This is a pre-production phase. When your maps are ready for production, after you deployed the maps to a Sterling B2B Integrator server map repository, there are additional tasks to complete using the Sterling B2B Integrator Graphical Process Modeler (GPM) user interface. For information about running maps in a production environment, see the [IBM Sterling B2B Integrator documentation](#).

Testing Transformation Extender maps

You can test running an IBM Transformation Extender map on a Sterling B2B Integrator server.

When you run an IBM Transformation Extender map, the map run operation attaches the compiled map and input files to a request message and sends it to the Sterling B2B Integrator server. After processing has completed successfully, the Sterling B2B Integrator server returns a response message with the status of the map run, any error messages, and audit log file if available to the Design Studio.

To test running an IBM Transformation Extender map on a Sterling B2B Integrator server:

1. Configure the Design Studio connection properties for IBM Transformation Extender maps in the Sterling B2B Integrator preferences pane.
2. Complete any additional prerequisite steps depending on the map's functionality and settings.

Examples:

- You need to make sure that any maps that will be run by the **RUN** function are either checked into the Sterling B2B Integrator map repository, or located relative to the parent map location on the server.
 - If the Schema...Type...Metadata card setting specifies an XML schema, you need to check the schema into the Sterling B2B Integrator schema repository from the Sterling B2B Integrator Dashboard. For information about resolving XML entries, read the related topic in the IBM Transformation Extender for Sterling B2B Integrator documentation.
3. In the Outline view or Composition view, select one or more executable maps, and right-click the maps.
You can select multiple compiled IBM Transformation Extender maps to test run at the same time using the standard selection operations. Press Shift and click a range of maps, or press Ctrl and click individual maps.
 4. Click Run on Sterling B2B Integrator.
This starts the map run operation. For more details about what occurs during the map run operation, see [Run operation details](#).
 5. To run the maps in the background, from the Run on Sterling B2B Integrator: *host_URL* window, click Run in Background.
 6. You can monitor the operation's progress in the Progress view, and view the progress indicator. If you did not open the Progress view before you clicked Run on Sterling B2B Integrator, you can open it in either one of the following ways:
 - Click the progress icon.
 - Click Window...Show View...Other, and under General in the Show View window, click Progress.The Progress view opens.

You can also cancel the map run operation in this view.

The map run operation completes. For more details about the run results from testing an IBM Transformation Extender map on a Sterling B2B Integrator server, see [Result details from test running maps](#).

When you have completed testing your maps so that they run without errors and are ready for production, there are additional tasks to complete. You can deploy the maps to the production environment using the Deploy map to Sterling B2B Integrator option in the context menu of the selected maps. For information about running maps in a production environment, see the [IBM Sterling B2B Integrator documentation](#).

- **[Run operation details](#)**

When the operation to test running a IBM Transformation Extender map on a Sterling B2B Integrator server starts, several things occur.

- **[Result details from test running maps](#)**

These are the result details from test running IBM Transformation Extender maps on a Sterling B2B Integrator server.

Run operation details

When the operation to test running a IBM Transformation Extender map on a Sterling B2B Integrator server starts, several things occur.

The map run operation first verifies that the required connection properties in the Sterling B2B Integrator preferences pane contain values.

If you did not configure the required connection properties in the Sterling B2B Integrator preferences pane before doing this step and so the properties do not contain values, the verification process fails and the Error connecting window opens. The window has a link to the Sterling B2B Integrator preferences pane. You can click the Open preference page to update connection properties link, and in the Sterling B2B Integrator preferences pane, configure the connection properties. If you do not click the link, or you click the link but do not configure the connection properties in the Sterling B2B Integrator preferences pane, and then click OK, the map run operation ends and returns the presentation focus to the Design Studio.

You might be prompted to enter your Sterling B2B Integrator password depending on various scenarios. To learn more about the various scenarios, see [Setting password when test running or deploying map](#).

If the verification process passes, the Run on Sterling B2B Integrator: *host_URL* window opens and displays the progress of the map run operation. The window remains open and the progress of the map run operation continues to display until the operation completes. There is an option available for the maps to run in the background so that you can continue to access the Design Studio graphical user interface (GUI). You might want to select this option for long-running maps.

Result details from test running maps

These are the result details from test running IBM Transformation Extender maps on a Sterling B2B Integrator server.

If the operation ended successfully, the Request Status window opens and displays the `Processing completed successfully` message. The Sterling B2B Integrator server returns a response message with the status of the map or maps run.

If the Test Map service detects a problem, such as a problem connecting to the Sterling B2B Integrator server, or with the running of the map, the Problem Occurred window opens and displays the `Run on Sterling B2B Integrator: host_URL has encountered a problem` message, along with a summary of the problem. You can click Details to view the details about the problem.

If the Progress view was opened before you clicked Run on Sterling B2B Integrator, the status of the map run operation appears there. The Progress view displays the `Processing completed successfully` link. When you click the link, the Test Map Results window opens and lists the names of those map files that ran successfully. In the Test Map Results window, you can click the Show Sterling B2B Integrator Dashboard link to open the Sterling B2B Integrator Dashboard and verify the processes.

If the Progress view was not opened before you clicked Run on Sterling B2B Integrator, you can open the Progress view now. The status of the map run operation appears there. The Test Map Results window opens as if you clicked the `Processing completed successfully` link in the Progress view.

The status in the Progress view includes information such as the host and port name, and the time the process finished. The status indicates the progress of the map or maps run while the data is being mapped.

If you selected the Save service request and response messages option in the Sterling B2B Integrator preferences pane before you clicked Run on Sterling B2B Integrator, at run time, the map run operation saves the response files in the directory where the compiled map is located.

After running the map on Sterling B2B Integrator the output files are saved in the directories that the map's output card with File adapter specified.

The test runs under the MapTest business process. View additional details about the MapTest business process through the Sterling B2B Integrator Dashboard.

If you enabled the Map Audit function in the map's output card and specified `File` for the `MapAudit>.AuditLocation` map setting, the map run operation saves the audit log file in the file location that you specified. If you specified `Memory` for the `MapAudit>.AuditLocation` map setting, the map run operation saves the audit log file in the same working directory where the map is located.

Trace and backup files are not returned to the Design Studio. They are saved on the Sterling B2B Integrator server in the logs directory if map settings are set to map directory, or in custom directory, if custom directory is present.

The map run operation also creates a session log file as `map_name.sil`, which you can view through a text editor.

Deploying Transformation Extender maps

After you tested the running of IBM Transformation Extender maps in the Design Studio, you can deploy them from Design Studio to a Sterling B2B Integrator server map repository.

To deploy a IBM Transformation Extender map or maps to a Sterling B2B Integrator server map repository:

1. Configure the Design Studio connection properties for IBM Transformation Extender maps in the Sterling B2B Integrator preferences pane.
2. Configure the Design Studio Check In Map Service options for IBM Transformation Extender maps in the Sterling B2B Integrator preferences pane.

To make the version of the map that you deploy from the Design Studio the default map in the Sterling B2B Integrator map repository, see [Making the checked-in map the default](#).

3. Make the selections to deploy your map on a Sterling B2B Integrator server in either one of the two following ways, based on the view you use:
 - In the Navigator view or Extender Navigator view, select one or more compiled map files that you built manually for the Sterling B2B Integrator server platform, and right-click the map executable file.
 - In the Outline view or Composition view, select one or more maps, and right-click the maps.
You can select multiple compiled and executable IBM Transformation Extender maps to deploy at the same time using the standard selection operations. Press Shift and click a range of maps, or press Ctrl and click individual maps.
4. Click Deploy to Sterling B2B Integrator.
This starts the map deploy operation. For more details about what occurs during the map deploy operation, see [Deploy operation details](#).
5. To deploy the maps in the background, from the Deploy to Sterling B2B Integrator: *host_URL* window, click Run in Background.
6. You can monitor the operation's progress in the Progress view, and view the progress indicator. If you did not open the Progress view before you clicked Deploy to Sterling B2B Integrator, you can open it in either one of the following ways:
 - Click the green progress icon.
 - Click Window>Show View>Other, and under General in the Show View window, click Progress.The Progress view opens.

You can also cancel the map deploy operation in this view.

The map deploy operation completes. For more details about the run results from deploying IBM Transformation Extender maps from Design Studio to a Sterling B2B Integrator server map repository, see [Result details from deploying maps](#).

- **Making the checked-in map the default**

Select the Make checked in map default check box to make the version of the map that you deploy from the Design Studio the default map in the Sterling B2B Integrator map repository.

- **Deploy operation details**

This is what occurs when the operation to deploy a IBM Transformation Extender map to a Sterling B2B Integrator server map repository starts.

- **Result details from deploying maps**

These are the result details from deploying IBM Transformation Extender maps to a Sterling B2B Integrator server map repository.

Making the checked-in map the default

Select the Make checked in map default check box to make the version of the map that you deploy from the Design Studio the default map in the Sterling B2B Integrator map repository.

If there already is a version of the same map in the Sterling B2B Integrator map repository that is the default, this version of the map is made the default map instead.

When this version of the map is made the default map in the Sterling B2B Integrator map repository, this is the version that Sterling B2B Integrator uses for various processes such as validation, running as a step in a Business Process, translation or transformation, and for generating reports.

To keep the version of the same map that is already in the Sterling B2B Integrator map repository, as the default map, clear the Make checked in map the default check box.

Deploy operation details

This is what occurs when the operation to deploy a IBM Transformation Extender map to a Sterling B2B Integrator server map repository starts.

The map deploy operation first verifies that the required connection properties in the Sterling B2B Integrator preferences pane contain values.

If you did not configure the required connection properties in the Sterling B2B Integrator preferences pane before doing this step and so the properties do not contain values, the verification process fails and the Error connecting window opens. The window has a link to the Sterling B2B Integrator preferences pane. You can click the Open preference page to update connection properties link, and in the Sterling B2B Integrator preferences pane, configure the connection properties. If you do not click the link, or you click the link but do not configure the connection properties in the Sterling B2B Integrator preferences pane, and then click OK, the map deploy operation ends and returns the presentation focus to the Design Studio.

You might be prompted to enter your Sterling B2B Integrator password depending on various scenarios. To learn more about these scenarios, see the Setting password when test running or deploying map topic.

If the verification process passes, the Deploy to Sterling B2B Integrator: *host_URL* window opens and displays the progress of the map deploy operation. The window remains open and the progress of the map deploy operation continues to display until the operation completes. There is an option available for the maps to run in the background so that you can continue to access the Design Studio graphical user interface (GUI). You might want to select this option for long-running maps.

Result details from deploying maps

These are the result details from deploying IBM Transformation Extender maps to a Sterling B2B Integrator server map repository.

If the operation ended successfully, the Request Status window opens and displays the Processing completed successfully message. The IBM Transformation Extender map or maps are deployed to a Sterling B2B Integrator server map repository.

If the Checked In Map service detects a problem, such as a problem connecting to the Sterling B2B Integrator server, or with the deployment of the map, the Problem Occurred window opens and displays the Deploy to Sterling B2B Integrator: *host_URL* has encountered a problem message, along with a summary of the problem. You can click Details to view the details about the problem.

If the Progress view was opened before you clicked Deploy to Sterling B2B Integrator, the status of the map deploy operation appears there. The Progress view displays the Processing completed successfully link. When you click the link, the Check In Map Results window opens and lists the names of those map files that successfully finished being checked into a Sterling B2B Integrator server map repository. In the Check In Map Results window, you can click the Show Sterling B2B Integrator Dashboard link to open the Sterling B2B Integrator Dashboard and verify the checked-in files.

If the Progress view was not opened before you clicked Deploy to Sterling B2B Integrator, you can open the Progress view now. The status of the map deploy operation appears there. Alternatively, you can click the progress icon. The Check In Map Results window opens as if you clicked the Processing completed successfully link in the Progress view.

The status in the Progress view includes information such as the host and port name, and the time the process finished. The status indicates the progress of the map deploy operation while the map or maps are deployed to a Sterling B2B Integrator server map repository.

If you selected the Save service request and response messages option in the Sterling B2B Integrator preferences pane before you clicked Deploy to Sterling B2B Integrator, at run time, the map deploy operation saves the response files in the directory where the compiled map is located.

The map deploy operation also creates a session log file as *map_name.sil*, which you can view through a text editor.

Troubleshooting Transformation Extender maps

If you are testing running or deploying a IBM Transformation Extender map on a Sterling B2B Integrator server for the Test Map service or Check In Map service, and the map fails, follow these troubleshooting tips.

Verify that IBM Transformation Extender for Sterling B2B Integrator is installed correctly, and that Design Studio and Sterling B2B Integrator are configured and are running properly. To do this verification, run the Installation Verification Test (IVT) example that is under *install_dir\examples\integrations\SterlingB2B*, where *install_dir* refers to the path where IBM Transformation Extender is installed.

- [Troubleshooting for the Test Map service](#)
 - [Troubleshooting for the Check In Map service](#)
-

Troubleshooting for the Test Map service

To troubleshoot IBM Transformation Extender maps running on a Sterling B2B Integrator server using the Test Map service, you reference the error return codes, and the session and audit log files.

Error messages

When a map fails to run, it displays the error status in the Run window of the Design Studio. The following topics contain the error codes and messages that can be returned when your IBM Transformation Extender map fails.

- [Map execution error and warning messages](#)
- [Error messages in Design Studio for Sterling B2B Integrator services](#)

Log files for Transformation Extender maps

The following files are created automatically when running a IBM Transformation Extender map from the Design Studio:

- [Sterling B2B Integrator session log file](#)
- [Sterling B2B Integrator audit log file](#)

Each file is viewable with a text editor or from the Map Designer Organizer.

Trace and backup files are not returned to the Design Studio. They are saved on the Sterling B2B Integrator server in the logs directory if map settings are set to map directory, or in custom directory, if custom directory is present.

- [Sterling B2B Integrator session log file](#)
The Sterling B2B Integrator session log (.sil) file contains test session and execution status messages when running a IBM Transformation Extender map from the Design Studio to a Sterling B2B Integrator server.
 - [Sterling B2B Integrator audit log file](#)
The Sterling B2B Integrator audit log (.log) file is generated automatically at run time.
-

Sterling B2B Integrator session log file

The Sterling B2B Integrator session log (.sil) file contains test session and execution status messages when running a IBM Transformation Extender map from the Design Studio to a Sterling B2B Integrator server.

This log file also contains exceptions that occur during the session. The file is generated in the map directory, and is named after the executable map. The contents of this file can be viewed in a text editor.

Sterling B2B Integrator audit log file

The Sterling B2B Integrator audit log (.log) file is generated automatically at run time.

It is generated regardless of the Switch setting for MapAudit. If the map audit log is enabled on a Sterling B2B Integrator server, the map audit log that the server generates contains the same content as the audit log file that the map running locally generates.

The audit log file generally has content, regardless of a successful execution. It is named after the executable map (*map_name.log*) and is generated in the map directory. The contents of this file can also be viewed in the Audit Log view.

Troubleshooting for the Check In Map service

To troubleshoot IBM Transformation Extender maps deploying to a Sterling B2B Integrator server map repository using the Check In Map service, for specific Sterling B2B Integrator information, view the Sterling B2B Integrator log files. This is in addition to viewing the WebSphere Transformation Extender Design Studio log (.log) file located in the *your_workspace/.metadata* folder.

If the Check In Map service detects an error, the Problem Occurred window opens. Click OK to close the window, or Details to expand the window and view the error message.

For more information about the Sterling B2B Integrator log files and how to view them, see the [IBM Sterling B2B Integrator documentation](#).

WebSphere DataPower SOA Appliances

IBM DataPower SOA Appliances are network devices that can help secure and accelerate your XML and Web services deployments.

The Map Designer provides the features to build and test maps that you can run on, and deploy to IBM DataPower SOA Appliances.

See [DataPower appliance](#) to read about the benefits of using a DataPower appliance.

How the Map Designer works with DataPower appliances

Using the Map Designer, you can create maps that ultimately serve as executable programs designed to transform and route data. From the Map Designer, you test the maps on, and deploy them to the DataPower appliance, the appliance runs the executable maps, and then returns the results back to the Map Designer.

Getting started with DataPower appliances

To get started using the Map Designer with the WebSphere DataPower runtime option, use the following high-level task list in a preproduction phase.

- [Configure](#) your project to create DataPower maps by default, and enable the functions that are supported for IBM DataPower SOA Appliances.
- [Configure](#) the Map Designer for an WebSphere DataPower SOA Appliance.
- Define the structure of your data using the Type Designer and use the resulting schemas to create your DataPower maps.
- [Build](#) the DataPower maps with the map run time set to WebSphere DataPower.
- [Test](#) the DataPower maps in the Map Designer.
- [Deploy](#) the DataPower maps to be processed to an WebSphere DataPower SOA Appliance.

When your maps are ready for production, after you deployed the maps to an WebSphere DataPower SOA Appliance, there are additional tasks to complete using the DataPower appliance user interface. For information about running maps in a production environment, see the *IBM WebSphere DataPower SOA Appliances: Integrating with Transformation Extender* documentation that is published on the IBM DataPower Support website (<http://www.ibm.com/software/integration/datapower/support>).

- [Limitations of DataPower SOA appliances](#)
- [Project settings for DataPower maps](#)
- [Configuring for WebSphere DataPower SOA Appliance services](#)
- [Generating WebSphere DataPower maps](#)
- [Testing DataPower maps](#)
- [Deploying WebSphere DataPower maps](#)
- [Troubleshooting DataPower maps](#)

Limitations of DataPower SOA appliances

DataPower SOA appliances do not support all of the features of Map Designer. Some features are not available and the ability to use map and card settings is limited. The limitations on DataPower maps are listed here. For limitations on DataPower SOA appliances, see the [DataPower release notes](#).

Map settings

DataPower SOA appliances support all map settings except [MapTrace](#). For the [WorkSpace](#) setting, they support only the [Memory](#) value.

The mapping process sets the [Switch](#) setting for MapTrace to OFF, and the WorkSpace setting to Memory, and then runs the map by using those values. It also overrides the MapAudit and Warnings settings that are based on the settings on the WebSphere DataPower SOA Appliance.

If the ITX Audit Log map audit setting is enabled on the WebSphere DataPower SOA Appliance, that setting overrides the Switch setting for MapAudit to ON and the AuditLocation setting to Memory. If the map audit setting is not enabled on the Appliance, the mapping process sets the Switch setting for MapAudit to OFF. The mapping process generates the map audit log based on the map audit setting on the Appliance.

If the Support ITX Warnings map warning option is selected on the WebSphere DataPower SOA Appliance, the mapping process uses the values that you set for the Warnings property in the map settings. If the warning setting is not selected on the Appliance, the mapping process on the Appliance handles all map warnings as failures, unless the Warnings>Return setting is specified as Ignore. When you specify Warning>Custom, the mapping process on the Appliance handles those map warnings that are set as Warn or Fail as failures, while those map warnings set as Ignore, are ignored. To be able to use the Restart attribute and the REJECT function, the map warnings setting must be specified as Ignore. At run time, when the map detects invalid objects, it ignores them and continues processing.

To read about how to enable or disable map audit and warning settings on DataPower SOA appliances, see the DataPower documentation. For information about how to run a DataPower map locally, see [Running locally](#).

Card settings

DataPower SOA appliances support all card settings except the Backup setting. The mapping process sets the Switch setting for Backup to OFF for all of the input and output cards of a map. Then, the mapping process runs the map with that setting turned off. For information about how to run a DataPower map locally, see [Running locally](#).

The mapping process sets the DocumentVerification setting to Well Formed for all input cards of the map and then runs the map with that setting. It calls the external parser for document verification only for the well-formed document. So there is no external schema validation for DataPower maps.

Map Designer features

The following Map Designer features are not available for use with DataPower maps:

- [Debugger](#)
- [Profiler](#)

For information about how to run a DataPower map locally, see [Running locally](#).

Functions

Custom functions cannot be used in DataPower maps. In addition, DataPower SOA appliances do not support the following functions:

- **DBLOOKUP**
- **DBQUERY**
- **DDEQUERY**
- **EXIT**
- **GET**
- **GETANDSET**
- **GETDIRECTORY**
- **GETFILENAME**
- **GETRESOURCENAME**
- **JEXIT**
- **PUT**

The unsupported functions that are listed previously are applicable to both map and component rules.

XML functions

DataPower SOA appliances support the use of the following XML functions in map rules in DataPower maps:

- XPATH
- XSLT
- XVALIDATE

The above XML functions may need access to resources located either on the appliance or that are remotely accessible to the appliance. The resources referenced in the functions include XML, XSL, or XSD files. DataPower SOA appliances use their own methods to process these functions in the map rules.

Absolute resource paths can have only Uniform Resource Locators (URLs) that start with **local:///**, **store://**, **http://**, or **https://**. Resources that are on the DataPower SOA appliances can be accessed by the appliances through URL paths that start with **local:///** or **store://**. Resources that are located remotely can be accessed through URL paths that start with **http://** or **https://**.

Relative resource paths are resolved by the mapping process, which appends the relative path to the absolute path of its parent map. Relative resource paths of the top-level map are resolved based on the working directory that is set by the DataPower SOA appliance for the top-level map. If you specify relative paths that contain the parent directory (..) notation, which is not valid for DataPower SOA appliances, the mapping process returns an error.

DataPower maps also support paths that start with **current://**. Resource paths starting with **current://** are run like the relative resource paths defined above. Therefore, **current://** equates to the . (current directory) notation in relative paths and is substituted with the parent map path during transformation. DataPower maps do not support **file://** URL paths in the XML functions.

Since DataPower SOA Appliances use their own methods to process these XML functions, map runtime behavior differences may occur when running DataPower maps on an appliance against running them locally on the system. Errors reported with the **LASTERRORMSG** function may differ for appliance and local runs of DataPower maps.

For information about how to run a DataPower map locally, see [Running locally](#). To read about the functions, see the related topics in the *Functions and Expressions* documentation.

Resource adapters

DataPower SOA appliances support the following IBM Transformation Extender resource adapters:

- ECHO
- FILE
- SINK
- STATIC FILE

The FILE and SINK adapters are the only target options available for output. The ECHO, FILE, and STATIC FILE adapters are the only source options available for input. IBM DataPower SOA Appliances do not support the other IBM Transformation Extender resource adapters.

RUN maps

DataPower SOA appliances support the use of the **RUN** function in map and component rules in DataPower maps, but with specific limitations.

All of the limitations that are described in this *Limitations* topic apply as well to **RUN** maps. Here are more limitations that apply only to **RUN** maps.

- If the map input source and output target cards specify adapters other than ECHO, SINK, or STATIC FILE, you must use the **-IE**, **-OE**, or **-OASINK** command-line options to override the adapters. You can use the **ECHOIN** or **HANDLEIN** functions to override the adapters in the map input source cards. If you do not override the input source or output target adapters that are not supported for **RUN** maps, for example, the FILE adapter, the mapping process returns **Source not available (12)** or **Target not available (9)** errors.
- The following commands and command options are ignored:
 - **-T (MapTrace)** command
 - **-W (WorkSpace)** command
 - **K (Backup)** option with **-OASINK** command
 - **EN, EA, EF, ES** (DocumentVerification) options with **-IE**, **-OE**, or **-OASINK** commands
 - **U, +, !+, ={file|dir}** options with **-A (MapAudit)** command
- The map audit log is generated only when both of the following settings are specified:
 - The MapAudit setting is set to ON, either with the **MapAudit** switch map setting that is set to ON or with the **-A (MapAudit)** command line that is specified along with options.
 - The ITX Audit Log map audit setting is set on the DataPower SOA appliances.
- Absolute map paths can have only Uniform Resource Locators (URLs) that start with **local:///**, **store://**, **http://**, or **https://**. Maps that are on the DataPower SOA appliances can be accessed by the appliances through URL paths that start with **local:///** or **store://**. Maps that are located remotely can be accessed through URL paths that start with **http://** or **https://**.
- Relative map paths are resolved by the mapping process, which appends the relative path to the absolute path of its parent map. Relative **RUN** map paths of the top-level map are resolved based on the working directory that is set by the DataPower SOA appliances for the top-level map. If you specify relative paths that contain the parent directory (..) notation, which is not valid on DataPower SOA appliances, the mapping process returns **Could not open map(3)** error.

For information about how to run a DataPower map locally, see [Running locally](#). To read about the execution commands, and the command availability, which lists the commands that are valid with the **RUN** function, see the related topics in the *Execution Commands* documentation.

The Database Interface Designer is only available for use with IBM Transformation Extender maps. DataPower SOA appliances do not support IBM Transformation Extender database adapters.

Integration Flow Designer

The Integration Flow Designer is only available for use with IBM Transformation Extender maps. DataPower maps within a map source file are not visible in the Integration Flow Designer.

Resource Registry

The Resource Registry is only available for use with IBM Transformation Extender maps. DataPower maps do not support resource aliases in map settings, card settings, and map rules. Resource configuration (.mrc) files and Resource name (.mrn) files created by using the Resource Registry are not supported on DataPower SOA appliances.

Enterprise Application Packs and Industry Packs

DataPower SOA appliances do not support the IBM Transformation Extender Enterprise Application Packs.

For the latest information, see [DataPower SOA appliances support of IBM Transformation Extender Industry Packs](#).

For information about a specific release of the Industry Packs, see the IBM Transformation Extender [release notes](#).

WSDL Importer

DataPower SOA appliances do not support the Transformation Extender Web Services Description Language (WSDL) Importer.

Map runtime behavior differences

Maps that are run on a WebSphere DataPower SOA Appliance behave differently from maps that are run on other platforms on which IBM Transformation Extender maps run. The mapping process behaves differently in the following ways:

- Processes the NULL value in a character text item for XML types, which are specified in the schema with the Intent property set to XML, as an invalid character.
- Processes the year 0000 value in the Date & Time item as an invalid date.
- Overrides map and card settings before it runs the WebSphere DataPower maps. See [Map settings](#) and [Card settings](#).
- Processes some of the XML functions with its own methods. For more information, see [XML functions](#).

Tip: If you abide by these limitations, you can initially create your maps by using the standard IBM Transformation Extender run time to take advantage of features such as the Map Debugger and Profiler. When your map unit testing is successful, you can then change the runtime setting to WebSphere DataPower and proceed to build and run the maps. For information about how to run a DataPower map locally, see [Running locally](#).

Project settings for DataPower maps

You can configure settings for your project for DataPower maps.

There are two project settings that you can configure:

- Default Map Runtime
- Update Map Runtime

You configure the project settings in the Map Designer under Project > Properties > Transformation Extender. These project settings are used to set and update the MapRuntime setting to indicate the map runtime environment in source map (.mms) files.

Use the Default Map Runtime setting with the DataPower check box selected to create DataPower maps by default. This selection sets the MapRuntime setting to DataPower in source map (.mms) files that you create. If you clear the DataPower check box, it sets the MapRuntime setting to Integration Platform in source map (.mms) files that you create. You might want to use this setting when most of the maps you create are to be deployed to an WebSphere DataPower SOA Appliance.

Use the Update Map Runtime setting to update the MapRuntime setting with a value that you select in the MapRuntime list, in all of the executable maps in all of the map source (.mms) files in the project. Select one of the values in the MapRuntime list: DataPower or Integration Platform. Click Update and toggle between updating the MapRuntime setting to the selected value in all of the executable maps in all of the map source (.mms) files in the project. You might want to use this project setting when you need to compile all executable maps for IBM DataPower SOA Appliances, or environments other than IBM DataPower SOA Appliances, with one set of source maps.

- [Configuring your project to create DataPower maps by default](#)

You can configure your project to create DataPower maps by default.

- [Configuring your project to update the MapRuntime setting in all of the executable maps](#)

You can configure your project to update the MapRuntime setting in all of the executable maps in all of the map source (.mms) files in the project.

Configuring your project to create DataPower maps by default

You can configure your project to create DataPower maps by default.

If most of the maps you create are to be deployed to an WebSphere DataPower SOA Appliance, before you begin creating your DataPower maps, you can change the Default Map Runtime setting on your project to DataPower. All new maps that you create are DataPower maps by default. This setting makes all Map Designer functions that IBM DataPower SOA Appliances support available to you as you design your DataPower maps.

To configure the Default Map Runtime setting:

1. In the Map Designer, select the project in the Extender Navigator view, and click Project > Properties > Transformation Extender.
2. In the Default Map Runtime pane, select the DataPower check box.
3. Click OK to save the change.

When you create a map, DataPower is the value set in the MapRuntime setting by default. You can view this value in the MapRuntime setting in the Map Settings window.

Configuring your project to update the MapRuntime setting in all of the executable maps

You can configure your project to update the MapRuntime setting in all of the executable maps in all of the map source (.mms) files in the project.

When you need all of the executable maps in all of the map source (.mms) files in your project to have the same MapRuntime setting, you can use the MapRuntime setting that is in the Update Map Runtime pane. It updates the MapRuntime setting to indicate the map runtime environment in all of the source maps.

To configure your project to update the MapRuntime setting in all of the executable maps in all of the map source (.mms) files in your project:

1. In the Map Designer, select the project in the Extender Navigator view, and click Project > Properties > Transformation Extender.
2. In the Update Map Runtime pane, select from the MapRuntime list, either Integration Platform or DataPower.
3. Click Update.

All of the executable maps in all of the map source (.mms) files in your project are updated with the value that you selected from the MapRuntime list for the MapRuntime setting. If you selected DataPower, only those maps that use the functions and adapters that DataPower supports, are updated. If the source map is open, maps that were updated are listed in the Outline view or Composition view with their respective map icon.

You can click OK or Cancel to close the Properties window.

Configuring for WebSphere DataPower SOA Appliance services

You have to configure the Map Designer settings for the Interoperability Test Service and the XML Management Interface service to test and deploy DataPower maps on an WebSphere DataPower SOA Appliance.

- [Configuring for the Interoperability Test Service](#)
- [Configuring for the XML Management Interface service](#)

Configuring for the Interoperability Test Service

To test a DataPower map on an WebSphere DataPower SOA Appliance from the Map Designer, you must configure the connectivity settings for the Interoperability Test Service in the Map Designer. The host name or IP address, and a port number for the DataPower appliance are required. The connection is established using HyperText Transport Protocol (HTTP), which is the default connection type, or HyperText Transport Protocol Secure (HTTPS).

You can change default Map Designer settings pertaining to connectivity for the Interoperability Test Service in the DataPower preferences.

To configure a connection to the appliance for the Interoperability Test Service using HTTP in the Map Designer:

1. In the Map Designer, click Window > Preferences > Transformation Extender > Map > DataPower.
The DataPower preferences pane opens.
 2. Under Connectivity, enter the IP address or the host name for the DataPower appliance in the Host field. (For IPv6, enclose the IP address in brackets.)
The default value is `localhost`.
 3. Click the Interoperability Test Service tab.
 4. Enter a value for the Port field.
The default port for the HTTP Service is 9990. If you are using HTTPS, you must enter a different value for the Port field. See the topic about [configuring an HTTPS connection](#).
 5. Click OK to save changes.
- [Configuring an HTTPS connection](#)
 - [Save Messages options](#)
 - [Appliance Configuration options](#)
 - [Working Directory option](#)
 - [Retry map setting](#)

Configuring an HTTPS connection

The Map Designer establishes an HTTP connection with an WebSphere DataPower SOA Appliance, which is the default connection type, using the host and port parameters specified in the options for the Interoperability Test Service. To have a secure connection with the DataPower appliance, you can use Hypertext Transfer Protocol Secure. (HTTPS).

To establish a secure connection using HTTPS, you must first enable the security option in the Map Designer. When you run a DataPower map from Map Designer on the IBM DataPower SOA Appliances, you are prompted through a standard Certificate window to dynamically select the option to trust the certificate from the appliance, and add it to the certificate store (`install_dir\java\lib\security\cacerts`, where `install_dir` refers to the path where IBM Transformation Extender is installed). The server certificate and private key must be set up on the DataPower appliance. For information about how to set up the server certificate and private key on the DataPower appliance, see the related topics in the *IBM WebSphere DataPower SOA Appliances: Integrating with Transformation Extender* documentation that is published on the IBM WebSphere DataPower Support website (<http://www.ibm.com/software/integration/datapower/support>).

To configure a secure connection to the appliance for the Interoperability Test Service using HTTPS in the Map Designer:

1. In the Map Designer, click Window > Preferences > Transformation Extender > Map > DataPower.
The DataPower preferences pane opens.
2. Under Connectivity, enter the IP address or the host name for the DataPower appliance in the Host field. (For IPv6, enclose the IP address in brackets.)

The default value is localhost.

3. Click the Interoperability Test Service tab.
4. Enter a value for the Port field.
The default port for the HTTPS Service is 9991.
5. Select the Secure Transport check box.
6. Click OK to save changes.

Save Messages options

The Save Messages options for the Interoperability Test Service pertain to the request and response messages, and the session log file.

Use the files that are generated when you select the Save Messages options to debug the problems that your map might encounter while running on a DataPower appliance. You can view these files using a text editor. By default, the options are not selected.

To access these settings, click Window > Preferences > Transformation Extender > Map > DataPower, and under Connectivity, click the Interoperability Test Service tab.

Request option

Select the Request option to save a copy of the request message (*map_name_Request.msg*) in the map directory.

The request message is the initial communication passed from the Map Designer to the DataPower appliance. It contains information about the compiled map, input files, as well as information about the Appliance configuration options, and mapping information pertaining to the input and output cards.

Response option

Select the Response option to save a copy of the response message (*map_name_Response.msg*) in the map directory.

The response message is a message generated by the DataPower appliance that passes the output and audit log files back to the Map Designer.

Session option

Select the Session option to generate the DataPower session log (*map_name.dpl*) file in the map directory.

The DataPower session log file contains test session information about a DataPower map running on a DataPower appliance from the Map Designer. It logs exception, informational, and execution messages.

Appliance Configuration options

The Appliance Configuration options pertain to the ITX Audit Log map audit setting, and the Support ITX Warnings setting for the Interoperability Test Service, and for the DataPower maps running locally.

To access the Appliance Configuration options, click Window > Preferences > Transformation Extender > Map > DataPower, and under Connectivity, click the Interoperability Test Service tab.

ITX Audit Log

Specify the audit context name for the Interoperability Test Service that the WebSphere DataPower SOA Appliance references to generate and save the map audit log. If you do not specify the audit context name, the audit log is not generated, and therefore it is not saved. By default, the audit context name is not specified; the field is blank.

If the ITX Audit Log map audit setting is set on the WebSphere DataPower SOA Appliance for the Interoperability Test Service from the Map Designer, the audit log is generated regardless of the MapAudit->Switch map setting.

You must specify the ITX Audit Log map audit setting for DataPower maps running locally to generate the map audit log regardless of the MapAudit->Switch map setting. This specification is required to generate the map audit log even if you run the compiled DataPower map from the Extender Navigator view using the Map Settings option.

Support ITX Warnings

Select the Support ITX Warnings check box to support map warnings on the WebSphere DataPower SOA Appliance from the Map Designer for the Interoperability Test Service, when running DataPower maps locally. The Support ITX Warnings check box option is not selected by default.

If the Support ITX Warnings map warning option is selected on the WebSphere DataPower SOA Appliance for the Interoperability Test Service from the Map Designer, the mapping process uses the values that you set for the Warnings property in the map settings. If the warning setting is not selected on the Appliance, the mapping process on the Appliance handles all map warnings as failures, unless the Warnings->Return setting is specified as Ignore. When you specify Warning->Custom, the mapping process on the Appliance handles those map warnings set as Warn or Fail as failures, while those map warnings set as Ignore, are ignored.

You must specify the Support ITX Warnings map warning option for DataPower maps running locally to support map warnings. This option is required to support map warnings even if you run the compiled DataPower map from the Extender Navigator view using the Map Settings option.

Working Directory option

The Working Directory option for the Interoperability Test Service pertains to specifying the Uniform Resource Locator (URL). The Interoperability Test Service uses the working directory URL to resolve the relative **RUN** map paths of the DataPower map being tested on the WebSphere DataPower SOA Appliance from the Map Designer. This option is not used by the DataPower maps running locally.

To specify the working directory URL, click Window > Preferences > Transformation Extender > Map > DataPower, and under Connectivity, click the Interoperability Test Service tab.

URL

Select the Uniform Resource Locator (URL) from the URL list. The default URL is local:///{{project_name}}/.

Custom

If the URL you need is not in the URL list, you can create a custom URL by clicking Custom. After you click Custom, the Custom URL window is displayed.

In the Custom URL window, you can specify the URL to resolve the relative **RUN** map paths. The default is local:/// . You can define custom URL paths that start with local:///, store:///, http:// and https://.

Retry map setting

The Retry map setting can be used to request additional connection attempts to the DataPower appliance when a failure occurs. After you set the `RetryAttempts` setting to ON, you can enter maximum number of attempts and interval values. The `MaxAttempts` value is the maximum number of connection attempts to make. The default value for this setting is 10.

The Interval value determines the amount of time (in seconds) to allow between connection attempts. The default value for this setting is 5.

Configuring for the XML Management Interface service

To deploy a map on a WebSphere DataPower SOA Appliance, you must configure the connectivity settings for the XML Management Interface service in the Map Designer.

The host name or IP address, password, port number, domain, and file options for the DataPower appliance are required. The connection is established using HTTPS.

You can change default Map Designer settings pertaining to connectivity for DataPower maps in the DataPower preferences.

When you deploy a DataPower map from Map Designer on the IBM DataPower SOA Appliances, you are prompted to establish a secure connection using HTTPS, through a standard Certificate window, to dynamically select the option to trust the certificate from the appliance, and add it to the certificate store (`install_dir\java\lib\security\cacerts`, where `install_dir` refers to the path where IBM Transformation Extender is installed). The server certificate and private key must be set up on the DataPower appliance. For information about how to set up the server certificate and private key on the DataPower appliance, see the related topics in the *IBM WebSphere DataPower SOA Appliances: Integrating with Transformation Extender* documentation that is published on the [IBM DataPower Support website](#).

To configure a secure connection to the appliance using HTTPS in the Map Designer:

1. In the Map Designer, click Window > Preferences > Transformation Extender > Map > DataPower.
The DataPower preferences pane opens.
2. Under Connectivity, enter the IP address or the host name for the DataPower appliance in the Host field. (For IPv6, enclose the IP address in brackets.)
The default value is `localhost`.
3. Click the XML Management Interface Service tab.
4. Enter your DataPower user name and password for the User and Password fields.
These fields are the credentials to log on to the DataPower File Manager.
5. Enter a value for the Port field.
The default port for the XML Management Interface service is 5550.
6. Enter the DataPower deploy domain for the Domain field.
7. Under File Options, select the target location of the file being deployed by using one of the following file options:
 - Default file - Select one of the predefined location variables from the Default file list. The default predefined location value is `local:///{{project_name}}/{{map_name}}`.
 - Custom - Define your own custom target location other than the predefined locations in the Default file list.
 - Overwrite file without asking - Select to overwrite a file that was previously deployed to an WebSphere DataPower SOA Appliance without asking for permission.
8. Click OK to save changes.

- [Setting Custom File name](#)

Setting Custom File name

Use the Custom File window to define where the file is to be deployed if it is to a location other than the predefined locations in the Default file list.

To define the custom target location of the file being deployed:

1. In the Map Designer, click Window > Preferences > Transformation Extender > Map > DataPower.
The DataPower preferences pane opens.
2. Under Connectivity, click the XML Management Interface Service tab.
3. Under File Options, click Custom.
The Custom File window opens.
4. Select the custom file name from the list under Enter the custom file name.
The default value is `local://`.
5. Define your own location by specifying either the actual path or variables, or a combination of both path and variables.
6. Click OK.
This returns you to the Connectivity window with your location in the Default file field.

Generating WebSphere DataPower maps

Before building a DataPower map, the `MapRuntime` setting must be set to DataPower.

To set the map runtime setting:

1. In the Outline view or Composition view, select the map and do one of the following choices:
 - On the Map menu, click Map Settings.
 - On the toolbar, click the Map Settings icon.
 - Right-click the map and on the menu, click Map Settings.
2. In the Map Settings window, locate the `MapRuntime` setting.

3. Display the list, and select DataPower.

4. Click OK.

In the Outline view or Composition view, a DataPower map is represented by a distinct icon.

The next step is to build the map. After a successful build, you can run the map. When you run a DataPower map, the compiled map, and input files are sent to the WebSphere DataPower SOA Appliance for processing. The results are returned to the Map Designer.

This is a preproduction phase. When your maps are ready for production, there are additional tasks to complete using the WebSphere DataPower SOA Appliance user interface. For information about running maps in a production environment, see the DataPower documentation.

- [Building a WebSphere DataPower map](#)

Building a WebSphere DataPower map

Use the build process in the Map Designer to generate a compiled map with the DataPower appliance (.dpa) extension.

To build a DataPower map:

1. In the Outline view or Composition view, select the map and do one of the following choices:

- On the Map menu, click Build.
- On the toolbar, click the Build icon.
- Right-click the map and on the menu, click Build.

2. From the menu bar, click Map > Build.

A window opens that displays the build status and then closes.

To verify that the map was built without errors, open the Organizer folder and double-click the Build Results element. The Build Results view opens. A blank Build Results view indicates that there were no build errors.

The build results can also be viewed using a text editor. The file (*map_name.mme*) is located in the map directory.

If you are building multiple DataPower maps, select the map in the Outline view or Composition view, and from the menu bar, click Map > Build All.

During the build process, the Map Designer generates a compiled map with the DataPower appliance (.dpa) extension.

Testing DataPower maps

You can test a DataPower map by running it on an WebSphere DataPower SOA Appliance from the Map Designer, or by running it locally.

- [Running on a WebSphere DataPower SOA Appliance](#)
- [Running locally](#)

Running on a WebSphere DataPower SOA Appliance

When you run a DataPower map, the compiled map, and input files are attached to a request message that is sent to the WebSphere DataPower SOA Appliance. All RUN maps referenced by the DataPower map that is run on the appliance, must be in the specified RUN map URL paths. RUN maps are not packaged and sent to the appliance in the request message for the Interoperability Test Service. After processing completes, a response message is returned with the output and audit log files.

If you built a map to be run on an WebSphere DataPower SOA Appliance, to test it from the Map Designer, the Interoperability Test Service must be running on the appliance.

To run a DataPower map on an WebSphere DataPower SOA Appliance:

1. In the Outline view or Composition view, select the map and do one of the following choices:

- On the Map menu, click Run.
- On the toolbar, click the Run icon.
- Right-click the map and on the menu, click Run.

If you configured the Interoperability Test Service using HTTPS in the Map Designer, after you click Run to test the DataPower map, you are prompted through a standard Certificate window to dynamically select the option to trust the certificate from the appliance, and add it to the certificate store.

2. Click Cancel to close the IBM DataPower SOA Appliances window.

3. To view results, right-click the map and click Run Results on the menu.

The Run Results window opens.

4. Select the items you want to view and click OK.

The results are displayed.

The IBM DataPower SOA Appliances window opens and displays the executable map name, host and port information, status, and elapsed time. These statistics indicate the progress of the map run while the data is being mapped. After the mapping process completes, the outputs are saved as specified in the settings of the output cards, and the map audit log file is saved as specified in the map settings.

When you completed testing your maps so that they run without errors and are ready for production, there are additional tasks to complete. You can deploy the maps to the production environment using the Deploy to DataPower option in the menu of the selected maps. For information about running maps in a production environment, see the [IBM DataPower SOA Appliances: Integrating with Transformation Extender](#) documentation that is published on the [IBM DataPower Support website](#).

Running locally

The Run Locally command that is on the menu for the map, is enabled for DataPower maps.

- The command is used to run DataPower maps locally on the system, instead of on IBM DataPower SOA Appliances.
- All of the map settings, MapAudit in File, MapTrace, and WorkSpace in File, in addition to the Backup card setting that IBM DataPower SOA Appliances do not support, are supported when you run DataPower maps locally.
- To troubleshoot and debug your map, you can specify the following options to generate the map audit and trace log files:
 - Set the Switch setting for the MapTrace map setting to ON in the Map Settings window.
 - Specify the audit context name in the ITX Audit Log map audit setting in the Appliance Configuration options in the DataPower preferences window. When you run maps locally, you must specify the HIP Audit Log map audit setting to generate the map audit log. You specify this setting in this same way whether you are testing DataPower maps by running them on a WebSphere DataPower SOA Appliance, or locally, or whether you are running them on an WebSphere DataPower SOA Appliance from the Map Designer.

If the MapAudit.`_AuditLocation` setting on the DataPower map is set to Memory, the map audit log is saved in the map directory and named after the executable map (`map_name.log`).

- To support map warnings, select Support ITX Warnings option under Appliance Configuration options for DataPower. The mapping process uses the values that you set for the Warnings property in the map settings. If the warning setting is not selected, the mapping process handles all map warnings as failures, unless the Warnings.`_Return` setting is specified as Ignore. When you specify Warning.`_Custom`, the mapping process handles those map warnings that are set as Warn or Fail as failures, while those map warnings set as Ignore, are ignored. To be able to use the **Restart** attribute and the **REJECT** function, the map warnings setting must be specified as Ignore. At run time, when the map detects invalid objects, it ignores them and continues processing.
- The mapping process sets the DocumentVerification setting to Well Formed for all input cards of the map and then runs the map with that setting. It calls the external parser for document verification only for the well-formed document. So there is no external schema validation for DataPower maps.
- When you locally run a DataPower map that is compiled with a static input file, the mapping process validates the static input file in the compiled map only once, when the map loads, and creates a trace file in the map directory only if the static file input fails validation.
- If the map input source and output target cards specify adapters other than ECHO, SINK, or STATIC FILE, you must use the **-IE**, **-OE**, or **-OASINK** command-line options to override the adapters. You can use the **ECHOIN** or **HANDLEIN** functions to override the adapters in the map input source cards. If you do not override the input source or output target adapters that are not supported for **RUN** maps, for example, the FILE adapter, the mapping process returns `Source not available (12) or Target not available (9)` errors.
- The following commands and command options are supported when you run DataPower maps locally even though they are not supported on IBM DataPower SOA Appliances:
 - **-T (MapTrace)** command
 - **-W (WorkSpace)** command
 - **K (Backup)** option with **-OASINK** command
 - **U, +, !+, ={file|dir}** options with **-A (MapAudit)** command
- The **EN**, **EA**, **EF**, **ES** (DocumentVerification) options with **-IE**, **-OE**, or **-OASINK** commands are ignored.
- When you run DataPower maps locally, the mapping process ignores the absolute map URL path that references a **RUN** map. Instead, the mapping process searches for the **RUN** map in the map directory. Absolute map URL paths start with `local:///`, `store://`, `http://`, or `https://`.
- Relative paths that contain the parent directory (..) notation, are not valid on IBM DataPower SOA Appliances and when you run the DataPower map locally. If you specify relative paths in this way, the mapping process returns `Could not open map(3)` error. The mapping process resolves the relative **RUN** map paths that are based on the map directory of the parent map.
- When you run DataPower maps locally, the mapping process ignores the absolute resource URL path that references a resource located either on the appliance or that is remotely accessible to the appliance when specified for the XML functions (such as XPATH, XSLT, or XVALIDATE). Instead, the mapping process searches for the resource in the map directory. Absolute map URL paths start with `local:///`, `store://`, `http://`, or `https://`.
- Relative resource paths that contain the parent directory (..) notation are not valid on DataPower appliances when you run the DataPower map locally. If you specify relative resource paths in this manner for the XML functions, the mapping process returns an error. The mapping process resolves the relative resource paths that are based on the map directory of the parent map.
- DataPower maps that are run locally also support paths that start with `current://`. Resource paths starting with `current://` are run like the relative resource paths defined above. In this scenario, `current://` equates to the .. (current directory) notation in relative paths and is substituted for the parent map path during transformation. DataPower maps do not support `file://` URL paths in the XML functions.
- DataPower appliances use their own methods to process some XML functions (such as XPATH, XSLT, or XVALIDATE), and map runtime behavior differences may occur when running DataPower maps on an appliance against running them locally on the system. Errors reported with the LASTERRORMSG function might differ between appliance and local runs of DataPower maps.

Default map settings are set in the Run Options window that you open from Window > Preferences > Transformation Extender > Map > Run Options. After a map is created, select Map Settings from the Map menu to change the settings for the selected map.

You can override the original map and card settings of the DataPower map in the Map Designer. In the Extender Navigator view, right-click the DataPower (.dpa) map in the Map Executables folder, and on the menu, click Map Settings. Use this feature to test different combinations of map and card settings dynamically with the same DataPower (.dpa) map without recompiling the source map.

To run a DataPower map locally:

1. Run the DataPower map locally by using one of the following menu options:
 - In the Extender Navigator view, right-click the compiled DataPower (.dpa) map in the Map Executables folder, and click Run Locally.
 - In the Outline view or Composition view, right-click the executable DataPower, and on the menu, click Run Locally.
2. Click Cancel to close the window.
3. To view results, right-click the map and click Run Results on the menu.
The Run Results window opens.
4. Select the items that you want to view and click OK.
The results are displayed.

The Command Server window opens. It displays the executable map name, status, total number of input objects that are found, total number of output objects that are built, and elapsed time. These statistics indicate the progress of the map that is run, while the data is being mapped. After the mapping process completes, the outputs are saved as specified in the settings of the output cards. The map audit log and trace files are saved as specified in the map settings.

Deploying WebSphere DataPower maps

After you tested the DataPower maps in the Map Designer, you can deploy them from Map Designer to an WebSphere DataPower SOA Appliance.

To deploy a DataPower map or maps:

1. Configure the Map Designer deployment settings for DataPower maps using the DataPower preferences.
2. From the Map Designer, deploy the DataPower map using one of the following menu options:
 - In the Extender Navigator view, right-click the compiled DataPower (.dpa) map in the Map Executables folder, and select Deploy to DataPower.
 - In the Outline view or Composition view, right-click the executable DataPower map, and on the menu, click Deploy to DataPower.

You can select multiple compiled and executable DataPower maps to deploy at the same time using the standard selection operations. Press **Shift** and click a range of maps, or press **Ctrl** and click individual maps; then right-click and select Deploy to DataPower.

If you configured the XML Management Interface service in the Map Designer, after you click Deploy to DataPower to deploy the DataPower map from Map Designer to an WebSphere DataPower SOA Appliance, you are prompted through a standard Certificate window to dynamically select the option to trust the certificate from the appliance, and add it to the certificate store.

3. If the Overwrite file without asking checkbox in the XML Management Interface tab under Connectivity in the DataPower preferences pane is clear, that is, not selected, the Overwrite File window opens, and you can select one of several overwrite options for the deployment process.

The DataPower map or maps are deployed to an WebSphere DataPower SOA Appliance.

If the XML Management Interface service detects an error, the Problem Occurred window opens. Click OK to close the window, or Details to expand the window and view the error message.

- [Setting password for XML Management Interface service](#)
- [Overwriting a deployed file](#)

Setting password for XML Management Interface service

When you select Deploy to DataPower to deploy a DataPower map from the Map Designer to an WebSphere DataPower SOA Appliance, you might or might not be prompted to enter your DataPower password in the Enter Password window.

If the Enter Password window does open, it is because one of the following scenarios occurred:

- You did not enter a password under Connectivity, in the XML Management Interface Service tab of the DataPower preferences pane that opens from Window > Preferences > Transformation Extender > Map > DataPower.
- You entered a password in the DataPower preferences pane, but you subsequently closed the Map Designer application before deploying the DataPower map. When you enter your password, it is saved only in memory, and therefore, available to the deploy process until you exit the Map Designer. After you exit the Map Designer application, your password is no longer available when you restart Map Designer and attempt to deploy a DataPower map.

To set password:

1. Enter a password in the Enter Password window.
2. Click OK.
The Enter Password window closes.

The XML Management Interface service validates your DataPower user name, password and domain.

If the authentication is successful, the deployment process continues.

If the authentication fails, the Problem Occurred window opens. Click OK to close the window, or Details to expand the window and view the error message.

Overwriting a deployed file

Use the Overwrite File window to overwrite a file that was previously deployed to an WebSphere DataPower SOA Appliance without asking for permission. This window opens if the Overwrite file without asking check box in the XML Management Interface Service tab under Connectivity in the DataPower preferences pane is clear, that is, not selected. There are three file overwrite options for the deployment process.

To overwrite a file:

1. Click one of the following file overwrite options for the deployment process:
 - Overwrite the current file, which overwrites the same file that was previously deployed.
 - Deploy the file with a different name, which overwrites the file with the absolute path location where the file is to be deployed that you enter in the File field.
 - Cancel the deployment, which cancels the deployment process.
2. Select Remember my decision if you want subsequent deployments to use the settings you selected in this Overwrite File window.
When you select Remember my decision, Map Designer selects the Overwrite file without asking check box in the DataPower preferences pane. The next time you deploy a DataPower map to an WebSphere DataPower SOA Appliance, the deployment process does not open the Overwrite File window.
3. Click Proceed.

If you specified a new file name, the DataPower map is deployed to an WebSphere DataPower SOA Appliance with the new file name.

If the XML Management Interface service detects an error, the Problem Occurred window opens. Click OK to close the window, or Details to expand the window and view the error message.

Troubleshooting DataPower maps

If you are testing or deploying a DataPower map on a DataPower appliance for the Interoperability Test Service or XML Management Interface service, and the map fails to run on or deploy to the DataPower appliance, follow the troubleshooting tips listed here.

To troubleshoot DataPower environment-specific issues while you are testing or deploying a DataPower map on the appliance, see the troubleshooting tips in the DataPower documentation that is published on the [IBM DataPower Support website](#). Read about the DataPower logs on the appliance, to help you troubleshoot these issues, in the *IBM DataPower SOA Appliances: Problem Determination Guide* documentation that is published on the [IBM DataPower Support website](#).

- [Troubleshooting for the Interoperability Test Service](#)
- [Troubleshooting for the XML Management Interface service](#)

Troubleshooting for the Interoperability Test Service

To troubleshoot DataPower maps running on a WebSphere DataPower appliance using the Interoperability Test Service, you reference the error return codes, and the session and audit log files.

Error messages

When a map fails to run, it displays the error status in the Run dialog box of Map Designer. The following topics contain the error codes and messages that can be returned when your DataPower map fails.

- [Map execution error and warning messages](#)
- [Error messages for maps](#)

Log files for DataPower maps

The following files can be used for troubleshooting the errors when running a DataPower map from the Map Designer:

- DataPower session log file
- DataPower audit log file

Each file is viewable with a text editor or from the Organizer in the Map Designer.

- [DataPower session log file](#)
- [DataPower audit log file](#)

DataPower session log file

The DataPower session log (.dpl) file contains test session and execution status messages when running a DataPower map from the Map Designer. This log file also contains exceptions that occur during the session. The file is generated in the map directory, and is named after the executable map. The contents of this file can be viewed in the Session view as well as in a text editor.

- [Viewing the DataPower session log file](#)

Viewing the DataPower session log file

Use the Session view to examine the contents of the DataPower session log.

To view the DataPower session log:

1. Expand the map in the Outline view, and expand the Organizer folder.
2. Double-click the Session element.

The Session view opens, and you can view the contents of the DataPower session log file. The *map_name.dpl* is generated in the map directory and can also be viewed using a text editor.

DataPower audit log file

The DataPower audit log is generated at run time, regardless of the Switch setting for MapAudit. If the ITX Audit Log map audit setting is enabled on the WebSphere DataPower SOA Appliance for Interoperability Test Service, the audit log that the Appliance generates contains the same content as the map audit log file that the map running locally generates. If a problem occurs with the Interoperability Test Service that prevents DataPower map from running, and the ITX Audit Log map audit setting is enabled on the WebSphere DataPower SOA Appliance, the audit log that the Appliance generates contains different content, with dummy data, than the audit log that the map running locally generates.

The DataPower audit log that the WebSphere DataPower SOA Appliance generates is saved as you specified in the *MapAudit->AuditLocation* setting on the DataPower map. If the *MapAudit->AuditLocation* setting on the DataPower map is set to Memory, the DataPower audit log is saved in the map directory and named after the executable map (*map_name.log*).

The audit log file generally has content, regardless of a successful execution. The contents of this file can be viewed in the Audit Log view, as well as in a text editor.

Troubleshooting for the XML Management Interface service

To troubleshoot DataPower maps deploying to a DataPower appliance using the XML Management Interface service, in addition to viewing the WebSphere Transformation Extender Design Studio log (.log) file located in the *your_workspace/.metadata* folder, for specific DataPower information, view the DataPower log files.

If the XML Management Interface service detects an error, the Problem Occurred window opens. Click OK to close the window, or Details to expand the window and view the error message.

For more information about the DataPower log files and how to view them, see the *IBM WebSphere DataPower SOA Appliances: Problem Determination Guide* documentation that is published on the [IBM DataPower Support website](#).

Standards Processing Engine

When Standards Processing Engine (SPE) is installed on IBM Transformation Extender, you can use IBM Transformation Extender maps to de-envelope, transform, and envelope documents and to validate document compliance with data standards.

You install SPE on Design Studio to test and deploy IBM Transformation Extender maps, IBM Sterling B2B Integrator maps, XSLT maps, XML schemas, and DTDs on SPE.

- **[Configuring Map Designer for SPE](#)**

To configure Map Designer to work with the Standards Processing Engine (SPE), set up the connection to the SPE database and configure the test and deployment options.

- **[Testing a map on SPE and displaying results](#)**

To test-run a map on Standards Processing Engine, right-click the map in the appropriate view and click Run on Standards Processing Engine. To display the test results of XSLT and IBM Sterling B2B Integrator maps, right-click the map in the appropriate view and click View Run Results. IBM Transformation Extender map results display in the input and output cards as usual.

- **[Deploying a map to SPE](#)**

To deploy maps and other artifacts to Standards Processing Engine (SPE), right-click the artifact in the appropriate view and click Deploy to Standards Processing Engine Repository.

Configuring Map Designer for SPE

To configure Map Designer to work with the Standards Processing Engine (SPE), set up the connection to the SPE database and configure the test and deployment options.

- **[Creating a project and importing maps](#)**

A project organizes maps and artifacts in Design Studio. You can create a project for the maps and artifacts that run on the Standards Processing Engine (SPE). Import any maps that are not created by Design Studio (such as XSLT files and schemas) into the project.

- **[SPE database connection](#)**

The Standards Processing Engine (SPE) database stores maps and other artifacts. Design Studio supports a default database as well as other SPE databases.

- **[SPE database deployment options](#)**

Configure the Standards Processing Engine (SPE) deployment options to identify and organize the artifacts that you deploy from Design Studio to the SPE database.

- **[SPE test options](#)**

To run on the Standards Processing Engine (SPE), IBM Sterling B2B Integrator maps and XSLT maps require input and output files. You can configure default input and output files to avoid supplying the file names each time you test-run the map. You also can specify the name of a .properties file to use when you test-run a IBM Sterling B2B Integrator map.

- **[SPE Logging options](#)**

Use the Design Studio Standards Processing Engine preferences to configure the logging options that are in effect when you deploy an artifact to SPE or test-run a map on SPE.

Creating a project and importing maps

A project organizes maps and artifacts in Design Studio. You can create a project for the maps and artifacts that run on the Standards Processing Engine (SPE). Import any maps that are not created by Design Studio (such as XSLT files and schemas) into the project.

1. Start Design Studio.

2. Create a project:

- a. Select File > New > Extender Project.
- b. Specify a unique project name.
- c. Use the default path for the project, or clear the Use default location field and browse to another directory.
- d. Click Finish.

3. Import a map to the project:

- a. Click File > Import.
- b. Expand General.
- c. Click File System and click Next.
- d. Browse to the directory that contains the map or other artifact to import and click OK.
The source directory and its contents display.
- e. Select the maps and artifacts to import.
- f. Click Browse to select a folder in the current project to copy the artifacts into.
- g. Optional: Select the overwrite or folder structure options.
- h. Click Finish.

SPE database connection

The Standards Processing Engine (SPE) database stores maps and other artifacts. Design Studio supports a default database as well as other SPE databases.

To use Design Studio to test maps on SPE or deploy artifacts to SPE, you can configure a database in the Design Studio Preferences, or you can use the default database.

When you configure a database, you must supply the SPE database URL, user name, and password to connect Design Studio to the database. If you specify the user name and password in advance in the Manage Repository window, they are automatically in effect when you test on or deploy to the database on SPE. If you do not specify them in advance, you must supply the database user name and password each time you test on or deploy to the database on SPE.

You do not need to configure a URL, user name, or password to test or deploy maps to the SPE default database. The default database uses the values in the SPE dbprops.cfg file.

- [Configuring SPE database connections](#)

Design Studio can test-run maps and deploy artifacts to the Standards Processing Engine (SPE). Use the Design Studio preferences to configure the connection to an SPE database, or use the default SPE database.

Configuring SPE database connections

Design Studio can test-run maps and deploy artifacts to the Standards Processing Engine (SPE). Use the Design Studio preferences to configure the connection to an SPE database, or use the default SPE database.

These steps configure a connection to an SPE database other than the default SPE database. The default SPE database displays `DEFAULT` in the Name field, is always available, and requires no configuration.

1. Start Design Studio.
2. Click Window \rightarrow Preferences.
3. In the navigation pane, expand Transformation Extender and click Standards Processing Engine.
4. In the Repository section, click Manage to add, edit, or delete a database connection.
Note: You cannot edit or delete the default database connection.
5. In the Name field, enter a label for the database connection, for example, "Test" or "Production."
(The Name field displays `DEFAULT` when the SPE default database connection is in effect.)
6. In the URL field, enter either:
 - The URL of the SPE database in JDBC format. For example:

```
jdbc:derby://localhost:1527/spe  
jdbc:oracle:thin:@oracledb6.acme.com:1527:ora11  
jdbc:db2://db2db9.acme.com:50000/spetest
```

- The URL of a configuration file that contains the JDBC/OpenJPA connection parameters of the SPE database. For example:

```
file:///C:/oracledb.cfg  
file:///C:/SPEInstall/dbprops_SQLServer2012.txt
```

7. Specify the SPE database user name and password in one of the following ways:
 - Enter the information in the User Name and Password fields.
 - Include the user name and password in the URL for the SPE database. For example:

```
jdbc:derby://localhost:1527/spe;user=admin1;password=sec12cde
```

- Include the user name and password in the URL by specifying the `{dbuser}` and `{dbpswd}` variables. For example:

```
jdbc:derby://localhost:1527/spe;user={dbuser};password={dbpswd}
```

With this configuration, Design Studio prompts you to enter a user name and password each time you test-run a map or deploy an artifact to SPE.

- Specify the user name and password by specifying the `{dbuser}` and `{dbpswd}` variables in the User Name and Password fields. With this configuration, Design Studio prompts you to enter a user name and password each time you test-run a map or deploy an artifact to SPE.

8. In the Driver Name field, enter the name of the JDBC driver that connects to the SPE database.

9. Optional: In the Schema Name field, enter the name of the SPE database schema.

10. Click one of the following:

Update

To update an existing SPE database connection or add a new one.

Delete

To delete an existing SPE database connection.

Finish

To update or add an SPE database connection and close the Manage Repository window without a confirmation prompt.

Close

To cancel your changes and close the Manage Repository window.

SPE database deployment options

Configure the Standards Processing Engine (SPE) deployment options to identify and organize the artifacts that you deploy from Design Studio to the SPE database.

Tenant Identifier

Associates an identifier with the artifacts that you check in, and creates a segment in the database for the artifacts that are associated with the identifier. The tenant identifier is sometimes called the reference identifier.

Make checked in resource the default

You can store multiple versions of artifacts and maps in the SPE database. SPE uses the default version when you invoke the `ENVELOPE`, `DEENVELOPE`, or `TRANSFORM` command.

When you enable this option and deploy an artifact that already exists in the database, the new version becomes the default version in the database. If you do not enable the option and you deploy an artifact that already exists in the database, the new version is added to the database, but the current default remains the default version. If you do not enable the option and you deploy an artifact that does not exist in the database, the new artifact is added to the database and becomes the default version.

You can use the SPE trading partner user interface to change the default version of a map or artifact in the SPE database.

Use default comments

Enable this option to associate a pre-configured comment with an artifact that you deploy to the database. The comments display when you view the artifacts with SPE Trading Partner user interface.

Click Show Comments to display or edit the check-in comment. You can enter freehand comments, or you can use the variables in any order to dynamically create comments at check-in time. If the comment exceeds 255 characters when the variables are resolved, Design Studio truncates the comment.

For example, when you specify the following check-in comment as the default comment:

```
Test of {map_schema_dir} {map_schema_name} on {timestamp}
```

The check-in comment in the database looks similar to:

```
Test of install_dir 8.4.1 PreProcessEDI on  
Fri Oct 11 09:26:14 EDT 2013
```

{timestamp}

Adds the current time and date to the check-in comment.

{user_name}

Adds the system account user name to the check-in comment. The system account user name is configured in the Microsoft Windows system properties.

Note that this user name can be different from the SPE database user name.

{project_name}

Adds the name of the project from which the artifact is deployed to the SPE database.

{map_schema_dir}

Adds the directory of the map or schema on the local file system to the check-in comment.

{map_schema_name}

Adds the name of the map or schema that you are deploying to the check-in comment.

SPE test options

To run on the Standards Processing Engine (SPE), IBM Sterling B2B Integrator maps and XSLT maps require input and output files. You can configure default input and output files to avoid supplying the file names each time you test-run the map. You also can specify the name of a .properties file to use when you test-run a IBM Sterling B2B Integrator map.

- **Configuring default input and output files**

You can configure default input and output files to use each time you test-run XSLT and IBM Sterling B2B Integrator maps on the Standards Processing Engine. You can select default files from the Design Studio project or the file system, or generate the file names at run time. Without default file names, you must enter the name of the input and output file each time you run an XSLT or IBM Sterling B2B Integrator map on SPE.

- **Setting Sterling B2B Integrator translator properties**

You can specify the translator properties that are to be in effect when you test-run a IBM Sterling B2B Integrator map on the Standards Processing Engine (SPE). Use the Configuration file field to enter the .properties file name, or browse to the file in the project or on the file system. Design Studio passes the properties through SPE to the IBM Sterling B2B Integrator translator. The properties are available when the map runs.

Configuring default input and output files

You can configure default input and output files to use each time you test-run XSLT and IBM Sterling B2B Integrator maps on the Standards Processing Engine. You can select default files from the Design Studio project or the file system, or generate the file names at run time. Without default file names, you must enter the name of the input and output file each time you run an XSLT or IBM Sterling B2B Integrator map on SPE.

1. Start Design Studio.

2. Click Window> Preferences.

3. In the navigation pane, expand Transformation Extender and click Standards Processing Engine.

4. Click Use default input and output files for XSLT and Sterling B2B Integrator maps.

5. Specify the default file name:

- Generate the input and output file names at run time:

In each Default file field, use the down-arrow to select a predefined name-pattern or click Custom to enter your own pattern. You can use the following variables in the file name:

{map_dir}

Substitutes the directory of the map that uses the file as a source or a target.

{map_name}

Substitutes the name of the map that uses the file as a source or a target.

- Select an input or output file from the current project:

Click Custom and browse to the folder in the project where the default file is. Use the filter field to show only the folders that match the filter text.

- Select an input or output file from the local file system:

Click Custom>Browse and enter the name in the File field, or browse to the file on the file system.

Setting Sterling B2B Integrator translator properties

You can specify the translator properties that are to be in effect when you test-run a IBM Sterling B2B Integrator map on the Standards Processing Engine (SPE). Use the Configuration file field to enter the .properties file name, or browse to the file in the project or on the file system. Design Studio passes the properties through SPE to the IBM Sterling B2B Integrator translator. The properties are available when the map runs.

SPE Logging options

Use the Design Studio Standards Processing Engine preferences to configure the logging options that are in effect when you deploy an artifact to SPE or test-run a map on SPE.

Generate Session Log

Generates a log file when you test-run a map or deploy an artifact to SPE. The log file is created in the artifact directory on the file system with the name *artifact_name.slg*. If you do not enable this option, Design Studio does not produce a session log.

Append to an existing session log

Appends a new log to an existing log file, if one exists, or creates a new log. When this option is not enabled, the new log overwrites an existing log.

Logging Level

Controls the level of information that is logged. The Severe, Warning, and Information options correspond to Java™ logging levels and represent descending message severity and importance. The All option tracks all messages.

The Debug option logs exceptions and other information from both IBM Transformation Extender and SPE. The Debug option produces a log of Transformation Extender and SPE information even when no exceptions occur.

Click Project > Clean to delete session log files.

Testing a map on SPE and displaying results

To test-run a map on Standards Processing Engine, right-click the map in the appropriate view and click Run on Standards Processing Engine. To display the test results of XSLT and IBM Sterling B2B Integrator maps, right-click the map in the appropriate view and click View Run Results. IBM Transformation Extender map results display in the input and output cards as usual.

Table 1. Testing on SPE. Design Studio views for testing a map on SPE and displaying test results

Map type:	Test-run the map from:	Display the test results from:
Executable Transformation Extender map	<ul style="list-style-type: none">Outline viewComposition view	Map input and output cards
Compiled (for Microsoft Windows) Transformation Extender map	<ul style="list-style-type: none">Navigator viewExtender Navigator view	Map input and output cards
IBM Sterling B2B Integrator map	<ul style="list-style-type: none">Navigator viewExtender Navigator view	<ul style="list-style-type: none">Navigator viewExtender Navigator view
XSLT map	<ul style="list-style-type: none">Navigator viewExtender Navigator view	<ul style="list-style-type: none">Navigator viewExtender Navigator view

Deploying a map to SPE

To deploy maps and other artifacts to Standards Processing Engine (SPE), right-click the artifact in the appropriate view and click Deploy to Standards Processing Engine Repository.

Table 1. Deploying to SPE. Design Studio views for deploying artifacts to SPE

Map type:	Map and artifact file extension:	Deploy from:
Executable Transformation Extender map	.mms	<ul style="list-style-type: none">Outline viewComposition view
Compiled Transformation Extender map	<ul style="list-style-type: none">.mmc.aix.hpi.lnx.mvs.pxn.sun.znx	<ul style="list-style-type: none">Navigator viewExtender Navigator view
IBM Sterling B2B Integrator map	<ul style="list-style-type: none">.map.txo.ltx	<ul style="list-style-type: none">Navigator viewExtender Navigator view

Map type:	Map and artifact file extension:	Deploy from:
<ul style="list-style-type: none"> • XSLT map • XML schema • XML DTD file 	<ul style="list-style-type: none"> • .xsl • .xslt • .xsd • .dtd 	<ul style="list-style-type: none"> • Navigator view • Extender Navigator view

Global transaction management

Although this functionality is referred to as global transaction management, it can also be referred to as distributed transaction management or two-phase commit architecture.

Global transaction management is the monitoring of transactions that can include operations on two or more different data sources. This feature of transaction processing enables data resources to be returned to a pre-transaction state if some error occurs. Either all data resources are updated or none are. The advantage of this strategy is that data resources remain synchronized and data remains consistent.

- [Components](#)

Components

There are three components commonly found in global transaction management models:

- Client Application
- Transaction Manager
- one or more Resource Managers
- [Client application](#)
- [Transaction manager](#)
- [Resource managers](#)

Client application

The role of the client application in global transaction management is similar to what it was for local transaction management. It will initiate and complete transactions and perform standard data operations, as in getting messages and updating rows, upon data sources included in this transaction. However, with global transaction management being implemented, the main difference is when the transactions are either committed or rolled back as monitored by a transaction manager.

In this global transaction management model, client applications are those adapters that support this functionality. For more information about the adapters currently supported, refer to ["Supported Adapters"](#).

After the data operations have been successfully completed by the client applications, the transaction manager is notified to commit the transaction. For an example of the entire process, refer to ["Flow Example"](#).

Transaction manager

The role of the transaction manager is to serve as a coordinator for all of the components involved in the global transaction and to keep track of all of the steps involved in the transactions. The transaction manager works with all of the resource managers to reach a consistent decision as to whether to commit or cancel the transaction.

If all of the resource managers are ready and in unanimous agreement, the transaction manager instructs them all to commit their transactions. If any one of the resource managers involved in this global transaction is not ready or able to commit, or does not respond to the transaction manager request, *all* of the resource managers are instructed by the transaction manager to roll back their transactions.

In summary, there are three different results that the transaction manager may find on a local level:

- The client application either commits or cancels the transaction.
- One of the resource managers cancels the transaction.
- A failure occurs, for example, there is an interruption on the network.

In all of the situations, the transaction manager must signal the resource managers to either commit or roll back the entire transaction. If any failure in the system occurs, the transaction would be rolled back.

- [Transaction specification types](#)
- [Supported Third-party Transaction Managers](#)
- [Supported adapters](#)

Transaction specification types

The communication between the three components involved in global transaction has been defined in specifications. The two protocols supported in the IBM Transformation Extender implementation of the global transaction management functionality are:

- XA/TX protocol
- OLE transactions specification protocol

There are differences in the way that each of these protocols is used in the transaction management models.

XA/TX protocol

The XA/TX protocol consists of two X/Open specifications: XA and TX.

The XA interface is a bi-directional interface between a transaction manager and a resource manager. Both managers must implement some functions from this interface. These managers use functions from the XA interface to communicate during the transaction preparation and commit activities.

The TX interface is used by the client application to contact the transaction manager. Only the transaction manager can implement this interface. This is used by the client application to notify the transaction manager that the transaction has been successfully committed, or not. If there was an error in the commit, the transaction manager takes over the data recovery portion.

Most of the commonly used resource managers support the XA interface and can be called XA/TX-compliant resource managers. However, there are only a few XA/TX-compliant transaction managers that are widely accepted or acknowledged.

OLE transactions specification protocol

The OLE Transactions specification protocol is a protocol developed to support the COM model and must be used on Windows platforms.

Supported Third-party Transaction Managers

The third-party transaction managers currently supported are:

- **IBM® MQ Transaction Manager (MQS TM)**

This manager is installed as part of the IBM MQ product and supports the XA/TX protocol.

- **Microsoft Distributed Transaction Coordinator Transaction Manager (MSDTC TM)**

This manager is installed along with Microsoft Windows operating systems and supports the OLE Transactions specification protocol.

Supported adapters

The following adapters support¹ global transactions: These adapters are listed with the transaction manager required for each. For operating system and version-specific information, see the system requirements (www.ibm.com/software/integration/wdatastagtx/requirements/requirements.html). See also the [release notes](#).

Adapter

Transaction Manager

IBM® WebSphere® MQ
MQS TM

MSMQ
MSDTC TM

Oracle

MQS TM

MS SQL Server
MSDTC TM

Oracle AQ

MQS TM

There are special configuration requirements to implement global transaction management. For more information, see ["Configuration Requirements"](#). For specific information about these adapters, see the respective documentation.

¹Only the IBM WebSphere MQ and MS SQL Server adapters can initiate and participate in global transactions. The MSMQ, Oracle, and Oracle AQ adapters can participate only in global transactions initiated by the MS SQL Server or IBM WebSphere MQ adapters, respectively.

Resource managers

Resource managers are data source managers, such as database management systems (DBMSs), that participate in a global transaction. These resource managers track their activities at the local level in transaction log files. Additionally, they can prepare recovery information and acquire all of the necessary locks and resources to either commit or roll back transactions when instructed by the transaction manager. Often, the old and new data is saved in storage so that any commit or rollback instruction can be performed accurately, even after any failure in the system, including applications, the transaction manager, or the resource manager. The resource managers work in cooperation with the transaction manager to provide the client application with a guarantee of atomicity of the global transactions.

Configuration requirements

Before global transaction management can be implemented, the following requirements must be met:

- A transaction manager must be installed and running on the same machine as the IBM Transformation Extender products are installed.

- The transaction manager being used must be established in the TransManager setting of the **config.yaml** file under the /runtime/Resource Manager path. See "[TransManager Setting](#)".
 - Each input and output card in any map involved with the global transaction scenario must include the Global Transaction (-GTX) adapter command.
 - [Installing and running transaction managers](#)
 - [TransManager setting](#)
 - [Input and output card settings](#)
 - [Global transactions in multiple threads](#)
-

Installing and running transaction managers

Because the supported transaction managers, MQS TM and MSDTC TM,) are third-party applications, refer to the respective documentation for more information about how to install and run these services. However, make sure that the transaction manager is installed on the same machine where IBM Transformation Extender products are installed when you are ready to begin your processing.

TransManager setting

There is a new setting under the /runtime/Resource Manager path of the **config.yaml** file for the transaction manager. MQS is the default setting.

```
TransManager: {MQS[,qm_name][,trace_full_path]|
  MSDTC[,machine_name][,trace_full_path]}
```

Value

Value	Description
MQS	This indicates that the IBM® WebSphere® MQ Transaction Manager is the transaction manager for this machine.
<i>qm_name</i>	This indicates the name of the queue manager to which to connect. If not entered, the default queue manager is assumed.
<i>trace_full_path</i>	This indicates the fully qualified path name of the trace file to be used to track all of the activities surrounding this transaction. The reason to specify this trace file is because it captures information such as connections to and from the transaction manager, and the start of global transaction. This information is not included in the trace files generated at the adapter or card level.
MSDTC	This indicates that the Microsoft Distributed Transaction Coordinator Transaction Manager is the transaction manager for this machine.
<i>machine_name</i>	This indicates the name of the machine upon which the master MSDTC service is running. This master service controls resources and all of the slave MSDTCs. If not entered, the default is assumed to be the local machine.

Input and output card settings

The Global Transaction (-GTX) adapter command must be specified in all of the input and output cards involved in this transaction. If the command is missing from any card, the transaction will be processed as a local transaction, not as a global one. Again, only those [adapters](#) supporting this functionality can use this command.

Global transactions in multiple threads

When global transactions are run concurrently in multiple threads, additional steps are required to configure the environment.

A new setting, **ShareConnectionThreads**, has been added to the **config.yaml** file under /runtime/Connections Manager to enable data operations on different resources to be executed in the same thread. If you want to enable multi-threaded global transactions, you must configure this setting as follows:

```
ShareConnectionThreads:1
```

The value of 1 indicates that the connections from the cards with the -GTX adapter command belong to the same thread.

When the IBM® WebSphere® MQ, Oracle, and Oracle AQ adapters are participating in global transactions managed by the MQS TM, the XAResourceManager stanza for Oracle must be properly configured. For UNIX systems, this stanza is located in the **qm.ini** file of the queue manager that is being used as a transaction manager; for Windows systems, it is located in the Windows Registry. On UNIX systems, the **qm.ini** file must be manually edited. In Windows, you can use the graphical user interface (GUI) to configure this stanza.

The **ThreadOfControl** setting of this stanza must be set to THREAD, as indicated below:

```
ThreadOfControl=THREAD
```

Additionally, in the Oracle XA open string, the following setting must be added:

```
Threads=True
```

For more information about the settings in the **config.yaml** file, refer to the *Launcher documentation*.

Examples

This documentation contains the following examples:

- a global transaction between IBM® WebSphere® MQ and Oracle
- trace log files created by both the MSDTC and the MQS transaction managers
- [Flow example](#)
- [Trace log file examples](#)

Flow example

Refer to the information about the events that occur for each number in this example. Processes A and B must be run on the same machine. Process C can also run on the same machine or on another machine.

- [Legend for the example](#)

Legend for the example

1. The transformation server begins to run the map and detects the presence of the `-GTX` adapter command in the first input card.
2. The transformation server reads the value of the `TransManager` entry in the `config.yaml` file. In this example, the value is `MQS,qm_name` which indicates to the transformation server that this card is going to participate in an XA type of global transaction. The other option for the `TransManager` value would be `MSDTC,machine_name`, which would indicate the OLE Transactions type of the global transaction.
3. The transformation server calls the IBM® WebSphere® MQ adapter and the adapter receives a request to initiate a new XA global transaction.
The transformation server passes the `qm_name` value from the `config.yaml` file to the IBM WebSphere MQ adapter. This is so that the queue manager selected to serve as the XA transaction monitor recognizes the adapter.
4. The IBM WebSphere MQ adapter connects to the `qm_name` queue manager.
The IBM WebSphere MQ queue manager returns the connection handle to the adapter.
5. The IBM WebSphere MQ adapter requests that the IBM WebSphere MQ queue manager start a new global transaction. The IBM WebSphere MQ queue manager has already recognized the process context of the application that has requested to start the transaction (*Process A*).
6. The IBM WebSphere MQ queue manager searches to determine if there is any other XA resource manager registered with the `qm_name` queue manager. In this search, the Oracle entry is discovered.
The IBM WebSphere MQ queue manager then calls the Oracle resource manager to inform it about the new global transaction started on behalf of the *Process A* process.
- The Oracle resource manager acknowledges this and the IBM WebSphere MQ queue manager notifies the IBM WebSphere MQ adapter that a new global transaction started.
- The IBM WebSphere MQ adapter reports this to the transformation server.
7. The transformation server continues to run the map. The first input card is IBM WebSphere MQ.
The transformation server requests that the IBM WebSphere MQ adapter connect to the queue manager and queue specified in the adapter command of the card (`-QMN QMNAME -QN QUEUE1`).
8. The IBM WebSphere MQ adapter recognizes that this card belongs to a global transaction because the `-GTX` adapter command was specified. Because of this, instead of connecting again to the IBM WebSphere MQ queue manager, it uses the connection handle obtained in Step 4. Therefore, if the `-GTX` adapter command is used, the `-QMN` adapter command is irrelevant because the IBM WebSphere MQ adapter will use the `qm_name` queue manager.
9. The transformation server requests that the IBM WebSphere MQ adapter get a message.
10. The IBM WebSphere MQ adapter sends a `GET` request to the IBM WebSphere MQ queue manager.
The IBM WebSphere MQ queue manager checks the process context of the application and gets the message within the transaction started on behalf of this same process.
- The IBM WebSphere MQ queue manager returns the message to the adapter and the adapter passes it back to the transformation server.
11. The transformation server now examines the output card. It requests that the Oracle adapter connect to the Oracle resource manager.
12. The Oracle adapter recognizes that it is participating in a global transaction, because of the presence of the `-GTX` adapter command. Instead of using the regular connection mechanism, the Oracle adapter calls special Oracle XA functions to obtain the connection handles.
13. The Oracle resource manager checks the process context of the application and it recognizes that this application started the global transaction, as communicated by the IBM WebSphere MQ queue manager previously in Step 6.
The Oracle resource manager returns a special connection handle to the Oracle adapter. Operations performed on this connection will belong to the global transaction.
14. The transformation server passes data from the input card, an IBM WebSphere MQ message, to the output card.
15. The transformation server issues a request to the Oracle adapter to store this data in the database table.
16. The Oracle adapter calls the Oracle resource manager with the `INSERT` request, using the connection handle obtained in Step 13.
17. The Oracle resource manager stores data in the table and notifies the Oracle adapter, which then informs the transformation server that the database output operation was performed successfully.
18. The mapping was successful and the transformation server calls the IBM WebSphere MQ adapter to commit the global transaction.
19. The IBM WebSphere MQ adapter calls the IBM WebSphere MQ queue manager with the `COMMIT` request.
20. The IBM WebSphere MQ queue manager knows that this process started the global transaction and that it now should commit it.
The IBM WebSphere MQ queue manager and the Oracle resource manager now start working together to commit the transaction. They exchange information using the XA two-phase commit protocol to ensure that either everything is committed, message retrieved and data inserted in the table, or that everything is rolled back.

No partial outcome is possible. The XA protocol ensures this with its functions and the help of the transaction logs managed both by the IBM WebSphere MQ queue manager and the Oracle resource manager.

When the committing process is done, the IBM WebSphere MQ queue manager informs the IBM WebSphere MQ adapter about the successful commit and the IBM WebSphere MQ adapter passes this information to the transformation server.

21. The map has completed successfully.

Trace log file examples

There are two examples of trace log files:

- IBM® WebSphere® MQ (MQS TM)
- Microsoft Distributed Transaction Coordinator (MSDTC TM)
- [MQS TM trace file example](#)
- [MSDTC TM trace file example](#)

MQS TM trace file example

```
<2396-1852>: GLOBAL TX TRACE ON
<2396-1852>: [m4mqcConnectToTM]
<2396-1852>: | [intm4mqcObtainGtxThreadContext]
<2396-1852>: | | Created GTX thread context for the thread 1852
<2396-1852>: | [intm4mqcObtainGtxThreadContext] (rc = 0) OK
<2396-1852>: | Connected successfully to transaction/queue manager
<2396-1852>: | Transaction ID [QM handle] = 0x02F49440
<2396-1852>: [m4mqcConnectToTM] (rc = 0) OK
<2396-1852>: [m4mqcStartGlobalTX]
<2396-1852>: | [intm4mqcObtainGtxThreadContext]
<2396-1852>: | | The existing GTX thread context was found
<2396-1852>: | [intm4mqcObtainGtxThreadContext] (rc = 0) OK
<2396-1852>: | Global transaction started
<2396-1852>: [m4mqcStartGlobalTX] (rc = 0) OK
<2396-1852>: [m4mqcEndGlobalTX]
<2396-1852>: | [intm4mqcObtainGtxThreadContext]
<2396-1852>: | | The existing GTX thread context was found
<2396-1852>: | [intm4mqcObtainGtxThreadContext] (rc = 0) OK
<2396-1852>: | Transaction committed successfully
<2396-1852>: [m4mqcEndGlobalTX] (rc = 0) OK
<2396-1852>: [m4mqcDisconnectFromTM]
<2396-1852>: | [intm4mqcObtainGtxThreadContext]
<2396-1852>: | | The existing GTX thread context was found
<2396-1852>: | [intm4mqcObtainGtxThreadContext] (rc = 0) OK
<2396-1852>: | Disconnected successfully from transaction/queue manager
<2396-1852>: [m4mqcDisconnectFromTM] (rc = 0) OK
<2396-2004>: Freeing GTX thread context for the thread 1852
<2396-2004>: GLOBAL TX TRACE OFF
```

MSDTC TM trace file example

```
<2076-1252>: GLOBAL TRACE ON
<2076-1252>: [m4oledbConnectToTM]
<2076-1252>: | [intm4oledbObtainGtxThreadContext]
<2076-1252>: | | Created GTX thread context for the thread 1252
<2076-1252>: | [intm4oledbObtainGtxThreadContext] (rc = 0) OK
<2076-1252>: | Obtained IITransactionDispenser
<2076-1252>: [m4oledbConnectToTM] (rc = 0) OK
<2076-1252>: [m4oledbStartGlobalTX]
<2076-1252>: | [intm4oledbObtainGtxThreadContext]
<2076-1252>: | | The existing GTX thread context was found
<2076-1252>: | [intm4oledbObtainGtxThreadContext] (rc = 0) OK
<2076-1252>: | Global transaction started
<2076-1252>: | Transaction ID [&IITransaction] 0x04190050
<2076-1252>: [m4oledbStartGlobalTX] (rc = 0) OK
<2076-1252>: [intm4oledbObtainGtxThreadContext]
<2076-1252>: | The existing GTX thread context was found
<2076-1252>: [intm4oledbObtainGtxThreadContext] (rc = 0) OK
<2076-1252>: [m4oledbEndGlobalTX]
<2076-1252>: | [intm4oledbObtainGtxThreadContext]
<2076-1252>: | | The existing GTX thread context was found
<2076-1252>: | [intm4oledbObtainGtxThreadContext] (rc = 0) OK
<2076-1252>: | Transaction committed successfully
<2076-1252>: [m4oledbEndGlobalTX] (rc = 0) OK
<2076-1252>: [m4oledbDisconnectFromTM]
<2076-1252>: | [intm4oledbObtainGtxThreadContext]
<2076-1252>: | | The existing GTX thread context was found
<2076-1252>: | [intm4oledbObtainGtxThreadContext] (rc = 0) OK
<2076-1252>: | Released ITransasctionDispenser
<2076-1252>: [m4oledbDisconnectFromTM] (rc = 0) OK
```

Performance overview

This documentation contains general performance tips and recommendations to improve performance.

IBM® Transformation Extender offers the flexibility and power to solve integration needs. This flexibility provides a user with a variety of options to develop their transformation processes and the power to accomplish their objectives in the most optimum ways.

This documentation contains general performance tips and recommendations to improve performance. The purpose of this documentation is to present strategies for leveraging the flexibility and power of IBM Transformation Extender so that transformation needs can be met while achieving reasonable performance objectives.

Note that not all recommendation will apply to all platforms.

General performance overview

This documentation describes what takes place during product execution and presents performance tips and recommendations regardless of the platform on which a specific IBM® Transformation Extender product is running.

This information provides you with additional background to assist in defining performance objectives.

There are several platform-independent topics for which background information, performance tips, and recommendations are addressed:

- [Performance Factors](#)
- [Understanding Command Server Operation](#)
- [Understanding Launcher Operation](#)
- [Map Performance Tuning Tips](#)
- **Performance measurement factors**
When analyzing how well an application has performed, there are potentially a large number of measurements that can be used to gauge workload efficiency in terms of the specific resources the application used.
- [Command Server operation overview](#)
The first step in trying to achieve performance improvements running the Command Server in your environment is to understand Command Server operation.
- [Launcher operation overview](#)
The first step in trying to achieve performance improvements running the IBM Transformation Extender Launcher is to understand its operation.
- [Map performance and tuning best practices](#)
Many functions available to users for developing IBM Transformation Extender maps are very powerful and can perform difficult and complicated tasks. Some functions may sacrifice performance for powerful capabilities, and it is important to understand the implications of these.
- [Merging and Pruning Schemas and Type Trees](#)
You can merge and prune schemas and type trees. Always practice save file management. Copy the source type tree to another name before you make any modifications.

Performance measurement factors

When analyzing how well an application has performed, there are potentially a large number of measurements that can be used to gauge workload efficiency in terms of the specific resources the application used.

The most common resource used in measuring performance is time. Memory, disk space, and other resource usage are often used, too, because of their impact on performance.

Comparisons can be made between workload and the various resources that impact performance. In this documentation, when the word *performance* is used without a qualifier, it refers to workload/time comparisons. This shortcut is used since this general resource usage is the most common. Other types of comparisons will be explicitly noted by using a qualifier with the word *performance*. The phrase, *all types of performance*, will refer to any workload/resource usage comparison, such as memory and disk space comparisons to workload as well as execution-time comparisons.

For every different application execution, there are a wide variety of performance factors that influence all types of performance for the Command Server and Launcher. The significance of each factor varies for every application execution. Most of these factors are dependent on the performance of other factors; few of the factors are completely independent. Accounting for these inter-relationships among performance factors increases the difficulty in tuning the performance of an application.

Although there can be a wide variety of performance factors, the following list shows a subset of the more significant and commonly-encountered performance factors when running applications using IBM® Transformation Extender maps. They represent the specific potential areas that you can target for analysis when tuning the performance of your applications. They are listed below organized by logical groupings.

- Mapping
 - Type design
 - Map design
- Hardware
 - CPUs (ISA, MHz, #)
 - Memory architecture
 - Physical memory
- Load
 - Memory contention
 - CPU contention
 - I/O contention
- Compiler
 - C/C++ runtime
 - Code generation

- Input/Output
 - File size
 - File/Disk configuration
 - File system
- Map execution
 - Paging
 - Workspace options
 - Burst/Integral
- Operating system
 - Virtual memory
 - Context switching
 - Load time

Many application executions have a specific constraining factor that impacts performance more so than other factors. Reducing the impact of that factor can provide a more substantial performance improvement than committing time and effort in examining other factors. Often, identifying the constraining factor requires additional time and effort. That additional time is balanced by a more significant performance improvement, which might be sufficient to preclude further analysis of those other factors.

- **Load and performance measurements**

There are many factors that can impact system load and performance, and these can be measured with the goal of improving performance.

- **Throughput and CPU measurements**

There are many factors that can impact system throughput and CPU performance, and these can be measured with the goal of improving performance.

Load and performance measurements

There are many factors that can impact system load and performance, and these can be measured with the goal of improving performance.

When measuring performance, it is important to account for extraneous load. If a system has a significant amount of usage outside the process being measured, the elapsed time for the process is likely to be longer than when the system has less use. Even though the process has much of the same resource needs, such as memory or CPU processing power, in a busy or a quiet environment, heavy competition for those resources can stall execution. For this reason, elapsed time, or wall time as an example, is not the most appropriate measure of performance for a heavily used system.

Performance metrics should attempt to isolate the process being monitored. Better alternatives to elapsed time include user/system time or CPU time. When measuring CPU use, the proportion used by each application should be noted. Lastly, performance metrics should be done on a system with as little extraneous load as possible.

Occasionally, the combined user/system time or the CPU time of an execution will be acceptable, but the elapsed time is too high. In such cases, reducing the impact of extraneous load should reduce the gap between the two times. The impact results from a scarcity of one or more resources. Whether the scarce resource is CPU processing power or memory, reducing the impact of external load requires increasing the amount of the contested resources available.

Increasing the amount of a specific resource can usually be done in a number of ways.

For example, if a system is at 100% CPU use because of extraneous load, there are a number of options for reducing that load:

- Tune the application. On a system constrained by CPU power, tuning the application for CPU usage will reduce its overall run time.
- Tune other applications. Reducing the CPU needs of the applications making up the extraneous load will free up some computational power. You should tune the focal application before tuning the other applications. For best results, request assistance from your systems administrator.
- Reduce the application load. Reducing the number of applications running will free up CPU power for the focal application. Unfortunately, this is not an option in many cases.
- Obtain additional processors. Additional processors will increase the amount of CPU power available. Assuming that the extraneous load remains constant, this leaves more CPU power available for the focal application. The amount of improvement will depend on the applications running and the underlying operating system.
- Use faster processors. Replacing the existing processors with faster models will increase the amount of CPU power available.

Because every situation presents its own set of performance variables, each situation must be evaluated to determine the best option to use.

Throughput and CPU measurements

There are many factors that can impact system throughput and CPU performance, and these can be measured with the goal of improving performance.

The execution of a single map occurs in the context of a single thread. Therefore, additional processors will not improve the execution time of a single transformation. The Command Server executes a simple map in this fashion. Execution time measurements of a single transformation performed with the Command Server will only account for the computational power of a single processor.

The affect additional processors have on the Command Server contrasts with the affect additional processors might have on the Launcher. Depending on how event triggers occur, separate transformations can execute on multiple processors concurrently. Even in this situation, each transformation still runs in a single thread.

In gauging performance, it is important to choose a measure of performance appropriate to the architecture of the mapping process. The performance measurement should agree with the performance goal for the system as a whole. If the system includes a number of maps running in parallel on multi-processor hardware, transactional throughput is probably more important than the execution time of any single transformation.

For example, a given map might execute in 10 seconds. On a 4-processor box, 4 simultaneous executions of that same map might complete in 11 seconds. The transactional throughput is nearly four times that of the single execution scenario, but each of the 4 simultaneous executions still executed in 10 seconds.

Conversely, if the process involves a batch of data with a single map, map execution time might be the most appropriate primary measurement. Execution time is also quite appropriate when running a series of maps. In such cases, all of the work occurs in a single thread as previously mentioned.

Assuming that CPU processing power is the constraining factor in both cases, improving throughput and execution time require different approaches. Adding additional processors of the same type will not improve map execution time. Improving map execution time requires more computationally powerful (faster) processors. Conversely, additional processors with the same power can improve throughput. More processors of the same power can allow more maps to execute simultaneously.

This same principle applies when comparing map performance across hardware platforms. A hardware configuration might have more processors available, but if an individual processor lacks computational power, the execution time of a single map on that processor will suffer. However, even with weaker processors, this same configuration might provide better throughput. While computational power varies considerably among processors, clock cycle frequency is a loose predictor of relative computational strength.

Command Server operation overview

The first step in trying to achieve performance improvements running the Command Server in your environment is to understand Command Server operation.

- **[Command Server execution overview](#)**

When you want to achieve performance improvements when running the Command Server in your environment, you must understand Command Server execution.

- **[Command Server work and data files](#)**

Depending on the conditions of the transformation, such as input data file size and processing complexity, work and data files can significantly affect performance.

- **[Command Server paging settings](#)**

To minimize the memory footprint during transformation, IBM Transformation Extender keeps only a subset of the work and data file information in memory at a given time.

Command Server execution overview

When you want to achieve performance improvements when running the Command Server in your environment, you must understand Command Server execution.

After you have developed your map, you are ready to run it using the Command Server. During the map execution process, the transformation engine performs several operations. These operations generally fall under the following phases:

- Initialization
- Validation
- Transformation
- Finalization

- **[Command Server initialization](#)**

When you want to achieve performance improvements when running the Command Server in your environment, you must understand Command Server initialization.

- **[Command Server validation](#)**

When you want to achieve performance improvements when running the Command Server in your environment, you must understand Command Server validation.

- **[Command Server transformation](#)**

When you want to achieve performance improvements when running the Command Server in your environment, you must understand Command Server transformation.

- **[Command Server finalization steps](#)**

When you want to achieve performance improvements when running the Command Server in your environment, you must understand the steps that Command Server takes to finalize its processes after execution.

- **[Command Server execution time](#)**

When you want to achieve performance improvements when running the Command Server in your environment, you must understand the factors that can affect execution time.

Command Server initialization

When you want to achieve performance improvements when running the Command Server in your environment, you must understand Command Server initialization.

During initialization, the transformation engine loads maps and prepares data for validation. Adapters are loaded and initialized at this time. This phase of execution also includes allocation of memory for the paging subsystem.

Command Server validation

When you want to achieve performance improvements when running the Command Server in your environment, you must understand Command Server validation.

During validation, the transformation engine parses all the input data sources and identifies objects based on type definitions. The information about the objects within each input data source is stored in one or more temporary files called work files or in memory. Each work file locates and sets boundaries for objects within all input data sources.

Command Server transformation

When you want to achieve performance improvements when running the Command Server in your environment, you must understand Command Server transformation.

During transformation, the rules contained within the map are executed. If a map is designed to produce output, the result of this execution process is the expected output data. Map execution proceeds through the output cards, executing each contained rule.

Multiple input values or multiple combinations of data values might cause rules to be executed multiple times. If a rule calls a functional map, the rules contained within the functional map would be executed multiple times.

During transformation, both the workspace and the data sources are heavily used. Paging settings affect how much of these two information sources are available at any one time. You also have the option of creating the Burst and Data Audit Logs in this step.

Command Server finalization steps

When you want to achieve performance improvements when running the Command Server in your environment, you must understand the steps that Command Server takes to finalize its processes after execution.

During finalization, temporary work files used during transformation are deleted and other system clean-up tasks are performed. Optionally, the **Map Settings** and **Execution Audit** log creation also occur in this step.

Command Server execution time

When you want to achieve performance improvements when running the Command Server in your environment, you must understand the factors that can affect execution time.

Execution time can be affected by various factors. When most transformation processes are executed, the transformation itself consumes the bulk of execution times. Complex validation logic in type trees, especially restarts, can increase the proportion of execution time spent in validation.

The map execution process occurs in a single thread. By using a single thread of execution, the overhead of context switching is avoided in heavily-loaded systems. Excess context switching on heavily loaded systems can quickly dominate execution.

Command Server work and data files

Depending on the conditions of the transformation, such as input data file size and processing complexity, work and data files can significantly affect performance.

Work and data files are the two major sources of information used in the transformation phase of map execution. Data files are the input data specified at execution time. Work files created during validation contain data about the input and output data sources.

The data types defined by the transformation's type tree depend on the internal format of the data files and are ultimately determined by the business process. This internal format drives the validation phase. The storage characteristics of the data files also depend on the business process as well as the underlying operating system.

IBM® Transformation Extender offers a significant number of options for creating and using work files. Depending on the conditions of the transformation, such as input data file size and processing complexity, some of these options can significantly affect performance. [Reusing Work Files](#), [Paging](#) and [WorkSpace in Memory](#) present the performance impact of these options.

The size requirements for work files ultimately depend on the number of type instances found in the input data file and generated in the output data file. The number of type instances found in both cases depends greatly on the type definitions. To save work file space, type instances are only recorded in the workspace if needed later in the transformation process.

- [Reusing work files](#)

Depending on the conditions of the transformation, such as input data file size and processing complexity, work and data files can significantly affect performance. When possible, reusing work files can produce measurable performance gains.

Reusing work files

Depending on the conditions of the transformation, such as input data file size and processing complexity, work and data files can significantly affect performance. When possible, reusing work files can produce measurable performance gains.

Work files contain metadata about the various type objects found in each input data file. If the data content is static, this metadata can be reused in subsequent runs. Reusing work files allows the transformation engine to bypass the validation step for these sources. Reducing the number of steps results in a shorter overall execution time.

When possible, reusing work files can produce measurable performance gains, especially in input data files requiring complex validation logic. This option is ideal for map processes that make use of static data, such as lookup tables. Work file reuse is specified separately for each input card, either as an execution command or as an option in the Map Designer. Reusing work files with an execution command is done with the W option on the input card override (-I) execution command.

To reuse work files, you must generate a work file.

To generate a work file

1. Specify an input card override -I execution command for the corresponding input source for a compiled map with the (w) reuse option.
2. Execute the map.
A work file is generated.

Subsequent map executions with the same input override options will reuse the generated work file. As mentioned previously, this option is ideal if the data content is static. If the data content changes, the work file should be regenerated.

To regenerate a work file

1. Remove the old work file.

2. Specify an input card override `-I` execution command for the corresponding input source for a compiled map with the `(W)` reuse option.
3. Execute the map.
A work file is generated.

A generated work file can be used to determine the amount of workspace required for a given input source. This can be helpful when using workspace options such as `-WM`.

See the *Command Server* documentation and the *Execution Command* documentation for more information about work file reuse.

Command Server paging settings

To minimize the memory footprint during transformation, IBM® Transformation Extender keeps only a subset of the work and data file information in memory at a given time.

The total amount of memory available for storing data used in the transformation process is segmented into equal-sized portions called pages. Each page holds a contiguous section of either an input or a work file.

- **Paging and I/O usage**

Paging controls the amount of data available to the transformation portion of the map execution at any time. If a map rule requires a piece of data that is not in memory, the paging subsystem will fetch the appropriate data from disk. This can impact existing data in memory; if there are no unused pages, an existing page is swapped out of memory.

- **Scenario: configure paging by input file size**

The following test scenario was created to demonstrate how the input size might require a larger information window or page setting, so it does not impact performance.

- **Configuring optimal page subsystem settings**

The settings for the paging subsystem in IBM Transformation Extender can determine whether a transformation process runs bound by I/O bandwidth or by CPU capacity. These settings can impact system performance.

- **WorkSpace in Memory option**

The **WorkSpace in Memory** option can be a viable choice for reducing I/O usage during transformation if enough memory is available. With a smaller input data file, the transformation engine can attempt to process the work file entirely in memory.

Paging and I/O usage

Paging controls the amount of data available to the transformation portion of the map execution at any time. If a map rule requires a piece of data that is not in memory, the paging subsystem will fetch the appropriate data from disk. This can impact existing data in memory; if there are no unused pages, an existing page is swapped out of memory.

In situations where a great deal of information from various areas of the input are needed, increasing the number and size of pages available to the system can reduce I/O volume. Increasing the information window allows the map to process more information without loading pages from disk.

Paging portions of data files allow for processing a large transformation in a much smaller amount of memory. Choosing paging settings appropriate to the transformation can improve performance considerably.

The following paging-related activities affect performance:

- [Using the WorkSpace paging setting](#)
- [Setting information windows](#)
- [Knowing the size of the input](#)

Using the WorkSpace paging setting

In many maps, paging settings are explicitly specified in the maps themselves during design. If the designer of a map does not explicitly specify page settings, the page settings for the map will default to a page size of 64 KB and a page count of 8 KB. Optimal page size and count settings depend ultimately on the amount of workspace needed. Because the workspace requirements are not available at the beginning of execution, the estimated page size and count will not always result in an optimal execution time. In this situation or if the paging settings within a map are sub optimal, altering paging behavior at execution time might improve performance.

The primary method of altering runtime paging behavior during Command Server execution uses the **WorkSpace PageSize** (`-P`) execution command. With the `(-P)` command, the page size and count can be specified for a transformation. The page count determines the total number of pages available to the system for both data and work file usage. Half of the requested pages are allocated to each file usage. Page size determines the total amount of storage available on each page. Together, the two directly control the amount of information available at any given time, as an information window.

To determine the values for the page settings, you must understand how the transformation engine allocates memory.

When the **WorkArea** card setting on a map is set to **Memory**, the transformation engine will allocate memory as required for both data and workspace, with the page size controlled by the **PageSize** setting in the map.

When the **WorkArea** card setting on a map is set to **File**, the transformation engine will allocate memory to track the information written to file for both data and work space, with the page size controlled by the **PageSize** and **PageCount** settings.

As a result of how memory is allocated depending on the **WorkArea** card setting, if the **WorkArea** is set to **File**, and the **PageSize** and **PageCount** map settings are set to a high number, the transformation engine is more likely to exceed memory.

For most data transformations, a **PageSize** map setting of 1024 kilobytes is too large. Data transformations whose **PageSize** map settings are 64, 128, 256 or 384 kilobytes should be tested for performance results using sample data that mirrors the size and complexity of the data that will be used in your production environment. Performance will be impacted negatively if you use **PageSize** map settings that are lower or higher than these recommended settings.

When executing maps in the Launcher, the memory used at startup time includes the totals of all the maps included in the **.msl** files deployed, with the addition of the memory used by any cached maps.

Setting information windows

When specifying paging information, it is critical to understand the information requirements of the map. Certain usage patterns require a larger window into the data and the workspace than other usage patterns. If a paging setting does not provide a large enough window, the paging subsystem will generate additional I/O requests to meet the demands of the map.

Conversely, requesting too much memory through paging settings can have detrimental effects as well. If the amount of requested memory exceeds available physical memory, the underlying operating system will begin paging memory to disk. Selecting appropriate paging settings is a balance between the information needs of the map and the amount of available memory.

Setting an appropriate information window especially applies to series-consuming functions such as `EXTRACT()`, `LOOKUP()`, and `CHOOSE()`. Depending on context, series-consuming functions can require access to the entire data file to complete map execution. This contrasts with the much smaller requirements of simple sequential maps.

Knowing the size of the input

One significant factor in determining appropriate paging settings is the amount of data processed by a map. A larger amount of input might require more paging memory to attain optimal execution time. More paging memory results in a larger information window. The degree to which larger input might require a larger information window varies with the design of the map.

Scenario: configure paging by input file size

The following test scenario was created to demonstrate how the input size might require a larger information window or page setting, so it does not impact performance.

Two test cases were created and then executed using the same testing procedures.

- Maps: The same map is used in both tests.
- Inputs: There are two input data files.
 - Input 1 size: 3 MB
 - Input 2 size: 11 MB
- Process:
 - Map performed a number of operations such as a large number of `EXTRACT()` calls that spanned each of the input data files.
 - These requests mandated that a certain percentage of the input data be available at any given time for optimal performance.
- Objective:
 - Determine the page size to get the optimal execution time for each of the input data files.
- Test Case 1 Result:
 - Quantitative measurement of execution time at various page counts and sizes identified an optimal execution time at the following workpage settings for the 3 MB input data file: size 64 K and count 12. This is represented by `-P64:12`, where `-P` is **WorkSpace Paging**, 64 is the page size and 12 is the page count.
- Test Case 2 Result:
 - The same testing procedure found `-P64:53` to produce the optimal execution time for the 11 MB input data file.
- Conclusion:
 - Paging settings depend on input data size just as much as they do on the map rules. An optimal page setting for a given map and data file might not be optimal for the same map with a much larger or smaller data file.

Configuring optimal page subsystem settings

The settings for the paging subsystem in IBM® Transformation Extender can determine whether a transformation process runs bound by I/O bandwidth or by CPU capacity. These settings can impact system performance.

Behavior

The settings for the paging subsystem within IBM Transformation Extender can determine whether a transformation process runs bound by I/O bandwidth or by CPU capacity. A typical execution time/page count curve has two easily discernible parts, based on performance characteristics.

In a graph where the vertical coordinate measures execution time and the horizontal coordinate measures page count, at the low page count end of the graph, the execution time falls quickly as the page count increases. Here, the transformation is I/O bound. The data access patterns implicit in the map require more information than what is available in memory. Because of this, time spent driving I/O dominates execution time.

As page count increases, execution time decreases into a nearly flat portion of the graph. At that stage, the transformation is bound by the computational power of the executing processor. Increasing the amount of memory available through paging settings will probably not improve performance.

Tuning procedure

Finding the combination of page size and page count that optimizes performance is the goal of the following procedure. There are several steps to follow for tuning paging settings. The settings found for this procedure are specific to the behavior of the map and data used. Because of this, the data should represent future workloads as much as possible.

To tune page settings

1. Verify that the execution environment is stable. Execution times must be relatively repeatable. Run a few simple preliminary executions and time them to gauge the reproducibility of the timing procedure. If the resulting execution times are not similar to one another, then the execution environment is not stable enough for

- tuning paging settings. Improve reproducibility by reducing extraneous loads from other applications.
2. Select initial values for page size and count. The overall page size and count specify the total amount of memory available for paging data and workspace. This total amount of memory is allocated early in the execution process. A failure to allocate paging space will result in an early termination. Therefore, a memory request for C pages of S kilobytes should be expected to succeed, given available storage.

- Appropriate values for each setting vary highly on the data access patterns of the map and the design of the type tree. Not only is each setting a factor in execution time, but the overall amount of memory allocated also affects performance. The tuning procedure here will attempt to insulate page size from the effect of changing overall memory.
3. Execute the map and data to be tuned. Note the overall processing time and the processing time spent managing I/O during execution. Sub-optimal paging settings often result in an I/O bound transformation process.
4. Execute the map and data again with different paging settings. Vary the page size up or down. The page count should be adjusted so that the combined paging memory remains as constant as possible. For example, if you double the page size in the second run, the page count should be halved. Again, note the overall processing time and the processing time spent managing I/O during execution. The page count must still be small enough for the entire paging space to fit in the memory allocated to IBM Transformation Extender by the operating system.
5. Compare the collected times from both executions after the second execution completes.
6. Repeat this procedure with various page sizes to find the optimal page size for the map.
7. After an optimal page size is determined, a similar set of iterations to vary the page count should be performed, keeping the page size fixed.
If an increase in memory drastically reduces performance, it is likely that the overall amount of memory has started to throttle the operating system's virtual memory system. If this occurs, reduce the page count. The threshold at which throttling can occur depends heavily on the application load across the entire system. If a decrease in page count drastically reduces execution time, then the execution was probably I/O bound. Compare the difference in I/O usage between executions or the time used by the operating system. An increase indicates that the proportion of execution time devoted to handling I/O has increased. In this case, increase the page count.

Recent changes

Recent changes in the transformation engine altered the paging behavior of IBM Transformation Extender from release 6.7 onward. These recent changes simplify the tuning process by reducing the CPU overhead required to manage larger page counts. Previously, many maps had a single page count setting that would minimize execution time. Choosing a page count larger than this setting would require more CPU bandwidth. Choosing a smaller page count setting would require more I/O bandwidth, putting the execution into an I/O bound state.

WorkSpace in Memory option

The **WorkSpace in Memory** option can be a viable choice for reducing I/O usage during transformation if enough memory is available. With a smaller input data file, the transformation engine can attempt to process the work file entirely in memory.

Using this option:

- Significantly reduces the number of I/O requests issued across the execution.
- Produces optimal performance benefits when the work files do not exceed available physical memory. For more information about determining work file size, see the [Reusing Work Files](#).
- When producing work files that exceed the amount of available physical memory, will cause operating system level-memory management to simulate additional memory with disk storage. This will ultimately degrade performance.

Launcher operation overview

The first step in trying to achieve performance improvements running the IBM® Transformation Extender Launcher is to understand its operation.

- [Launcher and Command Server performance comparison](#)

The Launcher is similar to the Command Server in the way it executes maps. However, the Launcher offers multiple simultaneous transformations, each on its own execution thread.

Launcher and Command Server performance comparison

The Launcher is similar to the Command Server in the way it executes maps. However, the Launcher offers multiple simultaneous transformations, each on its own execution thread.

Triggering logic specified in the Launcher configuration determines the number of maps that can be executed at any one time.

Important:

The ability of the Launcher on a given platform to take advantage of multiple processors is dependant on the configuration of the underlying operating system. More information about Launcher configuration can be found in *Configuration Parameters* and *Launcher Settings* in the *Launcher* documentation.

By launching each transformation in its own thread, the Launcher can take advantage of multiple CPU resources concurrently. Using multiple CPU resources can increase the total throughput and transactional volume. While the transactional volume might increase, the execution time of a simple map, measured against how the Command Server would perform, will be unchanged or possibly longer. The improved throughput results from parallel execution of multiple maps and not faster execution of individual transformations.

Map performance and tuning best practices

Many functions available to users for developing IBM® Transformation Extender maps are very powerful and can perform difficult and complicated tasks. Some functions may sacrifice performance for powerful capabilities, and it is important to understand the implications of these.

It is important to understand when it is better to use one function over another that can perform the same task but does so differently. Each function has advantages and disadvantages and users have the flexibility to choose the function that best suits their specific transformation needs.

These are suggestions for tuning maps to achieve the best performance results through the various map functions. The functions fall under the following categories:

- [Data Searches](#)
- [Routing](#)

Also, the following utilities can be used to help you identify the areas in your maps that might be causing performance degradation and then you can make the appropriate changes to your maps to improve their performance:

- [Map Profiler \(dtxprof\)](#)
 - [Page Setting Assistant for Maps \(dtxpage\)](#)
 - [Data search performance best practices](#)
- There are many ways to develop IBM Transformation Extender maps that search through data. These different options provide varying balances between functionality and performance.
- [Routing](#)
 - [Map Profiler \(dtxprof\)](#)
 - [Page Setting Assistant for Maps \(dtxpage\)](#)

Data search performance best practices

There are many ways to develop IBM® Transformation Extender maps that search through data. These different options provide varying balances between functionality and performance.

The functions available to the map developer to do data searches are:

- EXTRACT()
- LOOKUP()
- SEARCHUP() or SEARCHDOWN()
- CHOOSE() - see POSITIONAL INDEXING in the table below

The following table contains the functions available to the map developer to do data searches.

Function	Use	Advantage	Disadvantage
EXTRACT()	When expected results are multiple objects.	One of the most powerful ways of searching through a data file.	Not the most efficient way to locate objects if the expected result set is only a single object.
LOOKUP()	Where expected results are a single object. Scans a data file object-by-object, looking for an object that matches the specified criteria.	More efficient than Extract().	The average amount of time required for search depends proportionally on the number of objects in the data file.
SEARCHUP() or SEARCHDOWN()	When the set of items to be searched is sorted.	Increased performance. Takes advantage of the sorted order of the data file by traversing the data file as a binary tree. Search time is proportional to $\log_2(N)$, where N is the number of data items.	Cannot be used for unsorted data files. The smaller the input data file, the less performance results benefit.
POSITIONAL INDEXING	When used with CHOOSE() and INDEX(), locates objects within a data file by position.	Executes repeated CHOOSE() requests and maintains a cache of position information for the indexed data file. The cache continues to be used as long as repeated calls to CHOOSE() reference the same type, and indexing is fast.	When repeated calls to CHOOSE() don't reference the same type, the cache isn't used, and indexing is not as fast.

- [Data search functions](#)

As you develop your IBM Transformation Extender maps to search through data, you should consider the input format and the requirements for the output. Based upon those considerations, you can make your decision on which search function to use to for optimum performance results.

- [Search functions reference](#)

This documentation contains quantitative comparisons of the data search functions. It presents results of comparing the LOOKUP() against the SEARCHUP() or SEARCHDOWN() functions.

Data search functions

As you develop your IBM® Transformation Extender maps to search through data, you should consider the input format and the requirements for the output. Based upon those considerations, you can make your decision on which search function to use to for optimum performance results.

The following table lists search usage scenarios with the recommended function that should give you the best performance results.

Usage

Search Function Recommendation

Single output data item is the expected result and the input data cannot be sorted.

LOOKUP ()
 Multiple output data items are expected result.
EXTRACT ()
 The set of items to be searched is or can be sorted.
SEARCHUP () OR SEARCHDOWN ()
 Same data type is being referenced (Positional Indexing).
 In cases where **CHOOSE ()** consistently references the same type, execution performance can exceed **SEARCHUP ()** and **SEARCHDOWN ()**.
CHOOSE ()

Search functions reference

This documentation contains quantitative comparisons of the data search functions. It presents results of comparing the **LOOKUP ()** against the **SEARCHUP ()** or **SEARCHDOWN ()** functions.

The primary difference between the functionality of **LOOKUP ()** and **SEARCHUP ()** or **SEARCHDOWN ()** is the ability to use a full logical expression in **LOOKUP ()**. For example, **LOOKUP ()** can be used to find the first item whose value is less than 3. **SEARCHUP ()** or **SEARCHDOWN ()** can only find an exact match.

Functionality aside, **LOOKUP ()** and **SEARCHUP ()** or **SEARCHDOWN ()** differ in their performance characteristics. A **LOOKUP ()** search operates by scanning the appropriate set of items in sequence, attempting to match the logical condition. On average, a **LOOKUP ()** search takes time proportional to the number of items in the set.

In contrast, **SEARCHUP ()** and **SEARCHDOWN ()** take advantage of sorted input to provide faster response. Searches with **SEARCHUP ()** or **SEARCHDOWN ()** use a binary search index on the items in the data series. This results in execution time proportional to the logarithm of the item count. In other words, doubling the number of items in the data series causes only an incremental increase in the execution time required for that search.

To achieve this response-time benefit, **SEARCHUP ()** and **SEARCHDOWN ()** require sorted input. Additionally, **SEARCHUP ()** and **SEARCHDOWN ()** can only be used to find an exact match. If these conditions are suitable for the map, then using **SEARCHUP ()** or **SEARCHDOWN ()** can provide significant execution-time benefits. These benefits are especially noticeable for larger data series.

The following table presents the example results.

This table compares map execution times of the **SEARCHUP()** and **LOOKUP()** functions, based on the size of the input data.

Map Functions	16 KB	128 KB	1 MB	2 MB
SEARCHUP ()	0.23	0.46	3.04	5.83
LOOKUP ()	0.28	4.84	288.40	1220.27

This table captures execution times in seconds for two maps that both attempt to match each data element from one input file with another. The two maps differ in the function that each uses to match data. The first map uses **SEARCHUP ()** to locate matches quickly. The second map uses **LOOKUP ()**.

For small series of data, the difference in execution time is not as noticeable. However, as the input series size grows, the execution time for the **SEARCHUP** map grows much more slowly than the execution time for the **LOOKUP** map.

A search is executed for every data element in the input file. As a result, the overall execution time grows with the data size *and* the search response time. In the case of the **LOOKUP** map, this yields a map execution time that grows exponentially with input size. A doubling of the input size requires an execution time four times as long for this map.

Routing

Routing is the splitting of output to separate files based on the data values. This is a common operation in many IBM® Transformation Extender mapping procedures. This documentation discusses differences between the **PUT ()** and **RUN ()** functions when used to route data. The behavior of each is broader than simply routing, and neither function is an exact replacement for the other.

The various functions available to the map developer to do data routing are:

- **PUT()**
- **RUN()**

For both the **PUT ()** and **RUN ()** functions, the best performance for routing style maps is generally obtained by mapping the largest available group object.

The following table contains the two functions available to the map developer to do data routing.

Function	Use	Advantage	Disadvantage
PUT ()	Routes an output item to an adapter.	Overhead is less than using RUN () .	I/O and CPU overhead from repeated opening and closing of routing targets can be great for large maps.
RUN ()	Allows a map that is currently running to execute another map.	Is versatile. Using a second map to process output allows for a great deal of flexibility in output generation.	Overhead is much greater than using PUT () . I/O and CPU overhead from repeated opening and closing of routing targets can be great for large maps.

- [Data routing usage](#)
- [Comparisons between the routing functions](#)

Data routing usage

As you develop your IBM® Transformation Extender maps to route input data and create output, you consider the format of the input and the requirements for the output. Based upon those considerations, you can make your decision on which routing function to use to derive optimum performance results. The following table lists the routing usage with the recommended function that should give you the best performance results.

Usage

Routing Function Recommendation

Avoid using additional maps to route data. Instead, try to use `PUT()` with the appropriate adapter whenever possible.

PUT

Additional processing with a secondary map is required.

RUN

As stated previously, the functionality of `PUT()` and the functionality of `RUN()` differ significantly. Because `RUN()` enables the map to execute a secondary map, there might be many routing situations where `PUT()` might not be a feasible choice. In cases where `PUT()` can be used, usually better performance results.

Comparisons between the routing functions

This documentation contains quantitative comparisons of the data routing functions. It presents results of comparing the `RUN()` against the `PUT()` functions.

In this example, two maps that attempt to route input to any one of twenty possible output targets are executed. The two maps differ in their use of `RUN()` or `PUT()` to route data items to an output target.

For each map, two scenarios were tested, each processing input of different sizes. Varying the input was intended to indicate relative scalability between the two functions in a routing context.

The following table presents the example results.

Map Functions	Input Size (Bytes)	
	128 KB	4 MB
<code>PUT()</code>	0.74	13.43
<code>RUN()</code>	2.14	62.22

This table captures execution times in seconds for the two example maps. Even for input that is smaller in size, `PUT()` executes faster. For input that is larger in size, the overhead of the additional maps in `RUN()` processing is far more noticeable. Execution of the `PUT` map completes in nearly one-fifth the time of the `RUN` map.

Comparing type sizes

Regardless of whether `PUT()` or `RUN()` is used to route data, there is an execution time overhead in each routing attempt. Minimizing the number of routings performed will improve execution time. Careful attention to map and type tree design can reduce the number of routings done.

This example demonstrates the benefit of minimizing routing attempts. Four maps were run with inputs of two different sizes. The maps differ in two primary ways. They use two different functions, `PUT()` and `RUN()`, and different type tree designs.

The two different type tree designs are used for the same input sources. In the first type tree design, **N1**, the input card type tree consists of a simple series of items (`0:s`). In the second type tree design, **N100**, the input card type tree consists of a series of groups of items. The groups themselves are series, (`0:100`). Therefore, the second type tree has two levels of series where the first, **N1**, has one.

The following table presents the example results.

This table compares the execution times of two type-tree designs, based on input data size and whether the map uses the `PUT` function or the `RUN` function.

Type Tree Design	PUT()		RUN()	
	128 KB	4 MB	128 KB	4 MB
N1	0.74	13.43	2.14	62.22
N100	0.36	0.75	0.41	1.83

This table captures execution times in seconds for the four example maps. The savings in execution time are significant, especially at larger input sizes. Whether the map uses `RUN()` or `PUT()`, reducing the number of routes can improve execution time.

Simply grouping input together to route larger amounts of data is not always possible in every map design effort. The degree to which map or type tree design can address route reduction depends on the needs of the business process. This example demonstrates the potential for improving execution time by reducing the number of routing attempts.

Map Profiler (dtxprof)

The map profiler is a user-configurable utility that captures and reports map execution statistics.

The resulting output highlights those areas in your maps, such as component and mapping rules, and the functions and types within those rules, with very long processing times, which might be adversely affecting performance. It might show that specific rules and functions are being executed for an unusual number of times. It also might show that type objects are being accessed too many times or that the nesting level of an object is too deep. With this information, you can then make the appropriate changes to your maps to achieve your performance objectives.

For example, you might have output from the map profiler that indicates that the processing time for a `LOOKUP` function in your map is significantly more than the processing time for other functions. If the data is sorted, you might see the processing time decrease by using the `SEARCHUP` function in your rule instead.

See [Data Search Usage](#).

- [Using the Map Profiler](#)

Using the Map Profiler

The profiler can be configured and started from the Performance interface using the Map Designer, or from the command line using the `dtxprof` utility command.

See the *Map Designer* documentation for information on how to use the map profiler in the Map Designer GUI.

See the *Utility Commands* documentation for information about how to use the map profiler from the command line, outside the Map Designer GUI.

The map profiler is an important tool to assist you in improving the performance of your maps and achieving your performance goals.

Page Setting Assistant for Maps (dtxpage)

The **Page Setting Assistant for Maps** application is used to calculate suggested settings for memory page size and count for work files used in maps. It is started by using the `dtxpage` utility command. The `.mmc` file is a required parameter and is the file name of the map for which the calculation is being performed.

The application will run iterations of the specified map and therefore you should expect that the overall run time will be longer than a typical run.

See the *Utility Commands* documentation for information about how to use the `dtxpage` utility command from the command line.

The **Page Setting Assistant for Maps** application invoked by the `dtxpage` utility command is an important tool to assist you in improving the performance of your maps and achieving your performance goals.

Merging and Pruning Schemas and Type Trees

You can merge and prune schemas and type trees. Always practice save file management. Copy the source type tree to another name before you make any modifications.

To prune a schema or type tree to contain only a sub-set of standard transactions, copy the source schema or type tree, for example, `hipaa_x12` or `ansi8020`, to a new name. Or copy the source schema to a new design server project.

Design Studio:

1. Copy the file using windows file system or you can open the type tree using Design Studio.
2. Save the file using a new name. For example, copy `hipaa_x12` to new type tree `hipaa_x12_5010x22a1.mtt`

Design Server:

1. Create a new project.
 2. Open the new project.
 3. Using the bar at the top where the project name is displayed and select copy into.
 4. Select the project that contains the schema you want to prune.
 5. Expand the schemas and select the schema you want to prune for example, `hipaa_x12` or `ansi8020` schema.
- [Prune transactions sets from a schema or type tree](#)
 - [Merge pruned schema or type tree to destination schema or type tree](#)
- At this step the unused messages or transactions have been removed. Merging the pruned schema or type tree will remove references to unused segments and elements within the destination schema or type tree.

Prune transactions sets from a schema or type tree

1. In the source schema or type tree, locate the ANSI Funct'lGroup Partner Inbound type.
2. Remove the unnecessary transaction sets under ANSI Funct'lGroup Partner Inbound type.
3. Remove the unnecessary transaction sets under ANSI Funct'lGroup Partner Outbound type.
4. Save changes and continue below.

Create a destination schema or type tree

Design Studio Only:

1. Create a new schema or type tree.
2. Rename root type to **EDI**.

Merge pruned schema or type tree to destination schema or type tree

At this step the unused messages or transactions have been removed. Merging the pruned schema or type tree will remove references to unused segments and elements within the destination schema or type tree.

Design Studio:

1. In the pruned source type tree, locate the Transmission type.
2. Right-click the Transmission type and select Merge from the menu.
The Merge Type dialog box opens.
3. Click on the new type tree window.
The To tree field in the Merge Type dialog box is populated with destination type tree.
4. Click Merge in the Merge Type dialog box.
The new type tree is created.
5. Copy the %_Type_Tree_Information type from the source type as a subtype of EDI (the root type) in the destination type tree.
6. Update the %_Type_Tree_Information type to reflect the changes made.
7. Analyse the new type tree.
8. Save the new type tree under the appropriate name.

Design Server:

1. In the pruned source type tree, locate the Transmission type.
2. Right-click the Transmission type and select Copy To Schema from the menu.
The Copy Type dialog box opens.
3. Enter the Schema Name and select Folder from the drop-down list.
4. Click OK.
5. Open the new type tree and copy the %_Type_Tree_Information type from the source type as a subtype of EDI (the root type) in the destination type tree using Copy To Schema.
6. Update the %_Type_Tree_Information type to reflect the changes made.
7. Analyse the new type tree.
8. Save the new type tree under the appropriate name.

Mainframe performance

This documentation explains mainframe performance, what takes place during IBM® Transformation Extender product execution, as well as performance tips and recommendations for the Batch and CICS® execution environments. This information will provide some additional background to assist in defining performance objectives.

- [Overview](#)
- [Understanding mainframe performance](#)
- [Batch execution environment](#)
- [CICS execution environment](#)
- [Troubleshooting](#)

Overview

General mainframe performance is discussed in the following topic:

- ["Understanding Mainframe Performance"](#)

Performance tips and recommendations for the two execution environments are discussed in the following topics:

- ["Batch Execution Environment"](#)
- ["CICS Execution Environment"](#)

The ["Command Server Troubleshooting"](#) information provides recommendations and references to identify potential performance areas when executing IBM® Transformation Extender under either Batch or CICS® environments. Troubleshooting information is available for ["Batch Troubleshooting"](#) and ["CICS Troubleshooting"](#).

Understanding mainframe performance

An important step in trying to achieve performance improvements running the IBM® Transformation Extender under Batch and CICS® on the z/OS® operating environment is to understand the IBM Transformation Extender execution characteristics specific to these environments.

["General Performance"](#) includes performance tips and information on what occurs during product execution when running the IBM Transformation Extender on all available platforms.

There are additional performance and execution characteristics specific to running the IBM Transformation Extender under Batch and CICS on z/OS environments that can greatly affect their overall performance, including:

- Hardware Expectations and MIPS/MSU Ratings
- File Formats
- Pre-allocating Work Files (Temporary Data Sets)
- Paging in Memory
- External Factors
- [Hardware expectations and MIPS/MSU ratings](#)

- [File formats](#)
 - [Pre-allocating work files \(temporary data sets\)](#)
 - [Paging in memory](#)
 - [External factors](#)
-

Hardware expectations and MIPS/MSU ratings

Hardware performance ratings such as MIPS or MSUs indicate machine-wide throughput, and not the speed of a single processor. As mentioned in [Throughput and CPU Expectations](#), a measurement of CPU throughput over multiple CPUs might not accurately estimate the execution time of a single transformation. This results from the single-threaded nature of map executions.

- [Performance case](#)
 - [Performance case after normalizing](#)
-

Performance case

MIPS and MSU ratings account for all active processors on a particular hardware configuration, yet the Command Server or a single transformation can only use a single processor at any given time. Therefore, comparing MIPS or MSU ratings between machines to estimate transformation speed will not produce reliable estimates if the number of processors differs. To derive MSU ratings that can provide valid estimates, the number of processors must be taken into account.

Here is a hypothetical case using fictitious configurations to illustrate how estimating performance results might be affected by various factors. The considered factors include hardware configurations, MSU ratings, and taking into account the number of processors on the machine (normalizing) used to run maps or perform transformations.

Comparing performance

Analyze the following table of the MSU ratings for two different hardware configurations:

Configuration	Number of Processors	Rating (MSUs)
Config-02	two	28
Config-01	one	13

Conclusion of comparison

The MSU ratings and estimated MIPS ratings of the higher rated configuration, **Config-02**, imply that it yields twice the execution speed. This is not the case when measuring the execution time of a single map.

Performance case after normalizing

Normalizing a machine-wide MIPS rating for the number of processors on the machine results in a more representative estimate for comparative transformation performance. It accounts for the number of processors on a box.

Comparing performance

Analyze the following table of the MSU ratings that have been normalized for two different hardware configurations to account for the number of processors on each box:

Configuration	Number of Processors	Rating (MSUs) before Normalization	Rating (MSUs) after Normalization
Config-02	two	28	14
Config-01	one	13	13

Conclusion of comparison after normalization

Because the **Config-01** configuration has only a single processor, the normalized rating and non-normalized ratings are the same. This similarity in normalized MSU ratings accounts for similarities in execution times. All other things being equal between the two configurations, the two different machines should execute a single transformation, X , in roughly the same time, T .

Parallel execution

Even though the transformation execution times for the two boxes are roughly the same, the **Config-02** configuration with two processors, could execute two instances of that same transformation X in parallel. This assumes that the input and output data sets have no interdependence. In this case, the execution time for each transformation is roughly the same.

If both transformations are initiated at the same time, however, they should complete in that same time interval, T . Because both transformations executed in parallel, the amount of transformation work for both is twice as much as it would be for a single execution.

Twice the amount of transformation work in the same amount of time yields double the throughput, which represents a significant performance improvement.

File formats

The z/OS® operating system traditionally processes data in a record-oriented fashion. On other platforms (UNIX and Windows) it is byte (stream) oriented.

The z/OS operating system offers many file format choices, including record sizes, block sizes and record types for record-oriented processing.

Some of the most common record types are: Fixed or Fixed Block, and Variable or Variable Blocked record formats.

The IBM® Transformation Extender run under Batch and CICS® environments provides an additional Record Separators execution command (/V) to enable special handling of data sets with *variable length* record formats, if required.

The special handling of record separators for variable length records occurs during initialization, finalization, or both depending upon your source, input data, and targets, output data.

- [Record separators \(/V\) execution command](#)

Record separators (/V) execution command

For more efficient access during the transformation and validation processes, input files with a variable-length format are copied to temporary work files with the /V execution command-specified separators inserted directly into the file. This operation is reversed, and record separators removed, on targets where the output record format type is variable length.

Though from the Command Server processing perspective this is generally more efficient, there is a slight overhead cost of the copy-insertion-deletion operation to take into account for variable-length records

If your requirements can be met using a fixed format, then the use of Record Separator execution command (/V) will not be required along with its slight increased overhead.

If a data set of variable-length records already contains the separators defined to the map embedded in the data, this option should not be used.

See [z/OS Batch](#) and [z/OS CICS](#) for more information about the /V Record Separators execution command.

Pre-allocating work files (temporary data sets)

There are guidelines you can follow for pre-allocating work files before you run the IBM® Transformation Extender under Batch and CICS® on z/OS® environments.

- [Batch](#)
- [CICS](#)

Batch

The Batch Command Server, by default, and without explicit pre-allocation of work files, will attempt to allocate the work files dynamically during map execution.

Dynamic allocations incur a significant penalty compared to static pre-allocation of work files. Dynamic allocation inherits the overhead associated with the operating systems' dynamic allocation mechanisms. Therefore, pre-allocating work files before execution provides overall better performance.

A good guideline to use when pre-allocating work files is to allocate two temporary work files for each map card, plus one additional file. One of the two temporary work files for each card is used to process data sets containing variable-length records. The other temporary work file and the additional file are used as general-use work files.

See [Temporary Data Sets \(Work Files\)](#) for more details about allocating work files.

CICS

The IBM® Transformation Extender running under CICS® uses a predefined VSAM workspace file for its work file processing requirements.

For more details about allocating work files, see [Temporary Data Sets \(Work Files\)](#).

Paging in memory

Processing work files in memory can improve some performance characteristics of a transformation execution. Introduced in the IBM® Transformation Extender for z/OS® v6.7.2 release, two methods of processing the work file in memory are provided. They incorporate the **Work Files in Memory** and **Work Files in Memory - Hiperspace™** execution commands.

- [WorkSpace Work Files in Memory](#)
- [WorkSpace Work Files in Memory - Hiperspace](#)

WorkSpace Work Files in Memory

The **Work Files in Memory** execution command requests that the transformation engine handle all work file pages in address-space memory. An advantage in using this command is that it significantly reduces the I/O request volume involved in the transformation and therefore decreases EXCPs. This EXCP count reduction comes at a small CPU usage penalty ranging from 10% to 15% for some maps.

WorkSpace Work Files in Memory - Hiperspace

The **Work Files in Memory - Hiperspace** execution command requests that the transformation engine handle all work file pages in Hiperspace memory. An advantage in using this command is that it significantly reduces the I/O request volume involved in the transformation and decreases EXCPs like the **Work Files in Memory** execution command, but at a lower cost of CPU usage.

Carefully consider using these execution commands to ensure that they do not monopolize your available operating system physical memory for both the **Work Files in Memory** and **Work Files in Memory - Hiperspace** execution commands.

Indiscriminate use of these execution commands *might* result in excessive operating system paging. Use these execution command options on a case-by-case basis after consulting with your systems administration staff.

Valid **Work Files in Memory** execution commands for each execution environment are:

- Batch:
 - [WorkSpace \(Work Files in Memory\) \(-WM\) or \(/WM\)](#)
 - [WorkSpace \(Work Files in Memory - Hiperspace\) \(-WH\) or \(/WH\)](#)
- CICS®
 - [WorkSpace \(Work Files in Memory\) \(-WM\) or \(/WM\)](#)

External factors

External factors can greatly affect the overall performance of the IBM® Transformation Extender running under Batch or CICS® on the z/OS® operating environment and should be taken into account when assessing overall throughput and performance.

The following factors should be considered:

- Configuration of the operating system
- Work loads / Queue depths
- Work Load Manager policies and goals
- Batch-oriented scheduling issues relating to resource contentions or collisions, or both
- Transaction priorities

Some of these factors can significantly impact overall performance of the IBM Transformation Extender running under Batch or CICS on the z/OS operating environment, thereby requiring a more in-depth analysis or real-time tools to gather resource statistics to isolate and take corrective action.

Tools such as RMF (Resource Measurement Facility), in conjunction with SMF (System Management Facilities) records, help in monitoring and identifying any performance problems within traditional batch-oriented processing. Additionally, the CICS Performance Monitor can be used to identify performance problems within CICS environments.

Batch execution environment

When executing the Batch Command Server, there are several areas where you might be able to improve performance results. These areas are:

- IBM® Language Environment® (LE) Runtimes
- Input Data Sets (Source)
- Output DataSets (Target)
- Temporary Data Sets (Work Files)
- Execution Command Settings
- Striped Data Sets
- [IBM Language Environment \(LE\) runtimes](#)
- [Input Data Sets \(Source\)](#)
- [Output data sets \(Target\)](#)
- [Temporary data sets \(work files\)](#)
- [Execution command settings](#)
- [Striped data sets](#)
- [XPLINK performance build](#)

IBM Language Environment (LE) runtimes

The following recommendations are for the LE runtime library access modifications, which might improve the performance of batch jobs running under the Batch Command Server:

- **SCEERUN** and **SCEERUN2** runtime libraries into your **LNLST**
 - **SCEELPA** data set in your LPA list (LPALSTxx)
 - Heavily used modules in the Link Pack Area (LPA)
- If these IBM® recommendations have not already been implemented in your system's environment, contact your systems' programming personnel to inquire about them.

[Performance Considerations - adjusting LE runtime options](#) discusses the runtime options module shipped with the IBM Transformation Extender for z/OS® and set specifically to avoid known performance-degrading options.

See *Language Environment Customization* for additional information.

- [Performance considerations - adjusting LE runtime options](#)

Performance considerations - adjusting LE runtime options

One way to try to improve the performance of batch jobs running under the Batch Command Server is to override Language Environment® (LE) runtime options in your JCL that runs the executable DTXCMDSV program. See the topic about overriding Language Environment (LE) runtime options in the IBM® Transformation Extender *Command Server* documentation.

IBM Transformation Extender for z/OS® is shipped with a copy of the **CEEUOPT** runtime options module linked into the IBM Transformation Extender executable module. This copy of **CEEUOPT** is included to ensure that known performance-degrading options cannot be included from the existing runtime environment.

The **STORAGE** initialization options cause the most notable performance degradation to IBM Transformation Extender. **HEAPCHK** can also adversely affect the performance of IBM Transformation Extender. **HEAP** initial size and increment and **STACK** initial size and increment might provide better IBM Transformation Extender performance if tuned for a specific customer environment or set of maps.

See the IBM z/OS Language Environment documentation for details about the **CEEUOPT** options and how to remove and replace the **CEEUOPT CSECT**. The **CEEUOPT CSECT** need only be included in the **DTXCMDSV** module.

The copy of **CEEUOPT** used in IBM Transformation Extender is coded as follows:

```
CEEUOPT  CSECT
CEEUOPT  AMODE ANY
CEEUOPT  RMODE ANY
    CEEXOPT HEAP=(1M,256K,ANY,FREE,8K,4K),
            STACK=(128K,128K,ANY,KEEP),          X
            ALL31=(ON),                         X
            STORAGE=(NONE,NONE,NONE,8K),          X
            HEAPPOOLS=(ON),                     X
            TERMTHDACT=(UADUMP)
    END
```

The **CEEUOPT** is contained in the **DTXCMDSV** module in the runtime library.

To start tuning the **CEEUOPT** settings, you will need an IBM Transformation Extender map that does not perform well at your installation, and a representative sample of the data that the map processes. Start by turning on the **RPTSTG** option **RPTSTG=(ON)**, and then running Command Server with your map or maps, and data.

The **RPTSTG** option will produce a report showing the maximum amount of storage used for the **Language Environment** **STACK** and **HEAP**. Use these as initial values when you reassemble and link-edit your new **CEEUOPT**. The goal is to minimize the amount of **STACK** and **HEAP** allocations as these have a significant effect on the performance of IBM Transformation Extender.

Note: Turn off the **RPTSTG** feature before your final performance-checks, as **RPTSTG** also impacts the performance of the **Language Environment** storage allocation modules.

Input Data Sets (Source)

When running IBM® Transformation Extender maps using input (source) data sets, the following recommendation might improve the performance of batch jobs running under the Batch Command Server.

Verify that all input (source) data sets were originally created with an optimum block size. Optimum block sizes on IBM DASD devices are one half-track of data.

This verification should occur on a case-by-case (job-by-job) basis if you suspect there might be performance problems relating to running Batch Command Server jobs.

If through this verification process, you determine that the input (source) data sets do not have optimal block sizes, you can perform the following:

- Identify the creators of the data sets
- Confirm that re-blocking the input (source) data sets will not have any adverse affect on any other potential consumers of them
- Re-block the input (source) data sets

Note: There might be instances where re-blocking the input (source) data sets might not be a viable option for various reasons including but not limited to external vendor specifications or other factors driving the application requirements.

Note: See the following IBM manuals for specific details:

- *DFSMS: Using Data Sets*
- *DFSMS: Implementing System-Managed Storage*

Output data sets (Target)

When running IBM® Transformation Extender maps to create output (target) data sets, the following recommendation for output data set JCL data definition (DD) modifications might improve the performance of batch jobs running under the Batch Command Server.

- Enable the system to determine the block size on all physical sequential (DSORG=PS) output data sets (non-temporary) created using the Batch Command Server.
 - If your system is not running SMS, modify the REVERSEO (output data set) data definition (DD) that is in the DTXBMJCL JCL example included with the Batch Command Server installation as follows:

```
/* Define the output data set
REVERSEO DD DSN=&MAPOUT,
           DCB=(DSORG=PS,RECFM=VB,LRECL=80,BLKSIZE=0),
           UNIT=&UNIT,
           SPACE=(TRK,(1,1),RLSE),
           DISP=(NEW,CATLG,DELETE)
```

- The DSORG=PS, RECFM=VB and BLKSIZE=0 DCB specifications on the REVERSEO DD will ensure that the system determines the block size for optimum performance.

See *DFSMS: Using Data Sets* and *DFSMS: Implementing System-Managed Storage* for additional information.

Temporary data sets (work files)

The Batch Command Server uses temporary data sets (work files), as described in *z/OS Batch* of the *Command Server* documentation. Temporary data sets are used throughout the JCL examples included in the Batch Command Server installation.

Temporary data set names must begin with &&, which indicates a temporary data set. They can use VIO or NON-VIO and can be statically or non-statically (dynamically) allocated.

When running IBM® Transformation Extender maps, the following recommendations for using temporary data sets might improve the performance of batch jobs running under the Batch Command Server:

- All required temporary data sets *should* be statically allocated using VIO, unless you are reusing work files.
- Static temporary data set allocations *must* be defined using RECFM=FBS (Fixed Block Standard).
- Static temporary data set allocations *should* specify LRECL=32760.
- Static temporary data set allocations *should not* specify a block size.

Using the recommendations stated above will provide the most optimal I/O performance for the Batch Command Servers' work files.

For an example JCL, see the **SYSTMP01** (temporary data set) data definition (DD) fragment from the DTXBMJCL JCL that is included in the Batch Command Server installation.

See the *Command Server* documentation for more information about statically allocated temporary data work files.

See *DFSMS: Using Data Sets* and the *DFSMS: Implementing System-Managed Storage* documentation for additional information.

Execution command settings

Execution command settings refer to those commands that are used in the Batch Command Server. They are documented in *z/OS Batch*.

Some performance improvements might be achieved with the execution commands used in the Batch Command Server.

- [WorkSpace Pagesize \(Paging\) \(-Psize:count\) or \(/Psize:count\)](#)
- [WorkSpace \(Work Files in Memory\) \(-WM\) or \(/WM\)](#)
- [WorkSpace \(Work Files in Memory - Hiperspace\) \(-WH\) or \(/WH\)](#)

WorkSpace Pagesize (Paging) (-Psize:count) or (/Psize:count)

Sets the page size and count. See the *Execution Commands* documentation for more details.

The execution region should always have enough space to hold all requested pages. The minimum region size (kilobytes) can be calculated by multiplying page size by page count (size * count). An even larger region size is preferable.

WorkSpace (Work Files in Memory) (-WM) or (/WM)

Sets the **WorkSpace** work files to reside in memory. See the *Execution Commands* documentation for more details.

WorkSpace (Work Files in Memory - Hiperspace) (-WH) or (/WH)

Sets the **WorkSpace** work files to reside in standard Hiperspaces. See *Execution Commands* documentation for more information.

Carefully consider the use of these execution commands to ensure that they do not monopolize your available operating system physical memory for both the **Work Files in Memory** and **Work Files in Memory - Hiperspace** execution commands.

Indiscriminate usage of these execution commands *might* result in excessive operating system paging. Use these execution command options on a case-by-case basis after consulting with your systems administration staff. An **IEFUSI** installation exit can be installed in your system's environment. This exit can protect memory usage when using the **Work Files in Memory** execution command. However, it will *not* protect Hiperspace memory usage when using the **Work Files in Memory - Hiperspace** execution command.

Striped data sets

Striping allows you to divide data into blocks and place them in various partitions on several hard disks. Striping data sets makes it possible to achieve performance improvements.

When running IBM® Transformation Extender maps, striping data sets might improve the performance of batch jobs running under the Batch Command Server.

- Examine your environment and determine if you can stripe sequential and VSAM data sets.
 - Data striping sequential and VSAM data sets can reduce the processing time required for batch jobs resulting in lowered EXCPS.
 - Do not stripe temporary data sets (SYSTMP01-99) processed in IBM Transformation Extender maps.

For additional information, see *DFSMS: Using Data Sets* and *DFSMS: Implementing System-Managed Storage*.

XPLINK performance build

IBM® Transformation Extender provides two Batch Command Server load libraries: non-XPLINK (default) and XPLINK (optional).

IBM XPLINK (Extra Performance Linkage) uses call linkage conventions that are intended to optimize performance of new and existing Batch Command Server jobs.

See the IBM z/OS documentation for details about XPLINK.

CICS execution environment

When executing the IBM® Transformation Extender under CICS® on the z/OS® operating environment, there are several areas where you might be able to improve performance results. These areas are:

- IBM Language Environment® (LE) Runtimes
- Temporary Data Sets (Work Files)
- Enqueuing on Data Files (-Q)
- Execution Command Settings
- Striped Data Sets
- CICS Shared Data Tables
 - See "[CICS Troubleshooting](#)", *CICS System Definition Guide* and *Language Environment Customization* for additional information.
- [IBM Language Environment \(LE\) runtimes](#)
- [Temporary data sets \(work files\)](#)
- [Enqueuing on data files \(-Q\)](#)
- [Execution command settings](#)
- [Striped data sets](#)
- [CICS shared data tables](#)

IBM Language Environment (LE) runtimes

IBM CICS® requires LE runtime libraries to be installed and available to operate CICS. Language Environment® can employ automatic storage tuning for CICS. Automatic storage tuning can improve the performance of LE applications running under CICS.

Automatic storage tuning is only available at certain levels of CICS Transaction Server.

See *CICS System Definition Guide* for additional information.

Temporary data sets (work files)

The IBM® Transformation Extender under CICS® uses temporary data sets (work files) differently in CICS than in Batch.

When running IBM Transformation Extender maps, the following recommendation for using temporary data sets, might improve the performance of CICS applications running the IBM Transformation Extender under CICS:

- The CICS Execution Option uses a **VSAM RRDS** data set defined at product installation and configuration time for its workspace (temporary files) data set.
- Verify that the **CSDSTRNO** parameter correctly reflects the total number of **DATAFILS** configured *plus* the **MAP KSDS** in your system's environment. Incorrect settings can lead to string waits causing performance degradation.

See *z/OS CICS* for detailed information about configuration requirements and options.

Enqueuing on data files (-Q)

The CICS® Execution Option can use enqueueing on data files (-Q) as described in *z/OS CICS* of the *Command Server* documentation.

By default, I/O operations against inputs and outputs are *not* serialized. Therefore, it is recommended that you:

- Examine any potential sharing of inputs or outputs, or both, for maps that run concurrently, which could possibly necessitate usage of this option.

Execution command settings

Execution command settings refer to those commands that are used in the IBM® Transformation Extender under CICS. They are documented in *z/OS CICS*.

Some performance improvements might be achieved with the execution commands used in the IBM Transformation Extender under CICS®.

- [WorkSpace Pagesize \(Paging\) \(-Psize:count\) or \(/Psize:count\)](#)
- [WorkSpace \(Work Files in Memory\) \(-WM\) or \(/WM\)](#)

WorkSpace Pagesize (Paging) (-Psize:count) or (/Psize:count)

Sets the page size and count. See the *Execution Commands* documentation for more details.

WorkSpace (Work Files in Memory) (-WM) or (/WM)

Sets the **WorkSpace** work files to reside in memory. See the *Execution Commands* documentation for more details.

Carefully consider the use of these execution commands to ensure that they do not monopolize your available operating system physical memory for the **Work Files in Memory** execution command. Indiscriminate use of these execution commands might result in excessive operating system paging. Use these execution command options on a case-by-case basis after consulting with your systems administration staff.

Striped data sets

Striping allows you to divide data into blocks and place them in various partitions on several hard disks. Striping data sets makes it is possible to achieve performance improvements.

When running IBM® Transformation Extender maps, striping data sets might improve the performance of batch jobs running under the Batch Command Server.

- Examine your environment and determine if you can stripe sequential and VSAM data sets.
 - Data striping sequential and VSAM data sets can reduce the processing time required for batch jobs resulting in lowered EXCPS.
 - Do not stripe temporary data sets (SYSTMP01-99) processed in IBM Transformation Extender maps.

For additional information, see *DFSMS: Using Data Sets* and *DFSMS: Implementing System-Managed Storage*.

CICS shared data tables

CICS® applications or transactions that run the CICS Command Server can define KSDS outputs or KSDS inputs (source) data sets within CICS Shared Data Tables. Shared data tables are intended to optimize performance. See the IBM® CICS documentation for details.

Troubleshooting

This documentation lists some of the types of questions on performance optimization and helps identify some areas in the system's environment that can be modified to help achieve performance objectives. It is divided into the following environments:

- Batch Troubleshooting
- CICS® Troubleshooting
- [Batch troubleshooting](#)
- [CICS troubleshooting](#)

Batch troubleshooting

The Batch Command Server can be subject to a wide variety of overall performance throughput. The performance factors considered here extend through:

- Usage and interpretation of the designed map or maps being executed by the Batch Command Server.
 - Internal functions used within the maps themselves.
- External factors that traditionally influence batch performance as a whole such as:
 - Input and output data sets and their corresponding block sizes.
 - Batch scheduling issues potentially impacting performance where it relates to potential resource contention and collisions.
 - Your system's environment's Workload Manager (WLM) policies and goals.

Some key areas targeted for improvement are:

- Excessive EXPS (I/Os)
 - Excessive CPU Time Used
 - [Excessive EXCPS \(I/Os\)](#)
 - [Excessive CPU time used](#)
 - [External factors](#)
-

Excessive EXCPS (I/Os)

If your job executing a map or systems of maps seems to generate an excessive amount of EXCPS (Execute Channel Programs) (I/Os), there are a number of items you can immediately examine and correct to improve by reducing the total EXCPS.

The following table lists some recommendations to make better use of resources required for a particular map executed, along with the references to additional documentation:

Recommended Action

Documentation Reference For Batch Execution Environment

Identify and verify that the output data sets that your map (or maps) generates are being created using a system-determined block size.

[Output Data Sets \(Target\)](#)

Identify and verify that your input data sets are using optimum block sizes.

[Input Data Sets \(Source\)](#)

Statically allocate, use VIO for temporary data sets, set LRECL=32760 and set RECFM=FBS on their definitions.

[Temporary Data Sets \(Work Files\)](#)

Calculate new WorkSpace PageSize size:count settings based upon your maps work. Increasing the amount of available paging memory can reduce EXCPs.

[Execution Command Settings](#)

Use striped data sets if applicable and available.

[Striped Data Sets](#)

Examine the usage of the -WM (Work Files in Memory) option.

[Execution Command Settings](#)

Examine the usage of the -WH (Work Files in Memory - Hiperspace) option.

[Execution Command Settings](#)

Proceed with any possible map tuning, if applicable.

[Map Performance Tuning Tips](#)

Install and run XPLINK load library version.

[XPLINK Performance Build](#)

Excessive CPU time used

If your job executing a map or systems of maps seems to be using an excessive amount of CPU (Central Processing Unit) time, there are a number of items you can immediately examine and correct to improve and reduce the total CPU time.

The following table lists some recommended actions you can take to make better use of valuable central processing resources required for a particular map execution, along with the reference to additional documentation:

Recommended Action

Documentation Reference For Batch Execution Environment

Calculate new WorkSpace PageSize size:count settings based upon your map(s) work.

[Execution Command Settings](#)

Examine the usage of the -WH (Work Files in Memory - Hiperspace) option.

[Execution Command Settings](#)

Proceed with any possible map tuning, if applicable.

[Map Performance Tuning Tips](#)

Reduce EXCP count. Reducing the EXCP count can lower CPU time spent managing I/O.

[Excessive EXCPS \(I/Os\)](#)

External factors

In addition to the recommended actions in ["Batch Troubleshooting"](#), there are external factors that can influence the overall performance of the Batch Command Server.

The following table lists some of these factors along with suggested references.

External Factors

Reference

Batch scheduling collisions and resource contention

- SMF data and the associated RMF reports.
- Consult your systems' programming, operations or support personnel.

Workload Manager Policies and Goals

- Consult your systems' programming or support personnel.

CICS troubleshooting

The IBM® Transformation Extender running under CICS® can be subject to a wide variety of overall performance throughput. The performance factors considered here extend through:

- Usage and interpretation of the designed maps being executed by the CICS Execution Option
 - Internal functions used within the maps themselves
- CICS application design and purpose as it relates to map usage and volume of data upon which is being operated
- External factors that traditionally influence CICS performance as a whole such as CICS:
 - Transaction mix
 - Journaling
 - Dispatching priority

Some key areas targeted for improvement are:

- Excessive CICS I/Os, Transaction CPU Time
- External Factors
- [Excessive CICS I/Os, transaction CPU time](#)
- [External factors](#)

Excessive CICS I/Os, transaction CPU time

If your CICS® application or transaction, which is executing a map or systems of maps, seems to generate an excessive amount of EXCPS (I/Os) and your transaction itself is using an excessive amount of CPU time, there are a number of items you can immediately examine and correct if needed, to improve the total required EXCPS, and CPU time.

The following table lists recommendations to make better use of CPU resources required for a particular map execution, along with documentation that contains additional information:

Recommended Action

Documentation References

Identify and verify LSR (Local Shared Resources) pools, strings and usage of CICS shared data tables, if applicable.

See the IBM® CICS documentation.

Use striped data sets if applicable and available.

[Striped Data Sets](#)

Calculate new `WorkSpace PageSize size:count` settings based upon your map(s) work.

[Execution Command Settings](#)

Examine the usage of the `-WM` (Work Files in Memory) option.

[Execution Command Settings](#)

Proceed with any possible map tuning, if applicable.

[Map Performance Tuning Tips](#)

External factors

In addition to the recommended actions in ["CICS® Troubleshooting"](#), there are external factors that can influence the overall performance of the IBM® Transformation Extender running under CICS on the z/OS® operating environment.

The following table lists some of these factors along with suggested references.

External Factors

Reference

Workload Manager Policies and Goals

Consult your systems programming or support personnel.

Other CICS performance information

See the IBM CICS documentation.

Glossary

Term	Definition	Term	Definition
Batch	Running in the background with no user assistance	LRECL	Logical Record Length
CICS®	Customer Information Control System; online transaction processing program; running in the foreground communicating with users	LSR	Local Shared Resources

Term	Definition	Term	Definition
CPU	Central Processing Unit; contains the logic functionality and processes the instructions that drive a computer.	Mainframe	Large processors or servers
DCB	Data Control Block parameter of DD statement in JCL	MIPS	Million Instructions Per Second
DD	Data Definition statement in JCL	MSU	Measured Service Units
EXCP	Execute Channel Program	MVS™	Multiple Virtual Storage; mainframe operating system
Hiperspace™	A virtual storage area (up to 2gigabytes in size) outside the user's virtual address space residing in real or expanded storage and backed up by auxiliary storage	OS/390®	Mainframe operating system (was MVS) (old name - now referred to as z/OS®)
I/O	Input and Output	Systems Z	Line of IBM® computer systems running operating systems such as z/OS
LE	IBM Language Environment®	VIO	Virtual Input/Output
LNKLST	Link List	XPLINK	eXtra Performance Linkage; An IBM performance optimization technique using new linkage conventions
LPA	Link Pack Area	z/OS	Mainframe operating system
RECFM	Record Format used in DCB parameter of DD statement in JCL		

Utility Commands overview

You can use the utility commands to do various functions.

The functions you can perform using utility commands include:

- Analyzing schema
- Importing and exporting schema
- Converting schema
- Converting schema properties from bytes to characters
- Compiling maps
- Importing to and exporting map source files
- Generating a map report HTML file containing information about a map source file
- Deploying systems
- Importing to and exporting systems
- Calculating suggested memory page size and count for maps
- Analyzing map execution behavior
- Converting XML type definitions
- Creating one map that can transform any input data into XML output, and a second map that can transform the XML that the first map produced into output data in a format described by the schema that you specified
Use the second map only when the schema that you specified was imported from a **COBOL** copybook.
- Updating a resource or virtual server in MRN files

Some of the commands offer basic functions that are performed by the Design Studio, but allow them to be executed from the command line or within a command script without running the Design Studio applications. Other commands offer other related functions and are executed in the same way.

There are utility commands for the schema designer, map designer, and Integration Flow Designer (IFD) applications as well as for other related tasks.

Schema designer:

[tanalyze utility command](#)

Analyzes schemas.

[timport utility command](#)

Imports to schemas

[texport utility command](#)

Exports schemas.

[tbccconv utility command](#)

Converts schema properties from bytes to characters.

Map Designer:

[mcompile utility command](#)

Compiles maps

[mimport utility command](#)

Imports to maps

[mexport utility command](#)

Exports maps

[mreport utility command](#)

Generates a map report HTML file from a map source file

Integration Flow Designer:

[sdeploy utility command](#)

Deploys systems

[msdimport utility command](#)

Imports to system files

[msdexport Utility Command](#)

Exports system files

Map Tuning:

[dtxpage Utility Command](#)

Calculates suggested settings for memory page size and count for maps

[dtxprof Utility Command](#)

Profiles and analyzes map execution behavior

Utility commands for XML:

[dtxany2xml Utility Command](#)

Automatically produces a map that can transform any input data into XML output

[Utility Commands for Resource Registry](#)

ResourceRegistryHelper.bat

ResourceRegistryHelper.sh

Updates a resource or virtual server in resource name (.mrn) files.

Utility commands for troubleshooting

After executing a utility command, a value is returned into the *ERRORLEVEL* environment variable indicating the success or failure of the execution. The *ERRORLEVEL* variable is a Windows environment variable that contains the return code of the last DOS command you executed.

To retrieve the value of the *ERRORLEVEL* variable, run the following DOS command.

```
ECHO % ERRORLEVEL %
```

Utility commands return the following valid values in the *ERRORLEVEL* variable:

Return Code	Description
0	This value indicates that the utility command was successful.
1	This value indicates that the utility command was <i>not</i> successful.

The *ERRORLEVEL* variable can also be used in batch files to retrieve the value of the return code.

Utility Commands for schema designer

Schema Designer utility commands are used to analyze schemas, import to and export schemas, convert schemas and convert schema properties from bytes to characters.

The utility commands for this application are:

- ["tanalyze Utility Command"](#) - analyzes schemas
- ["import Utility Command"](#) - imports metadata definitions to schemas
- ["texport Utility Command"](#) - exports schemas
- ["dtxmlconv Utility Command"](#) - converts schemas that were generated in earlier versions of IBM Transformation Extender.
- ["tbconv Utility Command"](#) - converts schema properties from bytes to characters
- [tanalyze Utility Command](#)
The tanalyze utility command is used to analyze one or more schemas from the command line, outside the schema designer GUI.
- [timport Utility Command](#)
The timport utility command is used to import metadata definitions to a schema from the command line, outside the Type Designer GUI.
- [texport Utility Command](#)
The texport utility command is used to export a schema from the command line, outside the schema designer GUI.
- [dtxmlconv Utility Command](#)
The dtxmlconv utility command is used to convert schemas that were generated in earlier versions (before 8.0) of IBM Transformation Extender from the command line, outside the schema designer GUI. It is called the XML Schema Compatibility Utility in the GUI.
- [tbconv Utility Command](#)
The tbconv utility command is used to automatically convert certain objects in a schema sized in bytes to characters from the command line.

tanalyze utility command

The tanalyze utility command is used to analyze one or more schemas from the command line, outside the schema designer GUI.

The tanalyze utility command returns 0 if the analysis is successful, and 1 if it is not successful. It is a batch operation and can also be used for automation.

See ["Troubleshooting"](#) for details about capturing and evaluating the return code resulting from the execution of this utility command.

- [Syntax summary for tanalyze](#)

- [Utility command options for tanalyze](#)
 - [Command line help for tanalyze](#)
 - [Using the tanalyze command](#)
-

Syntax summary for tanalyze

The `tanalyze` utility command is used to analyze schemas.

The `.mtt` file is a required field and is the name of the schema file that needs to be analyzed. If the full path of the schema file name is not specified, the `tanalyze` utility command will search for the schema file specified in the current directory.

None of the options in the `tanalyze` utility command are case sensitive. The following is the syntax of the `tanalyze` utility command:

```
tanalyze <.mtt file> { -L | -S | -L -S }
  [-R [<.dbe file name/location>]] [-SAVE]
  [(-LOG [
    | <log file name/location>
    [-FAIL] [-VERBOSE] [-APPEND])
  | -NOLOG]
```

Utility command options for tanalyze

The following command options are available with the `tanalyze` utility command:

• -L tanalyze Option	• -SAVE tanalyze Option
• -S tanalyze Option	• -VERBOSE tanalyze Option
• -R tanalyze Option	• -APPEND tanalyze Option
• -LOG tanalyze Option	• -NOLOG tanalyze Option
• -FAIL tanalyze Option	

If you type `tanalyze` with no options, the command line help that describes the command will display on your screen.

- [-L tanalyze option](#)
- [-S tanalyze option](#)
- [-R tanalyze option](#)
- [-LOG tanalyze option](#)
- [-FAIL tanalyze option](#)
- [-SAVE tanalyze option](#)
- [-VERBOSE tanalyze option](#)
- [-APPEND tanalyze option](#)
- [-NOLOG tanalyze option](#)

-L tanalyze option

The logic (`-L`) option of the `tanalyze` utility command is used to analyze a schema for logic. If this option is specified, the schema logic will be analyzed.

-S tanalyze option

The structure (`-S`) option of the `tanalyze` utility command is used to analyze a schema by structure. If this option is specified, the schema structure will be analyzed. If both (`-L` and `-S`) options of the `tanalyze` utility commands are specified, the schema is analyzed for both logic and structure.

-R tanalyze option

The results file (`-R`) option of the `tanalyze` utility command has a parameter to specify the path and the file name for the analyze results file produced by the schema analyzer. The specified parameter could be the name of the directory where the analyze results file will be written, the full file path specification in which a wildcard can be used to represent the analyze result file, or the full path in which the complete filename can be specified.

The default naming convention for the analyze results file is: **schema_name.dbe**.

If a wildcard is specified in the full path, for example, `tanalyze myschema.mtt -R C:\MyDev\AnalyzeResults\2002-02-28_*.err`, the analyze results file will be produced in the **C:\MyDev\AnalyzeResults** directory with the **2002-02-28_myschema.err** naming convention.

If the complete file name is specified, for example, `tanalyze myschema.mtt -R C:\MyDev\AnalyzeResults\MyAnalyzeResults.txt`, the **MyAnalyzeResults.txt** file will be produced in the **C:\MyDev\AnalyzeResults** directory.

If the (-R) option is not specified, the analyze results will be written to the console, and if (-R) is specified without a parameter, the analyze results will be written to the same directory as the schema file using the default **schema_name.dbe** naming convention.

-LOG tanalyze option

The log (-LOG) option of the `tanalyze` utility command is used to enable logging. The log file produced shows the results of the schema analysis. This option has an optional parameter to specify the file name and location for the log file. The default file name for the `tanalyze` log will be **tanalyze_schema_name.log**. The default location for the `tanalyze` log will be the directory where the schema file is located.

The file name or the location parameter, if specified with the (-LOG) option of the `tanalyze` utility command, could be the name of the directory where the `tanalyze` log file will be written, the full file path specification in which a wildcard can be used to represent the schema file, or the full path in which the complete filename can be specified. If the . is specified with the (-LOG) option, the `tanalyze` log will be written to the default location (or the directory where the schema file is located) using the default naming convention. If a wildcard is specified in the full path, for example, `tanalyze myschema.mtt -LOG C:\MyDev\AnalyzeResults\2002-02-28_*.results`, the `tanalyze` log file will be produced in the **C:\MyDev\AnalyzeResults** directory with the **2002-02-28_myschema.results** naming convention. If the complete file name is specified, for example, `tanalyze myschema.mtt -LOG C:\MyDev\AnalyzeResults\myschema_results.txt`, the **myschema_results.txt** file will be produced in the **C:\MyDev\AnalyzeResults** directory.

If the `tanalyze` utility command is unable to create the `tanalyze` log file, the schema analyzing process will terminate, and the `Fatal error : Could not start the process, unable to create the log file` fatal error message will display on the console. If nothing is specified in the command line for -LOG (neither -LOG or -NOLOG) or if no parameter is provided for -LOG, the command line assumes the (-LOG) option is included in the command line, and writes the log messages to the console.

-FAIL tanalyze option

The fail (-FAIL) option of the `tanalyze` utility command is used to indicate that only the schemas that had analyze errors or warnings during the analysis should be included in the `tanalyze` log file. If the log failures only (-FAIL) option is selected, the `tanalyze` log will contain only the schemas that had errors or warnings. If the log failures only (-FAIL) option is not selected in the command line, all the analyze results will be written to the `tanalyze` log file.

-SAVE tanalyze option

The save (-SAVE) option of the `tanalyze` utility command is used to save the schema after analysis. If this option is specified, after the schema is analyzed, the schema will be saved. Use this command option to save any changes since the last time the schema structure analysis was done.

-VERBOSE tanalyze option

The verbose (-VERBOSE) option of the `tanalyze` utility command is used to indicate whether or not the verbose `tanalyze` log should be produced. The verbose `tanalyze` log will contain an entry for the schema being analyzed or attempted, the schema name, the analyzing start date/time, the analyzing end date/time, the analyzing result, and the analyze results file name. If the Verbose (-VERBOSE) option is not selected, a concise version of the `tanalyze` log will be produced, which includes an entry for the schema name and the result of the schema analysis.

-APPEND tanalyze option

The append (-APPEND) option of the `tanalyze` utility command is used to indicate that the current `tanalyze` log messages should be appended to the existing `tanalyze` log file if it already exists. If the `tanalyze` log -APPEND option is selected, the `tanalyze` log messages will be appended to the existing file if it exists; otherwise, a new file will be created. If the -APPEND option is not selected, the `tanalyze` log file will be created if no log file exists, or it will be overwritten if the log file does exist.

-NOLOG tanalyze option

The disable log (-NOLOG) option of the `tanalyze` utility command is used to turn off logging. If the disable logging option is selected, the log file will be not produced. The optional (-FAIL, -APPEND, and -VERBOSE) options of the (-LOG) option cannot be used with the disable log (-NOLOG) option. If these options are selected with the (-NOLOG) option, the command line is invalid, and the `Not a valid command line : (-LOG) optional commands (-APPEND), (-FAIL), (-VERBOSE) cannot be used with disable log command (-NOLOG)` error message will display on the console.

Command line help for tanalyze

There is help that describes the `tanalyze` utility command usage in the command line.

To view the command line help for the `tanalyze` utility command:

Enter `tanalyze` at the command line prompt in the *install_dir*.
The syntax and descriptions of all the options appear on the console.

Using the tanalyze command

Use the `tanalyze` utility command to analyze one or more schemas from the command line, outside the schema designer GUI.

The following example shows how you can use the command:

```
install_dir> tanalyze myschema.mtt
  -L -S
  -R C:\MyDev\AnalyzeResults\myschema.dbe
  -LOG -APPEND
```

When you run the example, the following result occurs:

- If the `myschema.mtt` schema file is analyzed successfully, the `Analysis successful.` message displays on the console.
- The `myschema.dbe` analyze results file is produced in the `C:\MyDev\AnalyzeResults` directory.
- If the `myschema.log` log file does not exist, it is produced in the same directory as the schema file. If the `myschema.log` log file already exists, the log messages are appended to the existing file.

timport utility command

The `timport` utility command is used to import metadata definitions to a schema from the command line, outside the Type Designer GUI.

The `timport` utility command returns 0 if the schema import is successful and 1 if it is not successful. It is a batch operation and can also be used for automation.

See "[Troubleshooting](#)" for details about capturing and evaluating the return code resulting from the execution of this utility command.

- [Syntax summary for timport](#)
- [Utility command options for timport](#)
- [Command line help for timport](#)
- [Using the timport command](#)

Syntax summary for timport

`timport` is the name of the utility command used to import metadata definitions to schemas.

All the options in the `timport` utility command are case sensitive. The following is the syntax of the `timport` utility command:

```
timport [-IMP <IMPORTER_OPTIONS>]
  [-NO]
  [-O <.mtt file name/location >]
  [(-LOG [. | <log file name/location>]
    [-FAIL] [-VERBOSE] [-APPEND]
    [-HELP] [-KEEPMTS <.mts filename>])
   | -NOLOG]
```

Utility command options for timport

The following command options are available with the `timport` utility command:

• -APPEND timport option	• -HINTS timport option	• -NO timport option
• -BYTEORDER timport option	• -IMP timport option	• -NOLOG timport option
• -CHARSET timport option	• -KEEPMTS timport option	• -O timport option
• -CICS timport option	• -LANG timport option	• -OPT timport option
• -FAIL timport option	• -LOG timport option	• -VALIDATION timport option
• -HELP timport option	• -ND timport option	• -VERBOSE timport option

If you type `timport` with no options, the command-line help that describes the command displays on your screen.

- [-APPEND timport option](#)
- [-BYTEORDER timport option](#)
- [-CHARSET timport option](#)
- [-CICS timport option](#)
- [-FAIL timport option](#)
- [-HELP timport option](#)
- [-HINTS timport option](#)
- [-IMP timport option](#)
- [-KEEPMTS timport option](#)
- [-LANG timport option](#)

- [-LOG timport option](#)
 - [-ND timport option](#)
 - [-NO timport option](#)
 - [-NOLOG timport option](#)
 - [-O timport option](#)
 - [-OPT timport option](#)
 - [-VALIDATION timport option](#)
 - [-VERBOSE timport option](#)
 - [IMPORTER OPTIONS](#)
-

-APPEND timport option

The append (-APPEND) option of the `timport` utility command is used to indicate that the current `timport` log messages should be appended to the existing `timport` log file if it already exists. If the `timport` log -APPEND option is selected, the `timport` log messages will be appended to the existing file, if it already exists. If it does not exist, a new file will be created. If the -APPEND option is not selected, the `timport` log file will be created if no log file exists, or it will be overwritten if the log file does exist.

-BYTEORDER timport option

The byte order (-BYTEORDER) option of the `timport` utility command is available to use with the COPYBOOK importer option to import metadata definitions to a schema with the COBOL Copybook importer. It is specified together with a byte order set argument. The byte order set is the convention a machine processor uses to position its lowest byte within a word to begin either from the leftmost or the rightmost position and describes the execution-time data. Valid values for the byte order set argument are NATIVE, BIGENDIAN and LITTLEENDIAN. If no byte order set argument is specified, the default value is NATIVE.

-CHARSET timport option

The character set (-CHARSET) option of the `timport` utility command is available to use with the COPYBOOK importer option to import metadata definitions to a schema with the COBOL Copybook importer. It is specified together with a character set argument. The character set is the standard collection of letters, numbers and symbols that describes the execution-time data. The values for the character set argument are any one of the valid character set codes. If no character set argument is specified, the default value is NATIVE. For the list of valid character set codes, see the character set codes topic in the Functions and Expressions documentation.

-CICS timport option

The CICS (-CICS) option of the `timport` utility command is available to use with the COPYBOOK importer option to import metadata definitions to a schema with the COBOL Copybook importer. It specifies to the importer that it should generate a schema for a CICS adapter.

-FAIL timport option

The fail (-FAIL) option of the `timport` utility command is used to indicate that only the schemas that had errors during the importing should be included in the `timport` log file. If the log failures only when the (-FAIL) option is selected, the `timport` log will contain only the schemas that had failed importing. If the log failures only -FAIL option is not selected in the command line, all the imported results will be written to the `timport` log file.

-HELP timport option

The help (-HELP) option of the `timport` utility command is used to display information about the specified importer. It is used with the importer options presented in the "[IMPORTER OPTIONS](#)".

The syntax is:

```
timport -IMP <importer option> -HELP
```

Example

An example of the syntax to use to display information about the COPYBOOK importer option is as follows:

```
timport -IMP COPYBOOK -HELP
```

-HINTS timport option

The XSDL HINTS (-HINTS) `timport` option is available to use with the XML Schema importer.

The default value for the `HINTS` `timport` option for any schemas that are generated, is `NONE`. The other available options that you can specify are `ALLELEMENTS`, `GLOBALELEMENTS`, and `ROOTELEMENT`.

-IMP timport option

The import (`-IMP`) option of the `timport` utility command is used to specify the importer type to use to import the metadata definitions to the schema. The available importer types are presented in the ["IMPORTER OPTIONS"](#).

-KEEPMTS timport option

The keep `mts` file import (`-KEEPMTS`) option of the `timport` utility command is specified together with the system file location and name. This system file is the intermediate `mts` file that is generated when the `timport` utility command runs. The `-KEEPMTS` option ensures that the specified system file generated by the utility command, remains after the command is run, instead of being deleted.

The keep `mts` file import (`-KEEPMTS`) option of the `timport` utility command is not available to be used with the `MTS` importer option.

This option is also only used for the deprecated COBOL Copybook Importer. If you specify the option for the non-deprecated (`-ND`) COBOL Copybook Importer, the import process ignores it. Since the process ignores the `KEEPMTS` importer option, it does not produce an error message.

-LANG timport option

The language (`-LANG`) option of the `timport` utility command is available to use with the `XMLSHEMA` and `XMLDTD` importer options to import the metadata definitions to a schema with the XML Schema or XML DTD importers. It is specified together with a language option argument. The language option argument is the language type that describes the execution-time data. Valid values for the language option argument are `JAPANESE` and `WESTERN`. If no language option argument is specified, the default value is `WESTERN`.

-LOG timport option

The log (`-LOG`) option of the `timport` utility command is used to enable logging. The log file will be produced showing the result of the imported schema. The `-LOG` option has an optional parameter to specify the location or file name, or both, for the `timport` log file. The default file name for the `timport` log will be `mts_file_name.log`. The default location for the `timport` log will be the directory where the `mts` file is located.

The file name or the location parameter, if specified with the (`-LOG`) option of the `timport` utility command, could be the name of the directory where the `timport` log file will be written, the full file path specification in which a wildcard can be used to represent the `mts` file, or the full path in which the complete filename can be specified. If `.` is specified with the `-LOG` option, the `timport` log will be written to the default location (directory where the schema file is located) using the default naming convention. If a wildcard is specified in the full path, for example, `timport mymts.mts -LOG C:\MyDev\ImportResults\2002-02-28 *.results`, the `timport` log file will be produced in the `C:\MyDev\ImportResults` directory with the `2002-02-28_mymts.results` naming convention. If the complete file name is specified, for example, `timport mymts.mts -LOG C:\MyDev\ImportResults\mymts_results.txt`, the `mymts_results.txt` will be produced in the `C:\MyDev\ImportResults` directory.

If the `timport` utility command is unable to create the `timport` log file, importing of the metadata definitions to the schema will terminate, and the Fatal error – Could not start importing the schema, unable to create the log file fatal error message will display on the console. If nothing is specified in the command line for the log, by default, the command line assumes the `-LOG` option will be included in the command line, and writes the log messages to the console.

-ND timport option

The `-ND` `timport` option triggers the run of the non-deprecated COBOL Copybook Importer.

To run the non-deprecated COBOL Copybook Importer, specify the `-ND` `timport` option with the `COPYBOOK` importer option. The importer uses the default `XML` file, `importdefs.xml`, that contains the default importer option settings. You can use the `-OPT` `timport` option with the `-ND` option to explicitly specify the default importer options `XML` file. You can also specify an alternative `XML` file that contains different default importer option settings.

The `-ND` `timport` option is not used for the deprecated COBOL Copybook Importer. To run the deprecated COBOL Copybook Importer, specify the `COPYBOOK` importer option without the `-ND` option.

-NO timport option

The no overwrite (`-NO`) option of the `timport` utility command is used to indicate that an existing imported `mts` file (`.mts`) is not to be overwritten. If the no overwrite (`-NO`) option is specified in the command line of the `timport` utility command, and the file with the name of the schema file resulting from the importing process exists, it will not be overwritten, the schema exporting will fail, and the The imported mts file cannot be overwritten, file already exists message will be written to the `timport` log file, if enabled. If the (`-NO`) option is not specified, the file with the name of the schema resulting from the importing process will be overwritten.

-NOLOG timport option

The disable log (-NOLOG) option of the `timport` utility command is used to turn off the logging function of the `timport` utility command. If the disable logging option is selected, the log file will not be produced. The optional (-LOG) options (-APPEND, and -VERBOSE) cannot be used with the disable log (-NOLOG) option. If these options are selected with the (-NOLOG) option, the command line will be invalid, and the `Not a valid command line - (LOG) optional commands (-APPEND), (-VERBOSE)` cannot be used with disable log command (-NOLOG). error message will display on the console.

-O timport option

The output (-O) option of the `timport` utility command is used to specify, through a parameter, the location or file name, or both, for the output `.mtt` schema file produced by importing the `.mts` file. The specified parameter could be the name of the directory where the `.mtt` file is written, the full file path specification in which a wildcard can be used to represent the `.mtt` file, or the full path in which the complete filename can be specified. The default naming convention for the `.mtt` file is `mts_file_name.mtt`. If a wildcard is specified in the full path, for example, `timport mymts.mts -O C:\MyDev\ImportOutputs\2002-02-28_*.mtt`, the resulting `.mtt` file is produced in the `C:\MyDev\ImportOutputs` directory with the `2002-02-28_mymts.mtt` naming convention. If the complete file name is specified, for example, `timport mymts.mts -O C:\MyDev\ImportOutputs\MyMtt.mtt`, the `MyMtt.mtt` file is produced in the `C:\MyDev\ImportOutputs` directory. If the (-O) is selected and if the file name is not specified, the command line is invalid, and the `Not a valid command line - the file name parameter is required if command option (-O) is selected` error message displays on the console.

-OPT timport option

The options file (-OPT) `timport` option is available to use with the COPYBOOK `timport` importer option that is specified with the -ND option.

It is only used for the non-deprecated COBOL Copybook Importer.

Use -OPT to specify some of the options that are available under Window,>Preferences,>Importer,>COBOL. If you specify the -OPT `timport` option, you must also specify an XML (.xml) file that contains importer option settings. You can specify the default XML file, `importdefs.xml`, that contains the default importer option settings. You can also specify an alternative XML file that contains different default importer option settings that you need the COBOL Copybook Importer process to use instead.

If you do not include the -OPT `timport` option in your command that includes the -ND option, the COBOL Copybook Importer process uses the default XML file.

To use the COBOL Copybook Importer command with different options, examine the default XML file to see how to set up the options in an alternative XML file.

This option is not used for the deprecated COBOL Copybook Importer. If you specify the -OPT option for the deprecated COBOL Copybook, the import process does not detect it. Since the process does not detect the -OPT importer option in this case, it does not produce an error message.

-VALIDATION timport option

The validation (-VALIDATION) option of the `timport` utility command is available to use with the XMLSCHEMA and XMLDTD importer options to import the metadata definitions to a schema with the XML Schema or XML DTD importers. It is specified together with a validation type option argument. The validation type option argument is the validation type that describes the execution-time data. Valid values for the validation option argument are XERCES and CLASSIC. If no validation option argument is specified, the default value is XERCES.

-VERBOSE timport option

The verbose (-VERBOSE) option of the `timport` utility command indicates if the verbose `timport` log should be produced. The verbose `timport` log will contain an entry for the `.mts` file being imported, the import start date/time, the import end date/time, and the result of the `timport` utility command. If the -VERBOSE option is not selected, a concise version of the `timport` log will be produced, which includes an entry for the `.mts` file name and the result of the import.

IMPORTER OPTIONS

The following available IMPORTER OPTIONS of the `timport` utility command can be supplied:

[MTS timport importer option](#) (Default)
[COPYBOOK timport importer option](#) | [XMLSCHEMA timport importer option](#) | [XMLDTD timport importer option](#)

The following section includes the descriptions and the command syntaxes for each of these importer options.

Each importer-specific command line must be contained within a set of brackets ([]). To see syntax examples, see the specific importer option documentation.

To see how to view the command-line help information for more syntax and usage details, see the "[-HELP timport option](#)".

If you do not specify the output (-O) option, the import process saves the schema in the port directory. But, if you are using the MTS `timport` option, and you specified a full path in the `FileName` tag in the schema script (.mts) file, the following results occur. The import process saves the schema in the specified path instead of the port directory.

If you do specify the output (-O) option, the import process saves the schema in the specified location.

MTS timport importer option

MTS is one of the options that can be specified for IMPORTER OPTIONS on the `timport` utility command. It is used to import an mts file and is specified as follows:

```
[MTS <.mts file>]  
- or -  
<.mts file>
```

Because MTS is the default, the MTS option is optional. Only the mts file must be specified. If the MTS option is specified, the mts file must be provided. If it is not provided, the command line is invalid, and the Not a valid command line - the file name parameter is required if command option (MTS) is selected error message displays on the console.

The .mts file is a required field and is the file name of the mts file that you want to import. If the full path of the mts file name is not specified, the timport utility command searches for the mts file in the current directory.

COPYBOOK timport importer option

COPYBOOK is one of the options that can be specified for IMPORTER OPTIONS on the timport utility command. It is used to designate the COBOL Copybook importer as the schema importer type and is specified as follows:

```
COPYBOOK [<.cpy file> [-BYTEORDER timport option <byte order>  
-CHARSET timport option <charset> -CICS timport option  
[-ND timport option [-OPT <importer_opts_xml_file.xml>]]]
```

Specify the full path of the copybook file in the command line.

XMLDTD timport importer option

XMLDTD is one of the options that can be specified for IMPORTER OPTIONS on the timport utility command. It is used to designate the XML DTD importer as the schema importer type and is specified as follows:

```
XMLDTD [<.dtd file> [-LANG timport option <language>  
-VALIDATION timport option <validation_type>]]
```

Specify the full path of the dtd file in the command line.

XMLSCHEMA timport importer option

XMLSCHEMA is one of the options that can be specified for IMPORTER OPTIONS on the timport utility command. It is used to designate the XML Schema importer as the schema importer type and is specified as follows:

```
XMLSCHEMA [<.xsd file> [-LANG timport option <language>  
-HINTS timport option <xsd1_hints>  
-VALIDATION timport option <validation_type>]]
```

Specify the full path of the xsd file in the command line.

Command line help for timport

There is help that describes the timport utility command usage in the command line.

To view the command line help for the timport utility command:

Enter timport at the command line prompt in the *install_dir*.
The syntax and descriptions of all the options appear on the console.

Using the timport command

Use the timport utility command to:

- import metadata definitions to a schema file from the command line, outside the schema designer GUI.
- import Copybook metadata definitions to a schema from the command line, outside the schema designer GUI.
- [Import a schema file example](#)
- [Import a copybook schema example](#)
- [Import a copybook schema with ND option example](#)

Import a schema file example

The following example shows how you can use the timport utility command-line option to import metadata definitions in the mymts.mts file to the mymts.mtt schema file.

```
install_dir> timport mymts.mts  
-O C:\MyDev\ImportResults\mymts.mtt  
-LOG -APPEND
```

When you run the example, the following result occurs:

- If the mymts.mts file is imported successfully, the `timport - completed successfully.` message displays on the console.
- A mymts.mtt schema is created in the C:\MyDev\ImportResults directory.
- If the mymts.log log file does not exist, it is produced in the same directory as the .mts file. If the mymts.log log file exists, the log messages are appended to the existing file.

Import a copybook schema example

The following example shows how you can use the `timport` utility command-line option to import the COBOL Copybook wrapped.cpy metadata file to a schema for a CICS adapter. The COBOL Copybook importer is specified to use the `NATIVE` byte order set, `NATIVE` character set, create the wrapped.log file, and to retain the generated intermediate wrapped.mts system file.

```
timport -IMP COPYBOOK
[c:\install_dir\examples\general\copybook\wrapped.cpy
-BYTEORDER NATIVE -CHARSET NATIVE -CICS]
-O c:\install_dir\examples\general\copybook\wrapped.mtt
-LOG c:\install_dir\examples\general\copybook\wrapped.log
-KEEPMTS
c:\install_dir\examples\general\copybook\wrapped.mts
```

Import a copybook schema with ND option example

The following example shows how you can use the `timport` utility command-line option to import the COBOL Copybook examplecpybook.cpy metadata file to a schema for a CICS adapter. The non-deprecated COBOL Copybook importer is specified to use the `NATIVE` byte order set, and `NATIVE` character set. It is specified to use the COBOL Copybook importer options that are set in the importeropts.xml XML file. The importer is also specified to create the test.log log file.

```
timport -IMP COPYBOOK
[c:\install_dir\examples\general\copybook\examplecpybook.cpy
-BYTEORDER NATIVE -CHARSET NATIVE -CICS
-ND -OPT c:\install_dir\examples\general\copybook\importeropts.xml]
-O c:\install_dir\examples\general\copybook\examplecpybook.mtt
-LOG c:\install_dir\examples\general\copybook\test.log -NO -VERBOSE
```

texport utility command

The `texport` utility command is used to export a schema from the command line, outside the schema designer GUI.

The `texport` utility command returns 0 if the schema export is successful and 1 if it is not successful. It is a batch operation and can also be used for automation.

See ["Troubleshooting"](#) for details about capturing and evaluating the return code resulting from the execution of this utility command.

- [Syntax summary for texport](#)
- [Utility command options for texport](#)
- [Command line help for texport](#)
- [Using the texport command](#)

Syntax summary for texport

`texport` is the name of the utility command used to export schemas. The `.mtt` file is a required field and is the file name of the schema file that needs to be exported. If the full path of the schema file name is not specified, the `texport` utility command will search for the schema file specified in the current directory.

None of the options in the `texport` utility command are case sensitive. The following is the syntax of the `texport` utility command:

```
texport <.mtt file>
[-T <type name>]
[-NO]
[-O <.mts file name/location>]
[(- -LOG [.      | <log file name/location>]
          [-FAIL] [-VERBOSE] [-APPEND])
 | -NOLOG]
```

Utility command options for texport

The following command options are available with the `texport` utility command:

• -T texport Option	• -FAIL texport Option
• -O texport Option	• -APPEND texport Option
• -NO texport Option	• -VERBOSE texport Option
• -LOG texport Option	• -NOLOG texport Option

If you type `texport` with no options, the command line help that describes the command will display on your screen.

- [**-T texport option**](#)
- [**-O texport option**](#)
- [**-NO texport option**](#)
- [**-LOG texport option**](#)
- [**-FAIL texport option**](#)
- [**-APPEND texport option**](#)
- [**-VERBOSE texport option**](#)
- [**-NOLOG texport option**](#)

-T texport option

The type name (-T) option of the `texport` utility command is used to specify through a parameter, the specific type that needs to be exported. The type name is case sensitive and it should be the full path of the type. Specify the full path in descending order starting from the root type in the schema to the type that you want to export. You must separate each type in the hierarchy by a semi-colon. An example of the format is: `roottype:ancestortype:parenttype:actualtype`. If you did not specify the (-T) option, the whole schema from the root is exported.

If you specified the type name (-T) option but did not specify type name in the command line, it is an invalid command line, and the `Not a valid command line : Type name parameter is required if type name command option (-T) is selected` error message displays on the console. If the type name that you specified in the command line is not found in the schema, the `texport` execution terminates and the `Not a valid command line : Type name specified is not valid` error message displays on the console.

-O texport option

The output (-O) option of the `texport` utility command is used to specify, through a parameter, the location or file name, or both, for the exported mts file produced by exporting the schema. The specified parameter could be the name of the directory where the exported mts file is written, the full file path specification in which a wildcard can be used to represent the exported mts file, or the full path in which the complete filename can be specified. The default naming convention for the exported mts file is `schema_name.mts`. If a wildcard is specified in the full path, for example, `texport myschema.mtt -O C:\MyDev\ExportResults\2002-02-28_*.mts`, the analyze results file is produced in the `C:\MyDev\ExportResults` directory with the `2002-02-28_myschema.mts` naming convention. If the complete file name is specified, for example, `texport myschema.mtt -O C:\MyDev\ExportResults\MyExportedMts.mts`, the `MyExportedMts.mts` mts file is produced in the `C:\MyDev\ExportResults` directory. If the export output (-O) option is selected and if the file name is not specified, the command line is invalid, and the `Not a valid command line - the file name parameter is required if export command option (-O) is selected` error message displays on the console.

-NO texport option

The no overwrite (-NO) option of the `texport` utility command is used to indicate that an existing exported mts file (`.mts`) is not to be overwritten. If the no overwrite (-NO) option is specified in the command line of the `texport` utility command, and the file with the name of the schema file resulting from the exporting process exists, it will not be overwritten, the schema exporting will fail, and the `The exported mts file cannot be overwritten, file already exists` message will be written to the `texport` log file, if enabled. If the (-NO) option is not specified, the file with the name of the schema resulting from the exporting process, will be overwritten.

-LOG texport option

The log (-LOG) option of the `texport` utility command is used to enable logging. The log file will be produced showing the result of the exported schema. (-LOG) has an optional parameter to specify the location or file name, or both, for the `texport` log file. The default file name for the `texport` log will be `schema_file_name.log`. The default location for the `texport` log will be the directory where the schema file is located.

The file name or the location parameter, if specified with the (-LOG) option of the `texport` utility command, could be the name of the directory where the `texport` log file will be written, the full file path specification in which a wildcard can be used to represent the schema file, or the full path in which the complete filename can be specified. If the . is specified with the (-LOG) option, the `texport` log will be written to the default location (directory where the schema file is located) using the default naming convention. If a wildcard is specified in the full path, for example, `texport myschema.mtt -LOG C:\MyDev\ExportResults\2002-02-28_*.results`, the `texport` log file will be produced in the `C:\MyDev\ExportResults` directory with the `2002-02-28_myschema.results` naming convention. If the complete file name is specified, for example, `texport myschema.mtt -LOG C:\MyDev\ExportResults\myschema_results.txt`, the `myschema_results.txt` will be produced in the `C:\MyDev\ExportResults` directory.

If the `texport` utility command is unable to create the `texport` log file, the exporting schema will terminate, and the `Fatal error - Could not start exporting the schema, unable to create the log file` fatal error message will display on the console. If nothing is specified in the command line for the log, by default, the command line assumes the (-LOG) option is included in the command line, and writes the log messages to the console.

-FAIL texport option

The fail (-FAIL) option of the `texport` utility command is used to indicate that only the schemas that had errors during the exporting should be included in the `texport` log file. If log failures only (-FAIL) option is selected, the `texport` log will contain only the schemas that had failed exporting. If the log failures only (-FAIL) option is not selected in the command line, all the exported results will be written to the `texport` log file.

-APPEND texport option

The append (-APPEND) option of the `texport` utility command is used to indicate that the current `texport` log messages should be appended to the existing `texport` log file if it already exists. If the `texport` log append (-APPEND) option is selected, the `texport` log messages will be appended to the existing file if it already exists; otherwise, a new file will be created. If the (-APPEND) option is not selected, the `texport` log file will be created if no log file exists, or it will be overwritten if the log file does exist.

-VERBOSE texport option

The verbose (-VERBOSE) option of the `texport` utility command is used to indicate if the verbose `texport` log should be produced. The verbose `texport` log will contain an entry for the schema being exported, the schema name, the exporting start date/time, the exporting end date/time, the exported xml file name, and the result of the `texport` utility command. If the verbose (-VERBOSE) option is not selected, a concise version of the `texport` log will be produced, which includes an entry for the schema, schema name, and exporting result.

-NOLOG texport option

The disable log (-NOLOG) option of the `texport` utility command is used to turn off the `texport` logging capability. If the disable logging option is selected, log file will not be produced. The (-LOG) optional (-APPEND, and -VERBOSE) options cannot be used with the disable log (-NOLOG) option. If these options are selected with the (-NOLOG) option, the command line is invalid, and the Not a valid command line - (LOG) optional commands (-APPEND), (-VERBOSE) cannot be used with disable log command (-NOLOG). error message will display on the console.

Command line help for texport

There is help that describes the `texport` utility command usage in the command line.

To view the command line help for the `texport` utility command:

Enter `texport` at the command line prompt in the `install_dir`.
The syntax and descriptions of all the options appear on the console.

Using the texport command

Use the `texport` command to export a schema from the command line, outside the schema designer GUI.

- [Example 1](#)
- [Example 2](#)

Example 1

Use the `texport` utility command to export a schema from the command line, outside the Type Designer GUI.

The following example shows how you can use the command:

```
install_dir> texport myschema.mtt  
-O C:\MyDev\ExportResults\myschema.mts  
-LOG -APPEND
```

When you run the example, the following result occurs:

- If the `myschema.mtt` schema file is exported successfully, the `texport - completed successfully.` message displays on the console.
- The exported `myschema.mts` xml file is produced in the `C:\MyDev\ExportResults` directory. Because no type name is specified, the whole type schema from the root is exported.
- If the `myschema.log` log file does not exist, it is produced in the same directory as the schema. If the `myschema.log` log file already exists, the log messages are appended to the existing file.

Example 2

Use the `texport` utility command to export a schema from the command line, outside the Type Designer GUI.

The following example shows how you can use the command:

```
install_dir> texport myschema.mtt  
-T MyRoot:MyAncestor:MyParent:MyType
```

```
-O C:\MyDev\ExportResults\2002-02-28_myschema.mts  
-NOLOG
```

When you run the example, the following result occurs:

- If the MyType type of the **myschema.mtt** schema file is exported successfully, the `texport - completed successfully.` message displays on the console.
- The exported **2002-02-28_myschema.mts** xml file is produced in the **C:\MyDev\ExportResults** directory. The **2002-02-28_myschema.mts** xml file contains only the exported information of the MyType type.
- If the MyType type does not exist, or the full path leading to it is not valid, the `Type name specified is not valid` message displays on the console.

No log file is produced because the (`-NOLOG`) option has been selected.

dtxmlconv Utility command

The `dtxmlconv` utility command is used to convert schemas that were generated in earlier versions (before 8.0) of IBM Transformation Extender from the command line, outside the schema designer GUI. It is called the XML Schema Compatibility Utility in the GUI.

When run in batch mode, the `dtxmlconv` utility command returns `0` if the schema conversion is successful and `1` if it is not successful. It is a batch operation, which can also be used for batch automation.

The `.mtt` file is a required field and is the file name of the schema file that needs to be converted to the current format. If the full path of the schemafile name is not specified, the `dtxmlconv` utility command searches for the schema file specified in the current directory.

See "[Troubleshooting](#)" for details about capturing and evaluating the return code resulting from the execution of this utility command.

- [Syntax summary for dtxmlconv](#)
- [Utility command options for dtxmlconv](#)
- [Command line help for dtxmlconv](#)
- [Using the dtxmlconv command](#)

Syntax summary for dtxmlconv

`dtxmlconv <.mtt_file_name/location>`

```
[ -V 6.7|6.7.1|6.7.2|7.5] [-L]  
[ -S DTD_or_SCHEMA_file_name/location ]  
[ -B backup_file_name/location [-O]]  
[ ( -LOG [.|< log_file_name/location >]  
[-FAIL] [-VERBOSE] [-APPEND])  
| -NOLOG]
```

Utility command options for dtxmlconv

The following command options are available with the `dtxmlconv` utility command:

• -V dtxmlconv Option	• -LOG dtxmlconv Option
• -L dtxmlconv Option	• -FAIL dtxmlconv Option
• -S dtxmlconv Option	• -APPEND dtxmlconv Option
• -B dtxmlconv Option	• -VERBOSE dtxmlconv Option
• -O dtxmlconv Option	• -NOLOG dtxmlconv Option

If you type `dtxmlconv` with no options, the command line help that describes the command will display on your screen.

- [-V dtxmlconv option](#)
- [-L dtxmlconv option](#)
- [-S dtxmlconv option](#)
- [-B dtxmlconv option](#)
- [-O dtxmlconv option](#)
- [-LOG dtxmlconv option](#)
- [-FAIL dtxmlconv option](#)
- [-APPEND dtxmlconv option](#)
- [-VERBOSE dtxmlconv option](#)
- [-NOLOG dtxmlconv option](#)

-V dtxmlconv option

The version (-v) option of the `dtxmlconv` utility command has parameters to specify the version of the importer used to generate the schema. The valid values are:

- 6.7
- 6.7.1
- 6.7.2
- 7.5 (default)

-L dtxmlconv option

The language (-L) option argument of the `dtxmlconv` utility command is the language type that describes the execution-time data. The valid values are:

- JAPANESE
- WESTERN

If no language option argument is specified, the default value is WESTERN.

-S dtxmlconv option

The source document file name or URL (-s) option of the `dtxmlconv` utility command will have a parameter to specify the source document used to generate the original schema. This will either be a **.dtd** (DTD) or **.xsd** (schema) file and will represent the new XML grammar that will be used to generate the schema.

If there is no value specified, the utility will automatically use the value that is specified in the schema. If a value is specified, the utility will use that specified value instead of the one from the schema.

-B dtxmlconv option

The backup file (-B) option of the `dtxmlconv` utility command will have a parameter to specify the name of the backup file to be created. If no option is specified, a default backup file with the same name and the **.omt** extension is created.

-O dtxmlconv option

The overwrite (-O) option of the `dtxmlconv` utility command is used to indicate that if a file with the same name as the file specified with the backup file (-B) option exists, then that file is overwritten. If this option is not specified and a file with the same name as the file specified with the backup file (-B) option does exist, The specified backup file already exists and the overwrite option is not set. error message displays on the console.

-LOG dtxmlconv option

The log (-LOG) option of the `dtxmlconv` utility command is used to enable logging. The log option has an optional parameter to specify the user-defined location or file name, or both, for the `dtxmlconv` log file.

If the user-defined location or file name, or both, is not specified, the results of the converted schema will appear on the screen.

If the user-defined location or file name, or both, is specified, a log file will be created containing the results of the converted schema.

The information will be produced when the `dtxmlconv` utility command is run.

-FAIL dtxmlconv option

The fail (-FAIL) option of the `dtxmlconv` utility command is used to indicate that only the type schemas that had errors during the converting should be included in the `dtxmlconv` log file. If log failures only (-FAIL) option is selected, the `dtxmlconv` log will contain only the schemas that had failed converting. If the log failures only (-FAIL) option is not selected in the command line, all the converted results will be written to the `dtxmlconv` log file.

-APPEND dtxmlconv option

The append (-APPEND) option of the `dtxmlconv` utility command is used to indicate that the current `dtxmlconv` log messages should be appended to the existing `dtxmlconv` log file if it already exists. If the `dtxmlconv` log append (-APPEND) option is selected, the `dtxmlconv` log messages will be appended to the existing file if it already exists; otherwise, a new file will be created. If the (-APPEND) option is not selected, the `dtxmlconv` log file will be created if no log file exists, or it will be overwritten if the log file does exist.

-VERBOSE dtxmlconv option

The verbose (-VERBOSE) option of the dtxmlconv utility command is used to indicate if the verbose dtxmlconv log should be produced. The verbose dtxmlconv log will contain an entry for the schema being converted, the schema name, the conversion start date/time, the conversion end date/time, the converted xml file name, and the result of the dtxmlconv utility command.

-NOLOG dtxmlconv option

The disable log (-NOLOG) option of the dtxmlconv utility command is used to turn off the dtxmlconv logging capability. If the disable logging option is selected, log file will not be produced. The (-LOG) optional (-APPEND, and -VERBOSE) options cannot be used with the disable log (-NOLOG) option. If these options are selected with the (-NOLOG) option, the command line is invalid, and the Invalid argument in command line. error message will display on the console.

If you specify both the (-NOLOG) and (-LOG) options, the command line is invalid, and the Disable log option (-NOLOG) cannot be specified when log messages command line option (-LOG) is specified error message will display on the console.

Command line help for dtxmlconv

There is help that describes the dtxmlconv utility command usage in the command line.

To view the command line help for the dtxmlconv utility command:

Enter dtxmlconv at the command line prompt in the *install_dir*.
The syntax and descriptions of all the options appear on the console.

Using the dtxmlconv command

Use the dtxmlconv utility command to convert schemas that were generated in earlier versions (before 8.0) of IBM Transformation Extender from the command line, outside the schema designer GUI.

- [dtxmlconv example](#)

dtxmlconv example

The following example shows how you can use the command:

```
install_dir> dtxmlconv myschema.mtt  
-S C:\MyDev\ConversionResults\mydtd.dtd  
-LOG -APPEND
```

When you run the example, the following result occurs:

- If the **myschema.mtt** schema file is converted successfully, the `Conversion succeeded.` message displays on the console.
- The **mydtd.dtd** DTD file, specified as the input parameter and represents the new XML grammar, is used to generate the schema.
- If the **myschema.log** log file does not exist, it is produced in the same directory as the schema. If the **myschema.log** log file already exists, the log messages are appended to the existing file.

tbccconv utility command

The tbccconv utility command is used to automatically convert certain objects in a schema sized in bytes to characters from the command line.

The behavior of the command depends on the object types in the schema being converted (Text, Syntax, Number, and Date & Time) and the specific command options you use. The object types are represented by the Item Subclass property values in the schema.

There are two ways to use the tbccconv utility command that specifically affect how the conversion is done.

- without the pad sizing (-P) option ([tbccconv default behavior \(without -P option\)](#))
- with the pad sizing ([-P tbccconv option](#))

When run in batch mode, the tbccconv utility command returns 0 if the bytes-to-characters schema conversion is successful and 1 if it is not successful. It is a batch operation, which can also be used for batch automation.

The **.mtt** file is a required field and is the file name of the schema file that needs to be converted. If the full path of the schema file name is not specified, the tbccconv utility command will search for the schema file specified in the current directory.

See ["Troubleshooting"](#) for details about capturing and evaluating the return code resulting from the execution of this utility command.

- [Syntax summary for tbccconv](#)
The tbccconv utility command is used to convert certain objects that are sized in bytes to be sized in characters.
- [Utility command options for tbccconv](#)
The following command options are available with the tbccconv utility command:
 - [Command line help for tbccconv](#)

- [Using the tbccconv command](#)

Use the tbccconv utility command to convert certain objects in a schema sized in bytes to characters from the command line, outside the schema designer GUI.

Syntax summary for tbccconv

The tbccconv utility command is used to convert certain objects that are sized in bytes to be sized in characters.

None of the options in the tbccconv utility command are case sensitive. The following is the syntax of the tbccconv utility command:

```
tbccconv <.mtt file>
  [-P]
  [-R [<.dbe file name/location >]]
  [(- -LOG [
  | <log file name/location >]
  [-FAIL] [-VERBOSE] [-APPEND])
  | -NOLOG]
```

The bytes-to-character schema conversion utility expects to find a schema (.mtt) file present after the tbccconv utility command when other options are also specified. Therefore, if you do not specify an .mtt file but you do specify options, the bytes-to-character conversion utility will try to interpret the first option it encounters as an .mtt file. It will fail and display a message on the console that it could not find an .mtt file.

Utility command options for tbccconv

The following command options are available with the tbccconv utility command:

• -APPEND tbccconv Option	• -P tbccconv Option
• -FAIL tbccconv Option	• -R tbccconv Option
• -LOG tbccconv Option	• -VERBOSE tbccconv Option
• -NOLOG tbccconv Option	

If you type tbccconv with no options, the command line help that describes the command will display on your screen.

- [-APPEND tbccconv option](#)

The append (-APPEND) option of the tbccconv utility command is used to indicate that the current tbccconv log messages should be appended to the tbccconv log file if it already exists.

- [-FAIL tbccconv option](#)

The fail (-FAIL) option of the tbccconv utility command is used to indicate that only the schema that had errors during the byte-to-characters conversion should be included in the tbccconv log file.

- [-LOG tbccconv option](#)

The log (-LOG) option of the tbccconv utility command is used to enable logging.

- [-NOLOG tbccconv option](#)

The disable log (-NOLOG) option of the tbccconv utility command is used to turn off the tbccconv logging capability.

- [-P tbccconv option](#)

The pad sizing (-P) option of the tbccconv utility command is used to indicate for those items that have the PAD characteristic turned on, the PAD sizing value for the SizedAs property will be changed from Bytes to Characters. It also determines how the PAD value is sized when the data is being validated and written as output.

- [-R tbccconv option](#)

The results file (-R) option of the tbccconv utility command has a parameter to specify the path and the file name for the conversion results file produced by the schema bytes-to-characters conversion utility.

- [-VERBOSE tbccconv option](#)

The verbose (-VERBOSE) option of the tbccconv utility command is used to indicate if a verbose tbccconv log should be produced.

-APPEND tbccconv option

The append (-APPEND) option of the tbccconv utility command is used to indicate that the current tbccconv log messages should be appended to the tbccconv log file if it already exists.

If the tbccconv log append (-APPEND) option is selected, the tbccconv log messages will be appended to the file if it already exists. If it does not exist, a new log file will be created. If the (-APPEND) option is not selected, the tbccconv log file will be created if no log file exists, or it will be overwritten if the log file does exist.

-FAIL tbccconv option

The fail (-FAIL) option of the tbccconv utility command is used to indicate that only the schema that had errors during the byte-to-characters conversion should be included in the tbccconv log file.

If the log failures only (-FAIL) option is selected, the tbccconv log will contain only the schema that had failed during the conversion. If the log failures only (-FAIL) option is not selected in the command line, all the converted results will be written to the tbccconv log file.

-LOG tbccconv option

The log (-LOG) option of the `tbccconv` utility command is used to enable logging.

The log option has an optional parameter to specify the user-defined location or file name, or both, for the `tbccconv` log file.

If the user-defined location or file name, or both, is not specified, the results of the converted data will appear on the screen.

If the user-defined location or file name, or both, is specified, a log file will be created containing the results of the converted data.

The information will be produced when the `tbccconv` utility command is run.

-NOLOG tbccconv option

The disable log (-NOLOG) option of the `tbccconv` utility command is used to turn off the `tbccconv` logging capability.

If the disable logging option is selected, the log file will not be produced. The (-LOG) optional (-APPEND and -VERBOSE) options cannot be used with the disable log (-NOLOG) option. If these options are selected with the (-NOLOG) option, the command line is invalid, and the `Invalid argument in command line.` error message will display on the console.

-P tbccconv option

The pad sizing (-P) option of the `tbccconv` utility command is used to indicate for those items that have the PAD characteristic turned on, the PAD sizing value for the `SizedAs` property will be changed from Bytes to Characters. It also determines how the PAD value is sized when the data is being validated and written as output.

Items that have the Pad characteristic are Text, Date & Time and Number. Syntax items have no Pad property.

There are no additional parameters for the pad sizing (-P) option.

For those objects that meet this criteria (the Pad property is set to Yes, and Byte is specified for Text, Date & Time and Number items for the `SizedAs` property in the schema), when the `tbccconv` utility command is run with the -P option, the value for the `SizedAs` property set as Byte will be changed to Character.

-R tbccconv option

The results file (-R) option of the `tbccconv` utility command has a parameter to specify the path and the file name for the conversion results file produced by the schema bytes-to-characters conversion utility.

The specified parameter could be the name of the directory where the conversion results file will be written, the full file path specification in which a wildcard can be used to represent the conversion result file, or the full path in which the complete filename can be specified.

The default naming convention for the conversion results file is: `schema_name.dbe`.

If a wildcard is specified in the full path, for example, `tbccconv myschema.mtt -R C:\MyDev\ConversionResults\2002-02-28_*.err`, the conversion results file will be produced in the **C:\MyDev\ConversionResults** directory with the **2002-02-28_myschema.err** naming convention.

If the complete file name is specified, for example, `tbccconv myschema.mtt -R C:\MyDev\ConversionResults\MyConversionResults.txt`, the **MyConversionResults.txt** file will be produced in the **C:\MyDev\ConversionResults** directory.

If the (-R) option is not specified, the conversion results will be written to the console, and if (-R) is specified without a parameter, the conversion results will be written to the same directory as the schema file using the default `schema_name.dbe` naming convention.

-VERBOSE tbccconv option

The verbose (-VERBOSE) option of the `tbccconv` utility command is used to indicate if a verbose `tbccconv` log should be produced.

The verbose `tbccconv` log will contain an entry for the schema being converted, the names of the schema, the conversion start date/time, the conversion end date/time, the converted file names, and the result of the `tbccconv` utility command. If the verbose (-VERBOSE) option is not selected, a concise version of the `tbccconv` log will be produced, which includes an entry for the schema, schema name, and conversion result.

Command line help for tbccconv

There is help that describes the `tbccconv` utility command usage in the command line.

To view the command line help for the `tbccconv` utility command:

Enter `tbccconv` at the command line prompt in the *install_dir*.

The syntax and descriptions of all the options appear on the console.

Using the tbccconv command

Use the `tbccconv` utility command to convert certain objects in a schema sized in bytes to characters from the command line, outside the schema designer GUI.

When you use the `tbccconv` utility command, it automatically saves the original schema .mtt file.

- [**tbccconv default behavior \(without -P option\)**](#)

The default usage of the `tbccconv` utility command is without the pad sizing (-P) option specified.

- [**tbccconv behavior \(with -P option\)**](#)

The `tbccconv` utility command can be used with the pad sizing (-P) option specified.

tbccconv default behavior (without -P option)

The default usage of the `tbccconv` utility command is without the pad sizing (-P) option specified.

When the `tbccconv` utility command is used without the -P option, only Text and Syntax items with values for Bytes specified for the Min and optionally Max properties will be converted to Min and Max Characters properties, and the Min and Max Bytes sizes will be reset, meaning that Min Bytes size will be automatically set to 0, and the Max Bytes size will be set with no byte constraint. It does not affect Date & Time and Number items, which are sized according to the existing formats used.

By leaving no value for the Min property under Bytes, the Min property will be automatically set to 0. By leaving no value for the Max property under Bytes, there will not be a byte constraint.

- [**tbccconv default behavior examples**](#)

The following table lists the behavior of the `tbccconv` utility command depending on the values of the Min and Max Bytes properties.

- [**tbccconv default syntax example**](#)

This is an example of the `tbccconv` utility command default syntax, which is without the pad sizing (-P) option.

tbccconv default behavior examples

The following table lists the behavior of the `tbccconv` utility command depending on the values of the Min and Max Bytes properties.

When...	Behavior
• Min Bytes property for a type is set to 0 and Max Bytes property is not set to a value.	<ul style="list-style-type: none">Min and Max Bytes properties will not be converted to the Min and Max Characters properties because <code>tbccconv</code> will interpret this type as having no Min and Max Bytes properties set.Min Bytes and Characters sizes will be automatically set to 0.For example, if the Min Bytes property for a type is set to 0 and the Max Bytes property is not set, they will not be converted to the Min and Max Characters property and the Min Bytes and Characters sizes will be automatically set to 0.See example 1: "No constraints"
• Min Bytes property for a type is set to a value but Max Bytes property is not set to a value.	<ul style="list-style-type: none">Min Bytes property will be converted to the Min Characters property.Min Bytes size will be automatically set to 0.Max Bytes property will be not be converted to the Max Characters property.For example, if the Min Bytes property for a type is set to 10 and the Max Bytes property is not set, the Min Bytes property will be converted to the Min Characters property as 10, the Min Bytes size will be automatically set to 0 and the Max Bytes property will not be converted to the Max Characters property.See example 2: "Min Bytes constraints"
• Min Bytes property for a type is not set to a value but Max Bytes property is set to a value.	<ul style="list-style-type: none">Min Bytes property will be not be converted to the Min Characters property.Min Bytes and Characters sizes will be automatically set to 0.Max Bytes property will be converted to the Max Characters property.Max Bytes size will be reset.For example, if the Min Bytes property for a type is not set and the Max Bytes property is set to 10, the Min Bytes property will not be converted to the Min Characters property, the Min Bytes and Characters sizes will be automatically set to 0, the Max Bytes property will be converted to the Max Characters property as 10 and the Max Bytes property will be reset.See example 3: "Max Bytes constraints"
• Both Min and Max Bytes properties for a type are set to values.	<ul style="list-style-type: none">Min and Max Bytes properties will be converted to the Min and Max Characters properties.Min Bytes size will be automatically set to 0.Max Bytes size will be reset.For example, if the Min and Max Bytes properties for a type are set to 10 and 20, they will be converted to the Min and Max Characters properties as 10 and 20, the Min Bytes size will be automatically set to 0 and the Max Bytes size will be reset.See example 4: "Min and Max Bytes constraints"

Conversion activity will be reported in the log file.

- [**tbccconv example 1 \(no constraints\)**](#)

This is an example of how to use the `tbccconv` utility command to convert certain objects in schemas sized in bytes to characters from the command line where there are no Bytes constraints.

- [**tbccconv example 2 \(Min Bytes constraints\)**](#)

This is an example of how to use the `tbccconv` utility command to convert certain objects in schemas sized in bytes to characters from the command line where there are Min Bytes constraints.

- [**tbccconv example 3 \(Max Bytes constraints\)**](#)

This is an example of how to use the `tbccconv` utility command to convert certain objects in schemas sized in bytes to characters from the command line where there are Max Bytes constraints.

- **tbccconv example 4 (Min and Max Bytes constraints)**

This is an example of how to use the `tbccconv` utility command to convert certain objects in schemas sized in bytes to characters from the command line where there are Min and Max Bytes constraints.

tbccconv example 1 (no constraints)

This is an example of how to use the `tbccconv` utility command to convert certain objects in schemas sized in bytes to characters from the command line where there are no Bytes constraints.

To demonstrate this example, assume the schema you want to convert has a `Text` Item Subclass property with the following size properties and values:

```
Size (content) |
Min           |
Bytes         |0
Characters    |
Max           |
Bytes         |
Characters    |
```

The converted schema will have the following size properties and values:

```
Size (content) |
Min           |
Bytes         |0
Characters    |0
Max           |
Bytes         |
Characters    |
```

tbccconv example 2 (Min Bytes constraints)

This is an example of how to use the `tbccconv` utility command to convert certain objects in schemas sized in bytes to characters from the command line where there are Min Bytes constraints.

To demonstrate this example, assume the schema you want to convert has a `Text` Item Subclass property with the following size properties and values:

```
Size (content) |
Min           |
Bytes         |10
Characters    |
Max           |
Bytes         |
Characters    |
```

The converted schema will have the following size properties and values:

```
Size (content) |
Min           |
Bytes         |0
Characters    |10
Max           |
Bytes         |
Characters    |
```

tbccconv example 3 (Max Bytes constraints)

This is an example of how to use the `tbccconv` utility command to convert certain objects in schemas sized in bytes to characters from the command line where there are Max Bytes constraints.

To demonstrate this example, assume the schema you want to convert has a `Text` Item Subclass property with the following size properties and values:

```
Size (content) |
Min           |
Bytes         |0
Characters    |
Max           |
Bytes         |10
Characters    |
```

The converted schema will have the following size properties and values:

```
Size (content) |
Min           |
Bytes         |0
Characters    |0
Max           |
Bytes         |
Characters    |
```

```
Bytes      |
Characters | 10
```

tbccconv example 4 (Min and Max Bytes constraints)

This is an example of how to use the `tbccconv` utility command to convert certain objects in schemas sized in bytes to characters from the command line where there are Min and Max Bytes constraints.

To demonstrate this example, assume the schema you want to convert has a `Text` Item Subclass property with the following size properties and values:

```
Size (content) |
  Min          |
    Bytes     | 10
    Characters |
  Max          |
    Bytes     | 20
    Characters |
```

The converted schema will have the following size properties and values:

```
Size (content) |
  Min          |
    Bytes     | 0
    Characters | 10
  Max          |
    Bytes     | 20
    Characters | 20
```

tbccconv default syntax example

This is an example of the `tbccconv` utility command default syntax, which is without the pad sizing (`-P`) option.

The following command syntax example shows how you can use the `tbccconv` utility command with its default settings and logging options:

```
install_dir> tbccconv myschema.mtt
              -LOG -APPEND
```

When you run the example, the following result occurs:

- If the `myschema.mtt` file is converted successfully, the `Conversion succeeded.` message displays on the console.
- If the `myschema.log` log file does not exist, it is produced in the same directory as the `tschema` file. If the `myschema.log` log file already exists, the log messages are appended to the existing file.

tbccconv behavior (with -P option)

The `tbccconv` utility command can be used with the pad sizing (`-P`) option specified.

- [tbccconv example with -P option](#)
This is an example of how to use the `tbccconv` utility command with the pad sizing (`-P`) option to convert certain objects in schemas sized in bytes to characters from the command line.
- [tbccconv example syntax with -P option](#)
This is an example of the `tbccconv` utility command syntax with the pad sizing (`-P`) option.

tbccconv example with -P option

This is an example of how to use the `tbccconv` utility command with the pad sizing (`-P`) option to convert certain objects in schemas sized in bytes to characters from the command line.

To demonstrate this example, assume the schema you want to convert has a `Text` Item Subclass property with the following pad and size properties and values:

```
Pad      | Yes
Value   | <sp>
Padded to | Fixed Size
Length   | 0
SizedAs  | Bytes
Justify   | Left
Apply pad | Fixed Group
```

The converted schema will have the following property values set for the `Text` Item Subclass property:

```
Pad      | Yes
Value   | <sp>
```

Padded to	Fixed Size
Length	0
SizedAs	Characters
Justify	Left
Apply pad	Fixed Group

tbccconv example syntax with -P option

This is an example of the `tbccconv` utility command syntax with the pad sizing (`-P`) option.

The following command syntax example shows how you can use the `tbccconv` utility command with the `-P` option in addition to logging options.

```
install_dir> tbccconv myschema.mtt
          -P
          -LOG -APPEND
```

When you run the example, the following result occurs:

- If the `myschema.mtt` schema file is converted successfully, the `Conversion succeeded.` message displays on the console.
- If the `myschema.log` log file does not exist, it is produced in the same directory as the schema file. If the `myschema.log` log file already exists, the log messages are appended to the existing file.

Utility commands for map designer

Map designer utility commands are used to compile maps, import to maps, export maps, and generate map report HTML files from map source files.

- [mcompile utility command](#)
The `mcompile` utility command is used to compile one or more maps from the command line, outside of the Map Designer GUI.
- [mimport utility command](#)
The `mimport` utility command is used to import an xml file to a map source file from the command line, outside the Map Designer GUI.
- [mexport utility command](#)
The `mexport` utility command is used to export a map source file from the command line, outside the Map Designer.
- [mreport utility command](#)
The `mreport` utility command is used to generate a map report HTML file containing information about a map source file, from the command line, outside the Map Designer.

mcompile utility command

The `mcompile` utility command is used to compile one or more maps from the command line, outside of the Map Designer GUI.

The `mcompile` utility command returns 0 if the compilation is successful, and 1 if not. It is a batch operation that can also be used for automation. `mcompile` is supported on Windows and Linux platforms.

See "[Troubleshooting](#)" for details about capturing and evaluating the return code resulting from running this utility command.

- [mcompile utility command syntax summary](#)
The syntax summary for the `mcompile` utility command.
- [mcompile utility command on Linux platform](#)
The syntax summary for the `mcompile` utility command on Linux platform.
- [mcompile utility command options](#)
The options that can be used with the `mcompile` utility command.
- [mcompile command line help](#)
You can view command line help for the `mcompile` utility command.
- [mcompile command usage](#)
Use the `mcompile` utility command to compile one or more maps from the command line, outside the Map Designer GUI.

mcompile utility command syntax summary

The syntax summary for the `mcompile` utility command.

The `mcompile` utility command is used to compile maps.

The `.mms` file is a required field and is the name of the map source file that needs to be compiled. The map source file can have any number of executable maps in it. The executable maps shall be defined as a map for which all sources and targets have the minimum amount of information specified for the actual source or target of data for that card's data. If the full path of the map source file name is not specified, the `mcompile` utility command will search for the map source file in the current directory. If the `.mms` file is not specified, the `Not a valid command line - map source file name should be given for the command` error message will display on the console.

The executable map name is the only option in the `mcompile` utility command that is case sensitive. The following is the syntax of the `mcompile` utility command:

```
mcompile <.mms file> -L [-DP | -TX]
```

```

- or -
mcompile <.mms file> -A
[-K] [-E] [-NO]
[-DP] [-TX]
[-P <platform>[;<platform>[;...]]]
[-O <.mmc file name/location>]
[-R <.mme file name/location>]
[( -LOG [
| <log file name/location>]
[-FAIL] [-VERBOSE] [-APPEND])
| -NOLOG]
- or -
mcompile <.mms file>
<executable map>
[-E]
[-NO]
[-P <platform>[;<platform>[;...]]]
[-O <.mmc file name/location>]
[-R <.mme file name/location>]
[( -LOG [
| <log file name/location>]
[-FAIL] [-VERBOSE] [-APPEND])
| -NOLOG]

```

mcompile utility command on Linux platform

The syntax summary for the mcompile utility command on Linux platform.

Syntax

The following is the syntax for the `mcompile` utility command on Linux platform:

```
mcompile <mms filename> [<executable map name> -O <compiled map file name>|-A|-L] [-P [WINDOWS; LINUX; AIX; ZOS; ZLINUX; PLINUX]]
```

<mms filename>: map source file, contains design of the maps, does not support wild characters.

<executable map name>: executable map name from the mms file, does not support wild characters.

-A: compile all maps in the specified mms file, the name of the compiled file name will be generated automatically <full path to mms name>.<executable map name>.mmc.

-O <compiled map file name>: the specified executable map will be compiled as specified.

-L List: All executable maps from the specified map source file.

The following are the examples of the `mcompile` utility command on Linux platform:

1. Compile processCVSDelimited:


```
>mcompile /myfolder/delimited.mms processCVSDelimited -O /my mmc folder/delimited.mms.processCVSDelimited.aol -P LINUX;WINDOWS
```
2. Compile processTABDelimited and processCVSDelimited, will generate two compiled map files, /myfolder/delimited.mms.processCVSDelimited.mmc and /myfolder/delimited.mms.processTABDelimited.mmc:


```
>mcompile /myfolder/delimited.mms -A -P LINUX;WINDOWS
```
3. List executable maps present in the mms file:


```
>mcompile /myfolder/delimited.mms -L
```

mcompile utility command options

The options that can be used with the `mcompile` utility command.

The following command options are available with the `mcompile` utility command:

• -L mcompile Option	• -TX mcompile Option
• -A mcompile Option	• -R mcompile Option
• -K mcompile Option	• -LOG mcompile Option
• -E mcompile Option	• -FAIL mcompile Option
• -P mcompile Option	• -APPEND mcompile Option
• -O mcompile Option	• -VERBOSE mcompile Option
• -NO mcompile Option	• -NOLOG mcompile Option
• -DP mcompile Option	

If you type `mcompile` with no options, the command line help that describes the command will display on your screen.

- **-L mcompile option**
The -L option that can be used with the mcompile utility command.
- **-A mcompile option**
The -A option that can be used with the mcompile utility command.
- **-K mcompile option**
The -K option that can be used with the mcompile utility command.
- **-E mcompile option**
The -E option that can be used with the mcompile utility command.
- **-P mcompile option**
The -P keyword of the **mcompile** utility command specifies one or more platforms for which to compile the executable map.
- **-O mcompile option**
The -O option that can be used with the mcompile utility command.
- **-NO mcompile option**
The -NO option that can be used with the mcompile utility command.
- **-DP mcompile option**
The -DP option that can be used with the mcompile utility command.
- **-TX mcompile option**
The -TX option that can be used with the mcompile utility command.
- **-R mcompile option**
The -R option can be used with the mcompile utility command.
- **-LOG mcompile option**
The -LOG option that can be used with the mcompile utility command.
- **-FAIL mcompile option**
The -FAIL option that can be used with the mcompile utility command.
- **-APPEND mcompile option**
The -APPEND option that can be used with the mcompile utility command.
- **-VERBOSE mcompile option**
The -VERBOSE option that can be used with the mcompile utility command.
- **-NOLOG mcompile option**
The -NOLOG option that can be used with the mcompile utility command.

-L mcompile option

The -L option that can be used with the mcompile utility command.

The list (-L) option of the **mcompile** utility command is used to list all executable maps that are present in the specified map source file. When this option is selected, the list of all executable maps will display in alphabetical order, each on a separate line. No other option can be selected with the (-L) option in the **mcompile** utility command, except the (-DP) and (-TX) options. If any other option of the **mcompile** utility command is selected with the (-L) option, other than the (-DP) and (-TX) options, the command line is invalid, and the **Not a valid command line - No other option is allowed with '-L' except -DP or -TX.** error message will display on the console.

Using the -L mcompile option with -TX or -DP options

The following table shows how the **mcompile** utility command behaves when you specify the (-L) option along with the different executable map type options.

When you specify the (-L) option along with...	Result
the (-TX) option of the mcompile utility command	mcompile will list IBM Transformation Extender executable maps that are present in the map source file
the (-DP) option of the mcompile utility command	mcompile will list IBM DataPower executable maps that are present in the map source file
both the (-TX) and (-DP) options of the mcompile utility command	The command line is invalid, and the Not a valid command line - No other option is allowed with '-L' except -DP or -TX. error message will display on the console.

-A mcompile option

The -A option that can be used with the mcompile utility command.

The all (-A) option of the **mcompile** utility command is used to compile all executable maps that are present in a map source file. The executable map name should not be specified in the command line if (-A) option is selected. If the executable map name is also specified with the (-A) option, the command line is invalid, and the **Not a valid command line - Executable map name should not be specified when '-A' all maps are selected** error message will display on the console.

Using the -A mcompile option with -TX and -DP options

The following table shows how the **mcompile** utility command behaves when you specify the (-A) option along with the different executable map type options.

When you specify the (-A) option along with...	mcompile will compile...
the (-TX) option of the mcompile utility command	IBM Transformation Extender executable maps that are present in the map source file
the (-DP) option of the mcompile utility command	IBM DataPower executable maps that are present in the map source file

When you specify the (-A) option along with...	mcompile will compile...
both the (-TX) and (-DP) options of the mcompile utility command	both IBM Transformation Extender and IBM DataPower executable maps that are present in the map source file, which is the same result as specifying (-A) without the two executable map type options

-K mcompile option

The -K option that can be used with the mcompile utility command.

The continue (-K) option of the **mcompile** utility command is used to continue the compiling of the executable maps that are present in the map source file when any errors are returned by the map compiler or upon the first **mcompile** error. This option can be used only with the all maps (-A) option. If the continue (-K) option is specified without (-A), the command line is invalid, and the **Not a valid command line - Continue compiling** option cannot be specified without '-A' all maps command option. error message will display on the console. If the continue compilation (-K) option is not selected, the compilation of the map on which the first error has occurred will be completed, and the **mcompile** will terminate. It will not continue with the rest of the executable maps that are present in the map source file. If the continue compilation (-K) option is selected, the map compilation will be performed for all the executable maps that are present in the map source file, regardless of whether the map compiler returns warnings or errors for any of the individual maps.

-E mcompile option

The -E option that can be used with the mcompile utility command.

The errors (-E) option of the **mcompile** utility command is used to control the type of messages (warnings or errors, or both) written to the build results file for each executable map that has been compiled. If the (-E) option is selected, the messages written to the build results file will be limited to map compiler errors only. The warning messages will be ignored and will not be written to the build results file. If the (-E) option is not selected, all the error and warning messages will be written to the build results.

-P mcompile option

The -P keyword of the **mcompile** utility command specifies one or more platforms for which to compile the executable map.

The command supports the following platforms as keywords:

- AIX
- HPITANIUM
- LINUX
- PLINUX
- SOLARIS
- WINDOWS
- ZLINUX
- ZOS

The -P keyword is optional. If you omit it, the **mcompile** command line uses **WINDOWS** as the default platform. If you specify more than one platform, the **mcompile** command compiles a multi-platform map. Delimit each platform with a semi-colon (;). For example:

```
mcompile test.mms -A -P WINDOWS;LINUX;SOLARIS;ZOS
```

The **mcompile** command is invalid and does not compile a map when:

- You specify the -P keyword but omit the platform.
- You specify one or more invalid platforms.

-O mcompile option

The -O option that can be used with the mcompile utility command.

The output (-O) option of the **mcompile** utility command is used to specify, through a parameter, the location or file name, or both, for the resulting compiled map file. There are 2 different cases where the output (-O) option can be used in the **mcompile** command.

In the first case, the output (-O) option is used with the compile all maps (-A) option. The specified parameter could be either the name of the directory where the resulting compiled map files are written or the full file path specification in which a wildcard can be used to represent the compiled map file naming convention. The default naming convention for the compiled map file is **executable_map_name.mmc**. If a wildcard is specified in the full path as **mcompile mymaps.mms -A -O C:\MyDev\Compiledmaps\2002-02-28_*.win**, the compiled maps are produced in the **C:\MyDev\Compiledmaps** directory with the **2002-02-28_<executable_map_name>.win** naming convention.

In the second case, the output (-O) option is used without the compile all maps (-A) option. The specified parameter could be the name of the directory where the resulting compiled map files are written, the full file path specification in which a wildcard can be used to represent the compiled map file, or the full path in which the complete filename can be specified. The default naming convention for the compiled map file is **executable_map_name.mmc**. If a wildcard is specified in the full path as **mcompile mymaps.mms MyExecMap -O C:\MyDev\Compiledmaps*.*.win**, the compiled maps are produced in the **C:\MyDev\Compiledmaps** directory with the **<executable_map_name>.win** naming convention. If the complete file name is specified, for example, **mcompile mymaps.mms MyExecMap -O C:\MyDev\Compiledmaps\MyExecForWin.win**, the **MyExecForWin.win** file is produced in the **C:\MyDev\Compiledmaps** directory.

If the (-o) option is not specified, the compile map is written to the same directory as of the map source file using the default naming convention executable_map_name.mmc. If the (-o) option is specified and the parameter is not provided, the command line is invalid, and the Not a valid command line - The file name for the resulting compiled map file should be provided when -O option is selected. error message displays on the console. If the file name or the path provided for the (-o) option is not valid or returns error while attempting to open or write to the file, the Compilation error - The file/path provided for the option (-O) is not valid, unable to write to the specified directory/file message is written to the mcompile log file if it is enabled.

-NO mcompile option

The -NO option that can be used with the mcompile utility command.

The no overwrite (-NO) option of the mcompile utility command is used to indicate that an existing compiled map file (.mmc) is not to be overwritten. If the no overwrite (-NO) option is specified in the command line of the mcompile utility command, and a file with the same name as the resulting compiled map file exists, it will not be overwritten, the map compilation will fail, and the Compilation error - The compiled map file cannot be overwritten, file already exists message will be written to the mcompile log file, if enabled. If the (-NO) option is not specified, the file with the same name as the compiled map file resulting from the compilation process will be overwritten.

-DP mcompile option

The -DP option that can be used with the mcompile utility command.

The IBM DataPower (-DP) option of the mcompile utility command is used to compile all IBM DataPower maps that are present in the specified map source file. This command option excludes any IBM Transformation Extender maps from being compiled.

For information about IBM DataPower maps, see the IBM Transformation Extender Map Designer documentation.

For information about IBM DataPower, see the IBM DataPower documentation.

Use the -L -DP combination to display a list of the IBM DataPower maps that are present in the map source file.

If you use both -DP and -TX options together with the (-A) option, the mcompile utility command will compile all maps present in the map source file. This behavior is the same as if you use the all (-A) option without specifying both -DP and -TX options together.

Any options used with -DP that are not valid for IBM DataPower maps are ignored. For example, because the platform (-P) option is not valid for IBM DataPower maps, it will not be used in the map compile process.

For more information about the other related mcompile options referenced here, see the following topics:

- [-L mcompile Option](#)
 - [-A mcompile Option](#)
 - [-TX mcompile Option](#)
-

-TX mcompile option

The -TX option that can be used with the mcompile utility command.

The IBM Transformation Extender (-TX) option of the mcompile utility command is used to compile all IBM Transformation Extender executable maps that are present in the specified map source file. This command option excludes any IBM DataPower maps from being compiled.

For information about IBM Transformation Extender maps, see the IBM Transformation Extender Map Designer documentation.

Use the -L -TX combination to display a list of the IBM Transformation Extender maps that are present in the map source file.

If you use both -DP and -TX options together, the default behavior of the mcompile utility command will be to use the all (-A) option and compile all maps present in the map source file.

For more information about the other related mcompile options referenced here, see the following topics:

- [-L mcompile Option](#)
 - [-A mcompile Option](#)
 - [-DP mcompile Option](#)
-

-R mcompile option

The -R option can be used with the mcompile utility command.

The results (-R) option of the mcompile utility command will have a parameter to specify the location or file name, or both, for the build results file produced by the map compiler for each executable map. There are 2 different cases where (-R) can be used in the mcompile utility command.

In the first case, the (-R) option is used with the compile all maps (-A) option. The specified parameter could be either the name of the directory where the build results files will be written or the full file path specification in which a wildcard can be used to represent the executable map naming convention. The default naming convention for the build results file is **executable_map_name.mme**. If a wildcard is specified in the full path as mcompile mymaps.mms -A -R

C:\MyDev\CompiledResults*.err, the build results files will be produced in the C:\MyDev\CompiledResults directory with the executable_map_name.err naming convention.

In the second case, the (-R) option is used without the compile all maps (-A) option. The specified parameter could be the name of the directory where the build results files will be written, the full file path specification in which a wildcard can be used to represent the build result file, or the full path in which the complete filename can be specified. The default naming convention for the build results file is executable_map_name.mme. If a wildcard is specified in the full path as mcompile mymaps.mms MyExecMap -R C:\MyDev\CompiledResults\2002-02-28_*.err, the build results file will be produced in the C:\MyDev\CompiledResults directory with 2002-02-28_MyExecMap.err naming convention. If the complete file name is specified as mcompile mymaps.mms MyExecMap -R C:\MyDev\CompiledResults\MyExecForWin.txt, the MyExecForWin.txt will be produced in the C:\MyDev\CompiledResults directory.

If the (-R) option is not specified, the build results will be written to the console. If the (-R) option is specified, but no parameter is provided, the build results will be written to the same directory as of the map source file using the default executable_map_name.mme naming convention. If the file name or the path provided for the (-R) option is not valid or returns error while attempting to open or write to the file, the Compilation error - The file/path provided for the option (-R) is not valid, unable to write to the specified directory/file message will be written to the mcompile log file, if it is enabled.

-LOG mcompile option

The -LOG option that can be used with the mcompile utility command.

The log (-LOG) option of the mcompile utility command is used to enable logging. The log file will be produced showing the results of each executable map's compilation. (-LOG) has an optional parameter to specify the location or file name, or both, for the mcompile log file. The mcompile log contains all the map compilations resulting from the current mcompile utility command on the command line. The default file name for the mcompile log will be map_source_file_name.mcl. The default location for the mcompile log will be the directory where the source map file is located.

The file name or the location parameter, if specified with the (-LOG) option of the mcompile utility command, could be the name of the directory where the mcompile log file will be written, the full file path specification in which a wildcard can be used to represent the map source file, or the full path in which the complete filename can be specified. The default naming convention for the mcompile log file is map_source_file_name.mcl. If the . is specified with the (-LOG) option, the mcompile log will be written to the default location (directory where the map source file is located) using the default naming convention. If a wildcard is specified in the full path as mcompile mymaps.mms -A -LOG C:\MyDev\CompileResults\2002-02-28_*.build.results, the mcompile log file will be produced in the C:\MyDev\CompileResults directory with the 2002-02-28_mymaps.build.results naming convention. If the complete file name is specified as mymaps.mms -A -LOG C:\MyDev\CompileResults\mymaps_buildresults.txt.txt, the mymaps_buildresults.txt will be produced in the C:\MyDev\CompileResults directory.

If the mcompile utility command is unable to create the mcompile log file, the map compilation will terminate and the Fatal error - Could not start map compilation, unable to create the log file fatal error message will display on the console. If nothing is specified in the command line for the log, by default, the command line assumes the (-LOG) option is included in the command line, and writes the log messages to the console.

-FAIL mcompile option

The -FAIL option that can be used with the mcompile utility command.

The fail (-FAIL) option of the mcompile utility command is used to indicate that only the maps that had compilation errors or warnings during compilation should be included in the mcompile log file. If the log failures only (-FAIL) option is selected, the mcompile log will contain only the maps that had errors or warnings. If the log failures only (-FAIL) option is not selected in the command line, all the compilation results will be written to the mcompile log file.

-APPEND mcompile option

The -APPEND option that can be used with the mcompile utility command.

The append (-APPEND) option of the mcompile utility command is used to indicate that the current mcompile log messages should be appended to the existing mcompile log file if it already exists. If the mcompile log append (-APPEND) option is selected, the mcompile log messages will be appended to the existing file if it already exists; otherwise, a new log file will be created. If the (-APPEND) option is not selected, the mcompile log file will be created if no log file exists, or it will be overwritten if the log file does exist.

-VERBOSE mcompile option

The -VERBOSE option that can be used with the mcompile utility command.

The verbose (-VERBOSE) option of the mcompile utility command is used to indicate that a verbose mcompile log should be produced. The verbose mcompile log will contain an entry for each map compilation attempted, the map name, the compile start date/time, the compile end date/time, the compilation result, and the name of the build results file. If the verbose (-VERBOSE) option is not selected, a concise version of the mcompile log will be produced, which includes an entry for each map compilation attempted, map name, and compilation result.

-NOLOG mcompile option

The -NOLOG option that can be used with the mcompile utility command.

The disable log (-NOLOG) option of the mcompile utility command is used to turn off logging. If the disable logging option is selected, the log file will not be produced. The (-APPEND, -FAIL, and -VERBOSE) options of the (-LOG) option cannot be used with the disable log (-NOLOG) option. If these options are selected with the (-NOLOG) option,

the command line is invalid, and the Not a valid command line - (LOG) optional commands (-APPEND), (-FAIL), (-VERBOSE) cannot be used with disable log command (-NOLOG). error message will display on the console.

mcompile command line help

You can view command line help for the mcompile utility command.

There is help that describes the `mcompile` utility command usage in the command line.

To view the command line help for the mcompile utility command:

Enter `mcompile` at the command line prompt in the *install_dir*.

The syntax and descriptions of all the options appear on the console.

mcompile command usage

Use the `mcompile` utility command to compile one or more maps from the command line, outside the Map Designer GUI.

The following examples show how you can use the command.

- [Scenario: run mcompile to display all executable maps](#)

In this scenario, you run the mcompile utility command to display all executable maps.

- [Scenario: run mcompile to compile all executable maps](#)

In this scenario, you run the mcompile utility command to compile all executable maps.

- [Scenario: run mcompile to compile an executable map](#)

In this scenario, you run the mcompile utility command to compile an executable map.

Scenario: run mcompile to display all executable maps

In this scenario, you run the mcompile utility command to display all executable maps.

Use the `mcompile` utility command to display all the executable maps on the console from the command line, outside the Map Designer GUI.

The following example shows how you can use the command with the `(-L)` option to display all the executable maps:

```
install_dir> mcompile mymaps.mms  
-L
```

Result:

```
Executable maps in mymaps.mms: FristMyExecMap MyExecMap NewExecMap  
Total executable maps: 3
```

The list of all executable maps for the specified map source file displays in alphabetical order on the console. The total number of executable maps display as well.

- [Scenario: run mcompile with -DP and -TX options](#)

In this scenario, you run the mcompile utility command with the `-DP` and `-TX` options.

Scenario: run mcompile with -DP and -TX options

In this scenario, you run the mcompile utility command with the `-DP` and `-TX` options.

In these examples, `myDPandTXmaps.mms` map source file contains five executable maps. Three are IBM DataPower executable maps (`MyDPEExec1Map`, `MyDPEExec3Map` and `MyDPEExec4Map`) and two are IBM Transformation Extender executable maps (`MyTXExec2Map` and `MyTXExec5Map`). The examples use combinations of the `(-DP)` and `(-TX)` mcompile command options along with the list `(-L)` option. The list of all executable maps for the specified map type and map source file will display in alphabetical order on the console. The total number of executable maps will display as well.

Display all IBM DataPower executable maps

An extension of the example is to use the `mcompile` utility command to display all the IBM DataPower executable maps on the console from the command line, outside the Map Designer GUI.

The following example shows how you can use the command with the `(-L)` option along with the `(-DP)` option:

```
install_dir> mcompile myDPandTXmaps.mms  
-L -DP
```

Result:

```
Executable maps in myDPandTXmaps.mms: MyDPEExec1Map MyDPEExec3Map MyDPEExec4Map  
Total executable maps: 3
```

Display all IBM Transformation Extender executable maps

An extension of the example is to use the `mcompile` utility command to display all the IBM Transformation Extender executable maps on the console from the command line, outside the Map Designer GUI.

The following example shows how you can use the command with the `(-L)` option along with the `(-TX)` option:

```
install_dir> mcompile myDPandTxmaps.mms  
-L -TX
```

Result:

```
Executable maps in myDPandTxmaps.mms: MyTXExec2Map MyTXExec5Map  
Total executable maps: 2
```

Display all IBM DataPower and IBM Transformation Extender executable maps

An extension of the example is to use the `mcompile` utility command to display all the IBM DataPower and IBM Transformation Extender executable maps on the console from the command line, outside the Map Designer GUI.

The following example shows how you can use the command with the `(-L)` option without specifying the `(-TX)` and `(-DP)` options:

```
install_dir> mcompile myDPandTxmaps.mms  
-L
```

Result:

```
Executable maps in myDPandTxmaps.mms: MyDPExec1Map MyTXExec2Map  
MyDPExec3Map MyDPExec4Map MyTXExec5Map  
Total executable maps: 5
```

Scenario: run `mcompile` to compile all executable maps

In this scenario, you run the `mcompile` utility command to compile all executable maps.

Use the `mcompile` utility command to compile all the executable maps from the command line, outside the Map Designer GUI.

The following example shows how you can use the command with the `(-A)` option to compile all the executable maps:

```
install_dir> mcompile mymaps.mms  
-A -E -P WINDOWS  
-O C:\MyDev\Compiledmaps\*.*.mmc  
-R C:\MyDev\CompiledResults\mymaps_*.*.mme  
-LOG C:\MyDev\CompiledResults\  
*.buildresults.txt  
-FAIL -APPEND
```

When you run the example, the following result occurs:

- If all the maps are compiled successfully, the `mcompile - completed successfully.` message displays on the console.
- The `FristMyExecMap.mmc`, `MyExecMap.mmc`, `NewExecMap.mmc` compiled map files are produced for WINDOWS in the `C:\MyDev\Compiledmaps` directory.
- The `mymaps_FristMyExecMap.mme`, `mymaps_MyExecMap.mme`, `mymaps_NewExecMap.mme` build results files are produced in the `C:\MyDev\CompiledResults` directory.

If the `mymaps_buildresults.txt` log file does not exist, it is produced in the `C:\MyDev\CompiledResults` directory. If the `mymaps_buildresults.txt` log file already exists, only the failed maps' log messages are appended to the existing file.

- [Scenario: run `mcompile` with `-DP` and `-TX` options](#)

In this scenario, you run the `mcompile` utility command with `-DP` and `-TX` options to compile all executable maps.

Scenario: run `mcompile` with `-DP` and `-TX` options

In this scenario, you run the `mcompile` utility command with `-DP` and `-TX` options to compile all executable maps.

In these examples, `myDPandTxmaps.mms` map source file contains five executable maps. Three are DP executable maps (`MyDPExec1Map`, `MyDPExec3Map` and `MyDPExec4Map`) and two are TX executable maps (`MyTXExec2Map` and `MyTXExec5Map`). The examples use combinations of the `(-DP)` and `(-TX)` `mcompile` command options along with the all `(-A)` option. All of the executable maps for the specified map type and map source file will be compiled.

Compile all IBM DataPower executable maps

An extension of the example is to use the `mcompile` utility command to compile all the IBM DataPower executable maps from the command line, outside the Map Designer GUI. The following command is an example of using the `(-A)` option on the `mcompile` utility command along with the `(-DP)` option:

```
install_dir> mcompile myDPandTxmaps.mms  
-A -DP
```

Result:

If all the executable IBM DataPower maps (MyDPExec1Map, MyDPExec3Map and MyDPExec4Map) in the myDPandTXmaps.mms map source file are compiled successfully, the `mcompile - completed successfully.` message will display on the console.

Compile all IBM Transformation Extender executable maps

An extension of the example is to use the `mcompile` utility command to compile all the IBM Transformation Extender executable maps from the command line, outside the Map Designer GUI. The following command is an example of using the `(-A)` option on the `mcompile` utility command along with the `(-TX)` option:

```
install_dir> mcompile myDPandTXmaps.mms  
-A -TX
```

Result:

If all the executable IBM Transformation Extender maps (MyTXExec2Map and MyTXExec5Map) in the myDPandTXmaps.mms map source file are compiled successfully, the `mcompile - completed successfully.` message will display on the console.

Compile all IBM DataPower and IBM Transformation Extender executable maps

An extension of the example is to use the `mcompile` utility command to display all the IBM DataPower and IBM Transformation Extender executable maps from the command line, outside the Map Designer GUI. The following command is an example of using the `(-A)` option on the `mcompile` utility command without specifying the `(-TX)` and `(-DP)` options:

```
install_dir> mcompile myDPandTXmaps.mms  
-A
```

Result:

If all the executable IBM DataPower and IBM Transformation Extender maps (MyDPExec1Map, MyTXExec2Map, MyDPExec3Map, MyDPExec4Map and MyTXExec5Map) in the myDPandTXmaps.mms map source file are compiled successfully, the `mcompile - completed successfully.` message will display on the console.

Scenario: run mcompile to compile an executable map

In this scenario, you run the `mcompile` utility command to compile an executable map.

Use the `mcompile` utility command to compile a specific executable map from the command line, outside the Map Designer GUI.

The following example shows how you can use the command to compile a specific executable map:

```
install_dir> mcompile MyExecMap  
-E -P WINDOWS  
-O C:\MyDev\Compiledmaps\MyCompiledMap.mmc  
-R C:\MyDev\CompiledResults\  
MyExecMapLogFile.err  
-NOLOG
```

When you run the example, the following result occurs:

- If the **MyExecMap** map compiled successfully, the `mcompile - completed successfully.` message displays on the console.
- The **MyCompiledMap.mmc** compiled map file is produced for **WINDOWS** in the **C:\MyDev\Compiledmaps** directory.
- The **MyExecMapLogFile.err** build results file is produced in the **C:\MyDev\CompiledResults** directory.

No log file is produced because (`-NOLOG`) option has been selected.

mimport utility command

The `mimport` utility command is used to import an xml file to a map source file from the command line, outside the Map Designer GUI.

The `mimport` utility command returns 0 if the map import is successful and 1 if it is not successful. It is a batch operation and can also be used for automation.

See "[Troubleshooting](#)" for details about capturing and evaluating the return code resulting from the execution of this utility command.

- **[mimport syntax summary](#)**
This is the syntax that is used with the `mimport` utility command.
- **[mimport utility command options](#)**
These are the command options that can be used with the `mimport` utility command.
- **[mimport command line help](#)**
You can access command line help for the `mimport` utility command.
- **[mimport command usage](#)**
You can use the `mimport` command to import an XML file to the map source file from the command line.

mimport syntax summary

This is the syntax that is used with the mimport utility command.

mimport is the name of the utility command used to import xml files to maps.

The .xml file is a required field and the file name of the xml source file. If the full path of the xml source file name is not specified, the mimport utility command will search for the xml source file in the current directory.

The following is the syntax of the mimport utility command:

```
mimport <.xml file>
[-NO]
[-O <.mms file name/location>]
[(-LOG [. | <log file name/location>]
[-FAIL] [-APPEND] [-VERBOSE])
| -NOLOG]
```

None of the options in the mimport utility command are case sensitive. Typing mimport with no options will display the command summary on the console.

mimport utility command options

These are the command options that can be used with the mimport utility command.

The following command options are available with the mimport utility command:

• -O mimport Option	• -APPEND mimport Option
• -NO mimport Option	• -VERBOSE mimport Option
• -LOG mimport Option	• -NOLOG mimport Option
• -FAIL mimport Option	

[-O mimport option](#)

The -O option can be used with the mimport utility command.

[-NO mimport option](#)

The -NO option can be used with the mimport utility command.

[-LOG mimport option](#)

The -LOG option can be used with the mimport utility command.

[-FAIL mimport option](#)

The -FAIL option can be used with the mimport utility command.

[-APPEND mimport option](#)

The -APPEND option can be used with the mimport utility command.

[-VERBOSE mimport option](#)

The -VERBOSE option can be used with the mimport utility command.

[-NOLOG mimport option](#)

The -NOLOG option can be used with the mimport utility command.

-O mimport option

The -O option can be used with the mimport utility command.

The output (-o) option of the mimport utility command is used to specify, through a parameter, the location or file name, or both, for the imported map source file produced by importing the xml file. The specified parameter could be the name of the directory where the imported map source file is written, the full file path specification in which a wildcard can be used to represent the imported map source file, or the full path in which the complete filename can be specified. The default naming convention for the imported map source file is **xml_file_name.mms**. If a wildcard is specified in the full path, for example, mimport myxmlfile.xml -O C:\MyDev\ImportResults\2002-02-28_*.mms, the resulting map source file is produced in the **C:\MyDev\ImportResults** directory with the **2002-02-28_myxmlfile.mms** naming convention. If the complete file name is specified, as in mimport myxmlfile.xml -O C:\MyDev\ImportResults\Myxmlmaps.mms, the **Myxmlmaps.mms** map source file is produced in the **C:\MyDev\ImportResults** directory. If the import output (-o) option is selected and if the file name is not specified, the command line is invalid, and the Not a valid command line - the file name parameter is required if import command option (-O) is selected error message displays on the console.

-NO mimport option

The -NO option can be used with the mimport utility command.

The no overwrite (-NO) option of the mimport utility command is used to indicate that an existing imported mms file (**.mms**) is not to be overwritten. If the no overwrite (-NO) option is specified in the command line of mimport , and if the file with the name of the map resulting from the imported process file exists, it will not be overwritten, the map importing will fail, and the Map Import error - The imported map source file cannot be overwritten, file already exists message will be written to the mimport log file if enabled. If the (-NO) option is not specified, the file with the name of the map resulting from the importing process will be overwritten.

-LOG mimport option

The **-LOG** option can be used with the `mimport` utility command.

The log **-LOG** option of the `mimport` utility command is used to enable logging. The log file will be produced showing the result of the imported map source file. (**-LOG**) has an optional parameter to specify the location or file name, or both, for the `mimport` log file. The default file name for the `mimport` log will be **xml_source_file_name.log**. The default location for the `mimport` log will be the directory where the xml file is located.

The file name or the location parameter, if specified with the (**-LOG**) option of the `mimport` utility command, could be the name of the directory where the `mimport` log file will be written, the full file path specification in which a wildcard can be used to represent the xml source file, or the full path in which the complete filename can be specified. If the `.` is specified with the (**-LOG**) option, the `mimport` log will be written to the default location (directory where the xml file is located) using the default naming convention. If a wildcard is specified in the full path, for example, `mimport myxmlfile.xml -LOG C:\MyDev\ImportResults\2002-02-28_*.results`, the `mimport` log file will be produced in the **C:\MyDev\ImportResults** directory with the **2002-02-28_myxmlfile.results** naming convention. If the complete file name is specified, for example, `mimport myxmlfile.xml -LOG C:\MyDev\ImportResults\myxmlimport_results.txt`, the **myxmlimport_results.txt** will be produced in the **C:\MyDev\ImportResults** directory.

If the `mimport` utility command is unable to create the `mimport` log file, the importing of the map source file will terminate and the **Fatal error - Could not start importing the map source file, unable to create the log file** fatal error message will display on the console. If nothing is specified in the command line for the log, by default, the command line assumes the (**-LOG**) option is included in the command line, and writes the log messages to the console.

-FAIL mimport option

The **-FAIL** option can be used with the `mimport` utility command.

The fail (**-FAIL**) option of the `mimport` utility command is used to indicate that only the maps that had errors during the importing should be included in the `mimport` log file. If the log failures only (**-FAIL**) option is selected, the `mimport` log will contain only the maps that had failed importing. If the log failures only (**-FAIL**) option is not selected in the command line, all the imported results will be written to the `mimport` log file.

-APPEND mimport option

The **-APPEND** option can be used with the `mimport` utility command.

The append (**-APPEND**) option of the `mimport` utility command is used to indicate that the current `mimport` log messages should be appended to the existing `mimport` log file if it already exists. If the `mimport` log append (**-APPEND**) option is selected, the `mimport` log messages will be appended to the existing file if it already exists; otherwise, a new file will be created. If the (**-APPEND**) option is not selected, the `mimport` log file will be created if no log file exists, or it will be overwritten if the log file does exist.

-VERBOSE mimport option

The **-VERBOSE** option can be used with the `mimport` utility command.

The verbose (**-VERBOSE**) option of the `mimport` utility command is used to indicate if the verbose `mimport` log should be produced. The verbose `mimport` log will contain an entry for the xml file being imported, the importing start date/time, the importing end date/time, the imported map source file name, and the result of the `mimport` utility command. If the verbose (**-VERBOSE**) option is not selected, a concise version of the `mimport` log will be produced, which includes an entry for the map source file name, and importing result.

-NOLOG mimport option

The **-NOLOG** option can be used with the `mimport` utility command.

The disable log (**-NOLOG**) option of the `mimport` utility command is used to turn off the `mimport` logging capability. If the disable logging option is selected, log file will not be produced. The (**-APPEND**, and **-VERBOSE**) optional commands of the (**-LOG**) option cannot be used with the disable log (**-NOLOG**) option. If these options are selected with the (**-NOLOG**) option, the command line is invalid, and the **Not a valid command line - (LOG) optional commands (-APPEND), (-VERBOSE) cannot be used with disable log command (-NOLOG)** . message will display on the console.

mimport command line help

You can access command line help for the `mimport` utility command.

There is help that describes the `mimport` utility command usage in the command line.

To view the command line help for the `mimport` utility command:

Enter `mimport` at the command line prompt in the `install_dir`.
The syntax and descriptions of all the options appear on the console.

mimport command usage

You can use the `mimport` command to import an XML file to the map source file from the command line.

Use the `mimport` utility command to import the xml file to the map source file from the command line, outside the Map Designer GUI.

The following example shows how you can use the command:

```
install_dir> mimport myxmlfile.xml  
-O C:\MyDev\ImportResults\myimportedmaps.mms  
-LOG -APPEND
```

When you run the example, the following result occurs:

- If the `myxmlfile.xml` xml file is imported successfully, the `mimport - completed successfully.` message displays on the console.
- The imported `myimportedmaps.mms` map source file is produced in the `C:\MyDev\ImportResults` directory.
- If the `myxmlfile.log` log file does not exist, it is produced in the same directory as the xml file. If the `myxmlfile.log` log file already exists, the log messages are appended to the existing file.

mexport utility command

The `mexport` utility command is used to export a map source file from the command line, outside the Map Designer.

The `mexport` utility command returns 0 if the map export is successful and 1 if it is not successful. It is a batch operation and can also be used for automation.

See "[Troubleshooting](#)" for details about capturing and evaluating the return code resulting from running this utility command.

- [mexport syntax summary](#)
This is the syntax that is used with the `mexport` utility command.
- [mexport utility command options](#)
These are the command options that can be used with the `mexport` utility command.
- [mexport command line help](#)
You can access command line help for the `mexport` utility command.
- [mexport command usage](#)
You can use the `mexport` command to export the map source file from the command line.

mexport syntax summary

This is the syntax that is used with the `mexport` utility command.

`mexport` is the name of the utility command used to export map source files.

The `.mms` file is a required field and is the file name of the map source file that needs to be exported. The map source file can have any number of maps in it. All maps will be exported to the same xml file if provided. If the full path of the map source file name is not specified, the `mexport` utility command will search for the map source file in the current directory.

The following is the syntax of the `mexport` utility command:

```
mexport <.mms file> [-NO]  
[-O <.xml file name/location>]  
[(-LOG [.| <log file name/location>]  
[-FAIL] [-APPEND] [-VERBOSE])  
|-NOLOG]
```

None of the options in the `mexport` utility command are case sensitive. Typing `mexport` with no options will display the command summary on the console.

mexport utility command options

These are the command options that can be used with the `mexport` utility command.

The following command options are available with the `mexport` utility command:

• -O mexport Option	• -APPEND mexport Option
• -NO mexport Option	• -VERBOSE mexport Option
• -LOG mexport Option	• -NOLOG mexport Option
• -FAIL mexport Option	

If you type `mexport` with no options, the command line help that describes the command will display on your screen.

- [-O mexport option](#)
The -O option can be used with the `mexport` utility command.
- [-NO mexport option](#)
The -NO option can be used with the `mexport` utility command.
- [-LOG mexport option](#)
The -LOG option can be used with the `mexport` utility command.

- **-FAIL mexport option**
The -FAIL option can be used with the mexport utility command.
 - **-APPEND mexport option**
The -APPEND option can be used with the mexport utility command.
 - **-VERBOSE mexport option**
The -VERBOSE option can be used with the mexport utility command.
 - **-NOLOG mexport option**
The -NOLOG option can be used with the mexport utility command.
-

-O mexport option

The -O option can be used with the mexport utility command.

The output (-o) option of the `mexport` utility command is used to specify, through a parameter, the location or file name, or both, for the exported xml file produced by exporting the map source file. The specified parameter could be the name of the directory where the exported xml file is written, the full file path specification in which a wildcard can be used to represent the exported xml file, or the full path in which the complete filename can be specified. The default naming convention for the exported xml file is `map_source_file_name.xml`. If a wildcard is specified in the full path, for example, `mexport mymaps.mms -O C:\MyDev\ExportResults\2002-02-28_*.xml`, the resulting xml file is produced in the **C:\MyDev\ExportResults** directory with the **2002-02-28_mymaps.xml** naming convention. If the complete file name is specified, for example, `mexport mymaps.mms -O C:\MyDev\ExportResults\MyMapsExportedfile.xml`, an xml file **MyMapsExportedfile.xml** is produced in the **C:\MyDev\ExportResults** directory. If the export output (-o) option is selected, and if the file name is not specified, the command line is invalid, and the **Not a valid command line - the file name parameter is required if export command option (-O) is selected** error message displays on the console.

-NO mexport option

The -NO option can be used with the mexport utility command.

The no output (-NO) option of the `mexport` utility command is used to indicate that an existing exported xml file (`.xml`) is not to be overwritten. If the no overwrite (-NO) option is specified in the command line of `mexport`, if the file with the name of the map source file resulting from the exporting process exists, it will not be overwritten, the map exporting process will fail, and the **Map Export error - The exported xml file cannot be overwritten, file already exists** message will be written to the `mexport` log file, if enabled. If the (-NO) option is not specified, the log file with the name of the map resulting from the exporting process will be overwritten.

-LOG mexport option

The -LOG option can be used with the mexport utility command.

The log (-LOG) option of the `mexport` utility command is used to enable the `mexport` logging capability. The log file will be produced showing the result of the exported map source file. (-LOG) has an optional parameter to specify the location or file name, or both, for the `mexport` log file. The default file name for the `mexport` log will be `map_source_file_name.log`. The default location for the `mexport` log will be the directory where the map source file is located.

The file name, or the location parameter, if specified with the (-LOG) option of the `mexport` utility command, could be the name of the directory where the `mexport` log file will be written, the full file path specification in which a wildcard can be used to represent the map source file, or the full path in which the complete filename can be specified. If the . is specified with the (-LOG) option, the `mexport` log will be written to the default location (directory where the map source file is located) using the default naming convention. If a wildcard is specified in the full path, for example, `mexport mymaps.mms -LOG C:\MyDev\ExportResults\2002-02-28_*.results` the `mexport` log file will be produced in the **C:\MyDev\ExportResults** directory with the **2002-02-28_mymaps.results** naming convention. If the complete file name is specified, for example, `mexport mymaps.mms -LOG C:\MyDev\ExportResults\mymaps_results.txt`, the **mymaps_results.txt** file will be produced in the **C:\MyDev\ExportResults** directory.

If the `mexport` utility command is unable to create the `mexport` log file, the exporting of the map source file will terminate and the **Fatal error - Could not start exporting the map source file, unable to create the log file** fatal error message will display on the console. If nothing is specified in the command line for the log, by default, the command line assumes the (-LOG) option is included in the command line, and writes the log messages to the console.

-FAIL mexport option

The -FAIL option can be used with the mexport utility command.

The fail (-FAIL) option of the `mexport` utility command is used to indicate that only the maps that had errors during the exporting should be included in the `mexport` log file. If the log failures only (-FAIL) option is selected, the `mexport` log will contain only the maps that had failed exporting. If the log failures only (-FAIL) option is not selected in the command line, all the exported results will be written to the `mexport` log file.

-APPEND mexport option

The -APPEND option can be used with the mexport utility command.

The append (-APPEND) option of the `mexport` utility command is used to indicate that the current `mexport` log messages should be appended to the existing `mexport` log file if it already exists. If the `mexport` log append (-APPEND) option is selected, the `mexport` log messages will be appended to the existing file if it already exists; otherwise, a new file will be created. If the (-APPEND) option is not selected, the `mexport` log file will be created if no log file exists, or it will be overwritten if the log file does exist.

-VERBOSE mexport option

The -VERBOSE option can be used with the mexport utility command.

The verbose (-VERBOSE) option of the `mexport` utility command is used to indicate if the verbose `mexport` log should be produced. The verbose `mexport` log will contain an entry for the map source file being exported, the map source name, the exporting start date/time, the exporting end date/time, the exported xml file name, and the result of the `mexport` utility command. If the verbose (-VERBOSE) option is not selected, a concise version of the `mexport` log will be produced, which includes entry for the map source file name and exporting result.

-NOLOG mexport option

The -NOLOG option can be used with the mexport utility command.

The disable log (-NOLOG) option of the `mexport` utility command is used to turn off the `mexport` logging capability. If the disable logging option is selected, the log file will not be produced. The optional (-APPEND, and -VERBOSE) options of the (-LOG) option cannot be used with the disable log (-NOLOG) option. If these options are selected with the (-NOLOG) option, the command line is invalid, and the Not a valid command line - (LOG) optional commands (-APPEND), (-VERBOSE) cannot be used with disable log command (-NOLOG). error message will display on the console.

mexport command line help

You can access command line help for the mexport utility command.

There is help that describes the `mexport` utility command usage in the command line.

To view the command line help for the mexport utility command:

Enter `mexport` at the command line prompt in the `install_dir`.

The syntax and descriptions of all the options appear on the console.

mexport command usage

You can use the `mexport` command to export the map source file from the command line.

Use the `mexport` utility command to export the map source file from the command line, outside the Map Designer GUI.

The following example shows how you can use the command:

```
install_dir> mexport mymaps.mms  
-O C:\MyDev\ExportResults\mymaps.xml  
-LOG -APPEND
```

When you run the example, the following result occurs:

- If the `mymaps.mms` map source file is exported successfully, the `mexport - completed successfully.` message displays on the console.
- The exported `mymaps.xml` xml file is produced in the `C:\MyDev\ExportResults` directory.
- If the `mymaps.log` log file does not exist, it is produced in the same directory as the map source file. If it already exists, the log messages are appended to the existing file.

mreport utility command

The `mreport` utility command is used to generate a map report HTML file containing information about a map source file, from the command line, outside the Map Designer.

The `mreport` utility command returns 0 if the map report generation is successful and 1 if it is not successful. It is a batch operation and can also be used for automation.

See "[Troubleshooting](#)" for details about capturing and evaluating the return code resulting from the execution of this utility command.

- [mreport syntax summary](#)
This is the syntax that is used with the mreport utility command.
- [mreport utility command options](#)
These are the command options that can be used with the mreport utility command.
- [mreport command line help](#)
You can access command line help for the mreport utility command.
- [mreport command usage](#)
You can use the mimport command to generate a map report HTML file containing information about a map source file.

mreport syntax summary

This is the syntax that is used with the mreport utility command.

`mreport` is the name of the utility command used to generate a map report HTML file containing information about a map source file.

The `.mms` file is a required field and the file name of the map source file. If the full path of the map source file name is not specified, the `mreport` utility command will search for the map source file in the current directory.

The following is the syntax of the `mreport` utility command:

```
mreport <.mms file>
[-NO]
[-O <.html file name/location>]
[(-LOG [. | <log file name/location>]
[-FAIL] [-VERBOSE] [-APPEND])
|-NOLOG]
```

None of the options in the `mreport` utility command are case sensitive. Typing `mreport` with no options will display the command summary on the console.

mreport utility command options

These are the command options that can be used with the `mreport` utility command.

The following command options are available with the `mreport` utility command:

• -NO mreport Option	• -VERBOSE mreport Option
• --O mreport Option	• -APPEND mreport Option
• -LOG mreport Option	• -NOLOG mreport Option
• -FAIL mreport Option	

• [**-O mreport option**](#)

The `-O` option can be used with the `mreport` utility command.

• [**-NO mreport option**](#)

The `-NO` option can be used with the `mreport` utility command.

• [**-LOG mreport option**](#)

The `-LOG` option can be used with the `mreport` utility command.

• [**-FAIL mreport option**](#)

The `-FAIL` option can be used with the `mreport` utility command.

• [**-VERBOSE mreport option**](#)

The `-VERBOSE` option can be used with the `mreport` utility command.

• [**-APPEND mreport option**](#)

The `-APPEND` option can be used with the `mreport` utility command.

• [**-NOLOG mreport option**](#)

The `-NOLOG` option can be used with the `mreport` utility command.

-O mreport option

The `-O` option can be used with the `mreport` utility command.

The output (`-O`) option of the `mreport` utility command is used to specify the location or file name, or both, for the generated HTML file. The specified parameter could be the name of the directory where the HTML file is written, the full file path specification in which a wildcard can be used to represent the HTML file, or the full path in which the complete filename can be specified. The default naming convention for the HTML file is `html_file_name.html`. If a wildcard is specified in the full path, for example, `mreport mymmsfile.mms -O C:\MyDev\MapReportResults\2002-02-28_*.html`, the resulting HTML file is produced in the **C:\MyDev\MapReportResults** directory with the **2002-02-28_mymmsfile.html** naming convention. If the complete file name is specified, as in `mreport mymmsfile.mms -O C:\MyDev\MapReportResults\Myhtmlfile.html`, the **Myhtmlfile.html** HTML file is produced in the **C:\MyDev\MapReportResults** directory. If the `mreport` output (`-O`) option is selected and if the file name is not specified, the command line is invalid, and the `Not a valid command line - the file name parameter is required if mreport command option (-O) is selected` error message displays on the console.

-NO mreport option

The `-NO` option can be used with the `mreport` utility command.

The no overwrite (`-NO`) option of the `mreport` utility command is used to indicate that an existing generated HTML file (`.html`) is not to be overwritten. If the no overwrite (`-NO`) option is specified in the command line of `mreport`, and if the file with the name of the HTML file resulting from the map report generation process exists, it will not be overwritten, the map report generation will fail, and the `Map Report error : The generated html file cannot be overwritten, file already exists : path\file.html` message will be written to the `mreport` log file if enabled. If the (`-NO`) option is not specified, the file with the name of the HTML file resulting from the map report generation process will be overwritten.

-LOG mreport option

The **-LOG** option can be used with the mreport utility command.

The log (**-LOG**) option of the `mreport` utility command is used to enable logging. The log file will be produced showing the result of the generated map report HTML file. (-**LOG**) has an optional parameter to specify the location or file name, or both, for the `mreport` log file. The default file name for the `mreport` log will be **map_source_file_name.log**. The default location for the `mreport` log will be the directory where the map source file is located.

The file name or the location parameter, if specified with the (**-LOG**) option of the `mreport` utility command, could be the name of the directory where the `mreport` log file will be written, the full file path specification in which a wildcard can be used to represent the HTML file, or the full path in which the complete filename can be specified. If the `.` is specified with the (**-LOG**) option, the `mreport` log will be written to the default location, which is the directory where the map source file is located, using the default naming convention. If a wildcard is specified in the full path, for example, `mreport mymmsfile.mms -LOG C:\MyDev\MapReportResults\2002-02-28_*.results`, the `mreport` log file will be produced in the **C:\MyDev\MapReportResults** directory with the **2002-02-28_mymmsfile.results** naming convention. If the complete file name is specified, for example, `mreport mymmsfile.mms -LOG C:\MyDev\MapReportResults\myhtmlmapreport_results.txt`, the **myhtmlmapreport_results.txt** file will be produced in the **C:\MyDev\MapReportResults** directory.

If the `mreport` utility command is unable to create the `mreport` log file, the map report generation of the map source file will terminate and the `Fatal error : Could not start the process, unable to create the log file` fatal error message will display on the console. There might be additional information in the error message depending on the cause of the fatal error. If nothing is specified in the command line for the log, by default, the command line assumes the (**-LOG**) option is included in the command line, and writes the log messages to the console.

-FAIL mreport option

The **-FAIL** option can be used with the mreport utility command.

The fail (**-FAIL**) option of the `mreport` utility command is used to indicate that only the maps that had errors during the report generation should be included in the `mreport` log file. If the log failures only (**-FAIL**) option is selected, the `mreport` log will contain only the maps that had failed the HTML file generation. If the log failures only (**-FAIL**) option is not selected in the command line, all the map report generation results will be written to the `mreport` log file.

-VERBOSE mreport option

The **-VERBOSE** option can be used with the mreport utility command.

The verbose (**-VERBOSE**) option of the `mreport` utility command is used to indicate if the verbose `mreport` log should be produced. The verbose `mreport` log will contain an entry for the HTML file being generated, the map report generation start date/time, the generation end date/time, the generated HTML file name, and the result of the `mreport` utility command. If the verbose (**-VERBOSE**) option is not selected, a concise version of the `mreport` log will be produced, which includes an entry for the HTML file name, and map report generation result.

-APPEND mreport option

The **-APPEND** option can be used with the mreport utility command.

The append (**-APPEND**) option of the `mreport` utility command is used to indicate that the current `mreport` log messages should be appended to the existing `mreport` log file if it already exists. If the `mreport` log append (**-APPEND**) option is selected, the `mreport` log messages will be appended to the existing file if it already exists; otherwise, a new file will be created. If the (**-APPEND**) option is not selected, the `mreport` log file will be created if no log file exists, or it will be overwritten if the log file does exist.

-NOLOG mreport option

The **-NOLOG** option can be used with the mreport utility command.

The disable log (**-NOLOG**) option of the `mreport` utility command is used to turn off the `mreport` logging capability. If the disable logging option is selected, the log file will not be produced. The (**-APPEND**, **FAIL**, and **-VERBOSE**) optional commands of the (**-LOG**) option cannot be used with the disable log (**-NOLOG**) option. If these options are selected with the (**-NOLOG**) option, the command line is invalid, and the `Not a valid command line : (-LOG) optional commands (-APPEND), (-FAIL), (-VERBOSE)` cannot be used with disable log command (**-NOLOG**). message will display on the console.

mreport command line help

You can access command line help for the mreport utility command.

There is help that describes the `mreport` utility command usage in the command line.

To view the command line help for the mreport utility command:

Enter `mreport` at the command line prompt in the `install_dir`.

The syntax and descriptions of all the options appear on the console.

mreport command usage

You can use the `mimport` command to generate a map report HTML file containing information about a map source file.

Use the `mreport` utility command to generate a map report HTML file containing information about a map source file, outside the Map Designer.

The following example shows how you can use the command:

```
install_dir> mreport mymmsfile.mms  
-O C:\MyDev\MapReportResults\myhtmlfile.html  
-LOG -APPEND
```

When you run the example, the following result occurs:

- If the `myhtmlfile.html` HTML file is generated successfully, the `Map report generated successfully.` message displays on the console.
- The generated `myhtmlfile.html` HTML file is produced in the `C:\MyDev\MapReportResults` directory.
- If the `myhtmlfile.log` log file does not exist, it is produced in the same directory as the map source file. If the `myhtmlfile.log` log file already exists, the log messages are appended to the existing file.

Utility commands for Integration Flow Designer

The Integration Flow Designer utility commands are used to deploy systems and import to and export systems.

- [sdeploy utility command](#)

The `sdeploy` utility command is used to analyze systems, build maps in the system, generate and deploy the system to the specified server from the command line, outside of the Integration Flow Designer.

- [ssftpdeploy utility command](#)

The `ssftpdeploy` command uses the SSH File Transfer Protocol (SFTP) to deploy an IBM Transformation Extender Launcher system to a remote host. The remote host must have an SSH server configured to accept remote SFTP requests. The computer that runs the `ssftpdeploy` command must be configured for SSH communication with the remote host. To establish SSH communication when SSH host key verification is in effect, the SSH public keys of the remote host must be specified in the `%userprofile%\.ssh\known_hosts` file on the host that deploys the system.

- [msdimport utility command](#)

The `msdimport` utility command is used to import an XML file to a system definition (`.msd`) source file in a system from the command line, outside of the Integration Flow Designer.

- [msdexport utility command](#)

The `msdexport` utility command is used to export a system source definition (`.msd`) file from the command line, outside of the Integration Flow Designer.

sdeploy utility command

The `sdeploy` utility command is used to analyze systems, build maps in the system, generate and deploy the system to the specified server from the command line, outside of the Integration Flow Designer.

The `sdeploy` utility command returns 0 if the system deployment is successful, and 1 if not. It is a batch operation that can also be used for automation.

See "[Troubleshooting](#)" for details about capturing and evaluating the return code resulting from running this utility command.

- [sdeploy syntax summary](#)

The `sdeploy` utility command is used to analyze systems, build maps in the system, and generate and deploy systems.

- [sdeploy utility command options](#)

These are the command options that can be used with the `sdeploy` utility command.

- [sdeploy command line help](#)

You can access command line help for the `sdeploy` utility command.

- [sdeploy command usage](#)

You can use the `sdeploy` command to analyze systems, build maps in the system, generate and deploy the system to the specified server from the command line, outside of the Integration Flow Designer.

sdeploy syntax summary

The `sdeploy` utility command is used to analyze systems, build maps in the system, and generate and deploy systems.

The `.msd` file is a required field and is the name of the system definition file, located on disk, that needs to be analyzed, compiled, generated, or deployed to the specified server. The system definition file can have any number of systems in it.

If you make modifications, such as changing the number or order of input or output cards in your maps, you must open the `.msd` file in the Integration Flow Designer and save the changes in the file there to synchronize the changes before you can deploy the system using the `sdeploy` utility command.

If the full path of the `.msd` file name is not specified, the `sdeploy` command will search for the `.msd` file in the current directory. Because the `sdeploy` command expects the map source definition (`.msd`) file to be specified after it, if the `.msd` file is not specified, the following error message will display on the console:

```
"Invalid or empty map source file. Cannot open file path\command_parameters"
```

The following is the syntax of the `sdeploy` utility command:

```
sdeploy <.msd file>  
-L  
- or -
```

```

sdeploy <.msd file>
  -A [-K] [-NO]
    <ANALYZE OPTIONS | BUILDMAPS OPTIONS
    | GENERATE OPTIONS>
  [(-LOG [. | <log file name/location>]
    [-FAIL] [-VERBOSE] [-APPEND])
  | -NOLOG]
  - or -
sdeploy <.msd file> <system> [-NO]
  <ANALYZE OPTIONS | BUILDMAPS OPTIONS
  | GENERATE OPTIONS | DEPLOY OPTIONS>
  [(-LOG [. | <log file name/location>]
    [-FAIL] [-VERBOSE] [-APPEND])
  | -NOLOG]

```

- sdeploy analyze syntax option**

The ANALYZE option can be used with the sdeploy utility command.

- sdeploy buildmaps syntax options**

The BUILDMAPS options can be used with the sdeploy utility command.

- sdeploy generate syntax option**

The GENERATE option can be used with the sdeploy utility command.

- sdeploy deploy syntax options**

The DEPLOY option can be used with the sdeploy utility command.

sdeploy analyze syntax option

The ANALYZE option can be used with the sdeploy utility command.

```
-ANALYZE [-EMODE {ES | CS}] [-R <MSEAnlz.txt file name/location>]
```

sdeploy buildmaps syntax options

The BUILDMAPS options can be used with the sdeploy utility command.

```
-BUILDMAPS [-P <platform>]
  [-O <file location>]
  | -A [-R <MSEBldz.txt file name/location>]
```

sdeploy generate syntax option

The GENERATE option can be used with the sdeploy utility command.

```
-GENERATE [-EMODE {ES | CS}] [-O <file name/location>]
```

sdeploy deploy syntax options

The DEPLOY option can be used with the sdeploy utility command.

```
-DEPLOY <script name>
  [-EMODE {ES | CS}]
  [-MRC <mrc file>]
  [<-SERVER <server name> | LOCAL>
    <-USER <userid>
    -PW <password>
    -IP <IP address>>]
  [-R <MSEDploy.txt file name/location>]
```

In the sdeploy utility command, the system name, script name, server name, if provided, userid and password options, are case sensitive.

Typing sdeploy with no options will display the command summary on the console.

sdeploy utility command options

These are the command options that can be used with the sdeploy utility command.

The following command options are available with the sdeploy utility command.

If you type sdeploy with no options, the command line help that describes the command will display on your screen.

- **-L sdeploy option**
The -L option can be used with the sdeploy utility command.
- **-A sdeploy option**
The -A option can be used with the sdeploy utility command.
- **-K sdeploy option**
The -K option can be used with the sdeploy utility command.
- **-EMODE sdeploy option**
The -EMODE option can be used with the sdeploy utility command.
- **-ANALYZE sdeploy option**
The -ANALYZE option can be used with the sdeploy utility command.
- **-BUILDMAPS sdeploy option**
The -BUILDMAPS option can be used with the sdeploy utility command.
- **-GENERATE sdeploy option**
The -GENERATE option can be used with the sdeploy utility command.
- **-DEPLOY sdeploy option**
The -DEPLOY option can be used with the sdeploy utility command.
- **-MRC sdeploy option**
The -MRC option can be used with the sdeploy utility command. The **-MRC** option specifies the path and file name of a resource configuration (.mrc) file. At system deployment time, the **sdeploy** command uses this resource configuration file to resolve aliases in the MapServerLocation, server user ID (-USER), and server password (-PW) settings.
- **-SERVER sdeploy option**
The -SERVER option can be used with the sdeploy utility command.
- **-USER -PW -IP -sdeploy options**
The -USER, -PW, and -IP options can be used with the sdeploy utility command.
- **-P sdeploy option**
The -P option can be used with the sdeploy utility command.
- **-O sdeploy option**
The -O option can be used with the sdeploy utility command.
- **-NO sdeploy option**
The -NO option can be used with the sdeploy utility command.
- **-R sdeploy option**
The -R option can be used with the sdeploy utility command.
- **-LOG sdeploy option**
The -LOG option can be used with the sdeploy utility command.
- **-FAIL sdeploy option**
The -FAIL option can be used with the sdeploy utility command.
- **-APPEND sdeploy option**
The -APPEND option can be used with the sdeploy utility command.
- **-VERBOSE sdeploy option**
The -VERBOSE option can be used with the sdeploy utility command.
- **-NOLOG sdeploy option**
The -NOLOG option can be used with the sdeploy utility command.

-L sdeploy option

The -L option can be used with the sdeploy utility command.

The list (-L) option of the **sdeploy** utility command is used to list all systems that are present in the specified msd file. When this option is selected, the list of all systems will display in alphabetical order, each on a separate line. No other option can be selected with the (-L) option in the **sdeploy** utility command. If any other option of the **sdeploy** utility command is selected with the (-L) option, the command line is invalid, and the **Not a valid command line - No other option is allowed to with '-L'** option error message will display on the console.

-A sdeploy option

The -A option can be used with the sdeploy utility command.

The all (-A) option of the **sdeploy** utility command is used to analyze all systems, compile all maps that are present in all systems and to generate msl files for all systems present in the msd file. The system name should not be specified in the command line if (-A) option is selected. If the system name is also specified with the (-A) option, the command line is invalid, and the **Not a valid command line - System name should not be specified when '-A' process all systems are selected** error message will display on the console.

-o command line option cannot be used with the -A command line option.

-K sdeploy option

The -K option can be used with the sdeploy utility command.

The continue (-K) option of the **sdeploy** utility command is used to indicate whether to continue the specified process for all systems even though an error is found for a system. This option can be used only with the all maps (-A) option. If the continue (-K) option is specified without the (-A) option, the command line is invalid, and the **Not a valid command line : Continue compilation option (-K) can only be used along with compile all option (-A)** error message will display on the console. If the continue compilation (-K) option is not selected, the processing of the systems will not be completed and the **sdeploy** will terminate. It will not

continue with the rest of the systems that are present in the msd file. If the continue compilation (-K) option is selected, the processing will be performed for all the systems that are present in the msd file, regardless of any errors found during the processing for any of the individual systems.

-EMODE sdeploy option

The -EMODE option can be used with the sdeploy utility command.

The execution mode (-EMODE) option of the sdeploy utility command is used to specify the mode in which the operations need to be performed. The available options for (-EMODE) are ES (Launcher mode) and CS (Command Server mode). If the (-EMODE) option is not specified, the ES (Launcher) mode will be the default. If the (-EMODE) option is specified and the (ES | CS) options are not specified, or if both are specified, the command line is invalid, and the Not a valid command line - ES(Launcher) or CS(Command Server) modes has to be specified when (-EMODE) option is selected. error message will display on the console. If the (-EMODE) option is specified with the (-BUILDMAPS) option, the command line is invalid, and the Not a valid command line - (-EMODE) option cannot be specified along with (-BUILDMAPS) command option. error message will display on the console.

-ANALYZE sdeploy option

The -ANALYZE option can be used with the sdeploy utility command.

The analyze (-ANALYZE) option of the sdeploy utility command is used to analyze the systems present in the msd file. If the (-ANALYZE) option is specified with the (-A) option, it processes all systems' options. It is specifying that all the systems must be analyzed, and the results files are generated, one for each system that was analyzed. If the (-ANALYZE) option is specified with a system name only, that system is analyzed.

For ANALYZE OPTIONS, the following information can be supplied:

```
-ANALYZE [-EMODE {ES | CS}]  
[-R <MSLEANlz.txt file name/location>]
```

-BUILDMAPS sdeploy option

The -BUILDMAPS option can be used with the sdeploy utility command.

The build maps (-BUILDMAPS) option of the sdeploy utility command is used to build all the maps that are in the systems present in the msd file. If the (-BUILDMAPS) option is specified with the (-A) option, process all the systems' options. It is specifying that all the maps in all systems need to be compiled with the platform specified in the command line, the output .mmc files will be produced in the specified directory, and the results files will be generated one for each system that has been called. If the (-BUILDMAPS) option is specified with a system name only, all the maps in that system will be compiled.

-O command line option cannot be used with the -A command line option.

BUILDMAPS OPTIONS information

For BUILDMAPS OPTIONS, the following information can be supplied:

```
-BUILDMAPS [-P <platform>]  
           [-O <file location>]  
           | -A [-R  
                 <MSEBlds.txt file name/location>]
```

-GENERATE sdeploy option

The -GENERATE option can be used with the sdeploy utility command.

The -GENERATE option of the sdeploy utility command generates the systems that are present in the system definition (.msd) file.

When you specify the -A option with the -GENERATE option, the command processes all of the systems in the system definition file. When you specify a system name with the -GENERATE option, the command produces the output file for the specified system only. Depending on the specified -EMODE option, the command produces the output Launcher (.msl) file or Command Server file in the specified output (-O) directory.

Launcher (.msl) files are binary files. Command Server files are text files and can have any extension, but avoid using the dedicated .msl file extension with Command Server mode.

GENERATE options

```
-GENERATE      [-EMODE {ES | CS}]      [-O file_path/file_name]
```

-DEPLOY sdeploy option

The -DEPLOY option can be used with the sdeploy utility command.

The deploy (-DEPLOY) option of the sdeploy utility command is used to deploy the deploy scripts to the specified server in the msd file. The (-DEPLOY) option cannot be specified with the (-A) option. It is specifying that all systems' options be processed as the script name is specific to each system, and can only be specified with a system name. The script defined in the msd file for a system can be deployed to the specified server with the (ES | CS) mode specified in the command line. The server can be specified with the (-SERVER) option. The **MSED**ploy_system_name.txt results file will be generated for the system that has been called.

DEPLOY OPTIONS information

For DEPLOY OPTIONS, the following information can be supplied:

```
-DEPLOY <script name>
  [-EMODE {ES | CS}]
  [-MRC <mrc_file>]
  [<-SERVER <server name> | LOCAL]
    <-USER <userid>
    -PW <password>
    -IP <IP address>>]
  [-R <MSEDploy.txt file name/location>]
```

-MRC sdeploy option

The -MRC option can be used with the sdeploy utility command. The -MRC option specifies the path and file name of a resource configuration (.mrc) file. At system deployment time, the sdeploy command uses this resource configuration file to resolve aliases in the MapServerLocation, server user ID (-USER), and server password (-PW) settings.

-SERVER sdeploy option

The -SERVER option can be used with the sdeploy utility command.

The server (-SERVER) option of the sdeploy utility command is used to specify the server name to which the deploy script needs to be deployed. The (-SERVER) option can be specified only with the (-DEPLOY) option. If specified with the (-ANALYZE, -BUILDMAPS, or -GENERATE) or the (-USER, -PW, or -IP) options, the command line is invalid, and the Not a valid command line - -SERVER option cannot be specified with -ANALYZE, -BUILDMAPS, -GENERATE, -USER, -PW, -IP. error message will display on the console. The server name can be LOCAL or any defined server name in the msd file. A new server can be defined by specifying the IP address of the server with the (-IP) option, the username with the (-USER) option, and the password with the (-PW) option.

-USER -PW -IP -sdeploy options

The -USER, -PW, and -IP options can be used with the sdeploy utility command.

The server (-USER, -PW, and -IP) options of the sdeploy utility command are used to specify a new server for deploying the deploy script. All three options are mandatory for specifying a new server. If any one of the three options are not specified, the command line is invalid, and the Not a valid command line - all three (-USER), (-PW), (-IP) need to be specified to for a new server. error message displays on the console. If you are able to create a server, the deploy script is deployed to this new server, or the Unable to create a server with the user options and IP given error message is written to the log file, if enabled, and the operation is terminated.

For IP address, you can specify one that is in IPv4 (for example, 1.2.3.4) or IPv6 (for example, a:b:c:d:0:1:2:3) format.

-P sdeploy option

The -P option can be used with the sdeploy utility command.

The parameter (-P) option of the sdeploy utility command is used to indicate a parameter to specify the platform for which the maps are to be compiled, the platform for which the systems must be generated, and to which the systems must be deployed. The sdeploy utility command line uses WINDOWS as the default platform if the (-P) option is not specified. The other platforms that are allowed are AIX, HPITANIUM, LINUX, PLINUX, SOLARIS, ZLINUX, and ZOS. If the (-P) option is specified and no platform parameter is provided, the command line is invalid, and the Not a valid command line : Platform parameter should be provided if option -P is selected. The valid platforms are AIX, SOLARIS, WINDOWS, ZOS, LINUX, HPITANIUM, ZLINUX, PLINUX. error message displays on the console.

-O sdeploy option

The -O option can be used with the sdeploy utility command.

The output (-o) option of the sdeploy utility command is used to specify, through a parameter, the location or file name, or both, for the resulting compiled map files when (-BUILDMAPS) option is chosen, or resulting generated msl file when (-GENERATE) option is chosen. This is a description of how the output (-o) option can be used in the sdeploy utility command.

The specified parameter for the output (-o) option could be the name of the directory where the resulting compiled map files or the resulting generated msl file is written, the full file path specification in which a wildcard can be used to represent the compiled map file or the generated msl file, or the full path in which the complete filename can be specified only for the msl file. The default naming convention for the compiled map file is **executable_map_name.mmc**, and for the generated msl file is **system_name.msl**. If a wildcard is specified in the sdeploy mysystems.msd MySystem -BUILDMAPS -O C:\MyDev\Compiledmaps*.win full path, the compiled

maps are produced in the **C:\MyDev\Compiledmaps** directory with the **<executable_map_name>.win** naming convention. If a wildcard is specified in the **sdeploy mysystems.msd MySystem -GENERATE -O C:\MyDev\Systemfiles*.*.win** full path, the generated msl is produced in the **C:\MyDev\Systemfiles** directory with the **<system_name>.win** naming convention. If the complete file name is specified, for example, **sdeploy mysystems.msd MySystem -GENERATE -O C:\MyDev\Systemfiles\MyExecForWin.win**, the **MyExecForWin.win** file is produced in the **C:\MyDev\Systemfiles** directory.

If the **(-O)** option is not specified, the compile map is written to the same directory as the map source file using the default **executable_map_name.mmc** naming convention and the msl file is written to the source directory of the msd file with the **system_name.msl** default naming convention. If the **(-O)** option is specified and the parameter is not provided, the command line is invalid, and the **Not a valid command line : The file name for the resulting output file should be provided when (-O) option is selected** error message displays on the console. If the file name or the path provided for the **(-O)** option is not valid or returns error while attempting to open or write to the file, the **Compilation error - The file/path provided for the option (-O) is not valid, unable to write to the specified directory/file** error message is written to the **sdeploy log file**, if it is enabled. If the **(-O)** option is specified without the **(-BUILDMAPS or -GENERATE) command options**, the command line is invalid, and the **Not a valid command line : (-GENERATE) or (-BUILDMAPS) or (-DEPLOY) or (-L) or (-ANALYZE) is missing**. error message displays on the console.

Note: **-O** command line option cannot be used with the **-A** command line option.

-NO sdeploy option

The **-NO** option can be used with the **sdeploy** utility command.

The no overwrite **(-NO)** option of the **sdeploy** utility command is used to indicate that an existing (**.mmc** or **.msl**) output file is not to be overwritten. If the no overwrite **(-NO)** option is specified in the command line of **sdeploy**, and a file with the same name as the resulting output file exists, it will not be overwritten, the **sdeploy** operation will fail, and the **Compilation error - The output file cannot be overwritten, file already exists** message will be written to the **sdeploy log file**, if enabled. If the **(-NO)** option is not specified, the file with the same name as the resulting output file will be overwritten.

-R sdeploy option

The **-R** option can be used with the **sdeploy** utility command.

The results **(-R)** option of the **sdeploy** utility command will have a parameter to specify the location or file name, or both, for the results file produced by the **sdeploy** operation for each system. There are 2 different cases where **(-R)** can be used in the **sdeploy** utility command.

In the first case, the **(-R)** option is used with the process all systems **(-A)** option. The specified parameter could either be the name of the directory where the results files will be written, or the full file path specification in which a wildcard can be used to represent the system or executable map naming convention. The default naming convention for the results file is **MSEBlds_system_name.txt** for the **(-BUILDMAPS)** option and **MSEAnlz_system_name.txt** for the **(-ANALYZE)** option. If a wildcard is specified in the **sdeploy mysystems.msd -ANALYZE -A -R C:\MyDev\AnalyzeResults*.*.err** full path, the results files will be produced in the **C:\MyDev\AnalyzeResults** directory with the **MSEAnlz_system_name.err** naming convention. If a wildcard is specified in the full path as **sdeploy mysystems.msd -BUILDRSULTS -A -R C:\MyDev\BuildResults*.*.err**, the results files will be produced in the **C:\MyDev\BuildResults** directory with the **MSEBlds_system_name.err** naming convention.

In the second case, the **(-R)** option is used without the process all systems **(-A)** option. The specified parameter could be the name of the directory where the results files will be written, the full file path specification in which a wildcard can be used to represent the result file, or the full path in which the complete filename can be specified. The default naming convention for the results file is **MSEBlds_system_name.txt** for the **(-BUILDMAPS)** option, **MSEAnlz_system_name.txt** for the **(-ANALYZE)**, and **MSEDploy_system_name.txt** for the **(-DEPLOY)** option. If a wildcard is specified in the full path as **sdeploy mysystems.msd mysystem -ANALYZE -R C:\MyDev\AnalyzeResults*.*.err**, the results files will be produced in the **C:\MyDev\AnalyzeResults** directory with the **MSEAnlz_system_name.err** naming convention. If a wildcard is specified in the full path as **sdeploy mysystems.msd mysystem -BUILDRSULTS -R C:\MyDev\BuildResults*.*.err**, the results files will be produced in the **C:\MyDev\BuildResults** directory with the **MSEBlds_system_name.err** naming convention. If a wildcard is specified in the full path as **sdeploy mysystems.msd mysystem -DEPLOY -R C:\MyDev\DeployResults*.*.err**, the results files will be produced in the **C:\MyDev\DeployResults** directory with the **MSEDploy_system_name.err** naming convention.

If the **(-R)** option is not specified, the results will be written to the console. If the **(-R)** option is specified, but no parameter is provided, the command line is invalid, and the **Not a valid command line - The file name for the resulting result file should be provided when -R option is selected** error message will display on the console. If the file name or the path provided for the **(-R)** option is not valid or returns error while attempting to open or write to the file, the **Compilation error - The file/path provided for the option (-R) is not valid, unable to write to the specified directory/file** message will be written to the **sdeploy log file**, if it is enabled. If the **(-R)** option is specified without the **(-ANALYZE, -BUILDMAPS, or -DEPLOY) options**, the command line is invalid, and the **Not a valid command line - (-R) option can be specified only with (-ANALYZE) or (-BUILDMAPS) or (-DEPLOY) command options**. error message will display on the console.

-LOG sdeploy option

The **-LOG** option can be used with the **sdeploy** utility command.

The log **(-LOG)** option of the **sdeploy** utility command is used to enable the **sdeploy** logging capability. The log file will be produced with the results of each operation performed. The **(-LOG)** option has an optional parameter to specify the location or file name, or both, for the **sdeploy** log file. The scope of the **sdeploy** log depends on which **(-ANALYZE, -BUILDMAPS, -GENERATE, or -DEPLOY)** option has been chosen with the **sdeploy** utility command. The default file name for the **sdeploy** log will be **msd_file_name.scl**. The default location for the **sdeploy** log will be the directory where the **msd** file is located.

The file name or the location parameter, if specified with the **(-LOG)** option of the **sdeploy** utility command, could be the name of the directory where the **sdeploy** log file will be written, the full file path specification in which a wildcard can be used to represent the **msd** file, or the full path in which the complete filename can be specified. The default naming convention for the **sdeploy** log file is **msd_file_name.scl**. If the **.** is specified with the **(-LOG)** option, the **sdeploy** log will be written to the default location (directory where the **msd** file is located) using the default naming convention. If a wildcard is specified in the full path as **sdeploy mysystems.msd -A -LOG C:\MyDev\Systemfiles\2002-02-28_*.results**, the **sdeploy** log file will be produced in the **C:\MyDev\Systemfiles** directory with the **2002-02-28_myystems.results** naming convention. If the complete file name is specified as **sdeploy mysystems.msd -A -LOG C:\MyDev\Systemfiles\mysystems_results.txt**, the **mysystems_results.txt** will be produced in the **C:\MyDev\Systemfiles** directory.

If the `sdeploy` utility command is unable to create the `sdeploy` log file, the any operation of `sdeploy` will terminate and the Fatal error - Could not start processing, unable to create the log file fatal error message will display on the console. If nothing is specified in the command line for the log, by default, the command line assumes the (`-LOG`) option is included in the command line, and writes the log messages to the console.

-FAIL sdeploy option

The `-FAIL` option can be used with the `sdeploy` utility command.

The fail (`-FAIL`) option of the `sdeploy` utility command is used to indicate that only the process that had errors during operation should be included in the `sdeploy` log file. If the log failures only (`-FAIL`) option is selected, the `sdeploy` log will contain only the logs of failed operations. If the log failures only (`-FAIL`) option is not selected in the command line, all the results will be written to the `sdeploy` log file.

-APPEND sdeploy option

The `-APPEND` option can be used with the `sdeploy` utility command.

The append (`-APPEND`) option of the `sdeploy` utility command is used to indicate that the current `sdeploy` log messages should be appended to the existing `sdeploy` log file if it already exists. If the `sdeploy` log append (`-APPEND`) option is selected, the `sdeploy` log messages will be appended to the existing file if it already exists; otherwise, a new file will be created. If the (`-APPEND`) option is not selected, the `sdeploy` log file will be created if no log file exists, or it will be overwritten if the log file does exist.

-VERBOSE sdeploy option

The `-VERBOSE` option can be used with the `sdeploy` utility command.

The verbose (`-VERBOSE`) option of the `sdeploy` utility command is used to indicate that a verbose `sdeploy` log should be produced. The verbose `sdeploy` log will contain an entry for each (`-ANALYZE`, `-BUILDMAPS`, `-GENERATE`, and `-DEPLOY`) operation attempted, the `.msd` file name, operation start date/time, operation end date/time, operation result, and the name of the results file, if it exists. If the verbose (`-VERBOSE`) option is not selected, a concise version of the `sdeploy` log will be produced, which includes an entry for each `msd` file, attempted system name, and operation result.

-NOLOG sdeploy option

The `-NOLOG` option can be used with the `sdeploy` utility command.

The disable log (`-NOLOG`) option of the `sdeploy` utility command is used to turn off the `sdeploy` logging capability. If the disable logging option is selected, the log file will not be produced. The (`-APPEND`, `-FAIL`, and `-VERBOSE`) options of the (`-LOG`) option cannot be used with the disable log (`-NOLOG`) option. If these options are selected with (`-NOLOG`), the command line is invalid, and the Not a valid command line - (`LOG`) optional commands (`-APPEND`), (`-FAIL`), (`-VERBOSE`) cannot be used with disable log command (`-NOLOG`). error message will display on the console.

sdeploy command line help

You can access command line help for the `sdeploy` utility command.

There is help that describes the `sdeploy` utility command usage in the command line.

To view the command line help for the `sdeploy` utility command:

Enter `sdeploy` at the command line prompt in the `install_dir`.

The syntax and descriptions of all the options appear on the console.

sdeploy command usage

You can use the `sdeploy` command to analyze systems, build maps in the system, generate and deploy the system to the specified server from the command line, outside of the Integration Flow Designer.

The following examples show how you can use the `sdeploy` utility command from the command line to:

- display all systems on the console
 - analyze all systems
 - build maps in all systems
 - generate all systems
 - deploy scripts to a specified server
- [Using sdeploy to display all systems on the console](#)
 - [Using sdeploy to analyze all systems](#)
 - [Using sdeploy to build maps from the command line - example 1](#)
 - [Using sdeploy to generate systems from the command line - example 2](#)

- [Using sdeploy to deploy scripts from the command line - example 3](#)
- [Using sdeploy to deploy scripts from the command line - example 4](#)

Using sdeploy to display all systems on the console

Use the `sdeploy` utility command to display all systems on the console from the command line, outside the Integration Flow Designer GUI.

The following example shows how you can use the `(-L)` option on the `sdeploy` utility command to display all the systems on the console:

```
install_dir> sdeploy mysystems.msd -L
```

When you run the example, the list of all systems displays in alphabetical order and the total number of systems display on the console:

```
Systems in mysystems.msd:  
  FristSystem  
  MySystem  
  NewSystem  
Total systems: 3
```

Using sdeploy to analyze all systems

Use the `sdeploy` utility command to analyze all systems from the command line, outside the Integration Flow Designer GUI.

The following example shows how you can use the `(-A)` option on the `sdeploy` utility command to analyze all the systems:

```
install_dir> sdeploy mysystems.msd  
-ANALYZE -A -EMODE ES  
-R C:\MyDev\AnalyzeResults\*.txt  
-LOG C:\MyDev\AnalyzeResults\*.results.txt  
-FAIL -APPEND
```

When you run the example, the following result occurs:

- If all the systems are analyzed successfully, the `sdeploy - completed successfully.` message displays on the console.
- The `MSEAnlz_FristSystem.txt`, `MSEAnlz_MySystem.txt`, `MSEAnlz_NewSystem.txt` analyzed results files are produced for launcher mode in the `C:\MyDev\AnalyzeResults` directory.
- If the `mysystems_results.txt` analyzed results file does not exist, the `sdeploy`'s log file is produced in the `C:\MyDev\AnalyzeResults` directory. If the `mysystems_results.txt` analyzed results file already exists, only the failed system's log messages are appended to the existing file.

Using sdeploy to build maps from the command line - example 1

Use the `sdeploy` utility command to build maps in all systems from the command line, outside the Integration Flow Designer GUI.

The following example shows how you can use the `(-A)` option on the `sdeploy` utility command to build all the maps in all systems:

```
install_dir> sdeploy mysystems.msd  
-BUILDMAPS -A -P WINDOWS  
-R C:\MyDev\BuildResults\*.txt  
-LOG C:\MyDev\BuildResults\*.results.txt  
-FAIL -APPEND
```

When you run the example, the following result occurs:

- If all the maps in the systems are compiled successfully, the `sdeploy - completed successfully.` message displays on the console.
- The `MSEBld_FristSystem.txt`, `MSEBld_MySystem.txt`, `MSEBld_NewSystem.txt` build results files and the compiled `executable_map_name.mmc` map are produced in the current directory.

If the `mysystems_results.txt` file does not exist, the `sdeploy`'s log file is produced in the `C:\MyDev\BuildResults` directory. If the `mysystems_results.txt` already exists, only the failed systems log messages are appended to the existing file.

Using sdeploy to generate systems from the command line - example 2

Use the `sdeploy` utility command to generate all systems from the command line, outside the Integration Flow Designer GUI.

The following example shows how you can use the `(-A)` option on the `sdeploy` utility command to generate all the systems:

```
install_dir> sdeploy mysystems.msd  
-GENERATE -A -EMODE ES -P WINDOWS  
-LOG C:\MyDev\GenerateResults\*.results.txt  
-FAIL -APPEND
```

When you run the example, the following result occurs:

- If all the systems are generated successfully, the `sdeploy - completed successfully.` message displays on the console.
- The `FristSystem.msl`, `MySystem.msl`, `NewSystem.msl` generated output files are produced in the current directory.
- If the `mysystems_results.txt` file does not exist, the `sdeploy`'s log file are produced in the `C:\MyDev\GenerateResults` directory. If the `mysystems_results.txt` file already exists, only the failed systems log messages are appended to the existing file.

Using `sdeploy` to deploy scripts from the command line - example 3

Use the `sdeploy` utility command to deploy scripts from the command line, outside the Integration Flow Designer GUI.

The following example shows how you can use the `sdeploy` utility command to deploy scripts to the specified server:

```
install_dir> sdeploy mysystems.msd NewSystem
-DEPLOY MyScript -EMODE ES -SERVER MyServer
-R C:\MyDev\DeployResults\*.txt
-LOG C:\MyDev\DeployResults\*.results.txt
-FAIL -APPEND
```

When you run the example, the following result occurs:

- If the deploy script is deployed successfully, the `sdeploy - completed successfully.` message displays on the console.
- The `MyScript` map is deployed to the `MyServer` and the results files are produced in the `C:\MyDev\DeployResults` directory.
- If the `mysystems_results.txt` file does not exist, the `sdeploy`'s log file is produced in the `C:\MyDev\DeployResults` directory. If `mysystems_results.txt` file already exists, only the failed systems log messages are appended to the existing file.

Using `sdeploy` to deploy scripts from the command line - example 4

Use the `sdeploy` utility command to deploy scripts from the command line, outside the Integration Flow Designer GUI.

The following example shows how you can use the `sdeploy` utility command to deploy scripts to the new server created with the specified IP address, the userid, and the password:

```
install_dir> sdeploy mysystems.msd NewSystem
-DEPLOY MyScript -EMODE ES
-USER userid -PW password -IP 192.82.65.46
-R C:\MyDev\DeployResults\*.txt
-LOG C:\MyDev\DeployResults\*.results.txt
-FAIL -APPEND
```

When you run the example, the following result occurs:

- If the deploy script is deployed successfully, the `sdeploy - completed successfully.` message displays on the console.
- The `MyScript` map is deployed to the new server created with the specified IP address, the userid, and the password, and the results files are produced in the `C:\MyDev\DeployResults` directory.
- If the `mysystems_results.txt` file does not exist, it is created in the `C:\MyDev\DeployResults` directory. If the file exists, append only the failed systems log messages.

You can specify an IP address that is in IPv4 (for example, `1.2.3.4`) or IPv6 (for example, `a:b:c:d:0:1:2:3`) format.

ssftpdeploy utility command

The `ssftpdeploy` command uses the SSH File Transfer Protocol (SFTP) to deploy an IBM Transformation Extender Launcher system to a remote host. The remote host must have an SSH server configured to accept remote SFTP requests. The computer that runs the `ssftpdeploy` command must be configured for SSH communication with the remote host. To establish SSH communication when SSH host key verification is in effect, the SSH public keys of the remote host must be specified in the `%userprofile%\.ssh\known_hosts` file on the host that deploys the system.

```
ssftpdeploy [deployment_options] sys_def.msd system deployment_script
```

Required parameters

`sys_def.msd`

The name and path of the system definition file, relative to the current directory. This parameter is required and positional.

`system`

The name of the system. This parameter is required and positional.

`deployment_script`

The name of the deployment script in the system definition (.msd) file. This parameter is required and positional.

Deployment options

The `deployment_options` of the `ssftpdeploy` command are:

```
[-EMODE {ES | CS}]
[-MRC resource_cfg.mrc]
[-SERVER server_name] [-USER user_name] [-PW password] [-IP IP_address]
```

```
[-R [results_file_location] [results_file_name]]
[-LOG [log_file_location] [log_file_name]]
[-VERIFYHOSTKEY]
[-H]
```

-EMODE mode

The processing mode for running the systems:

ES

Launcher mode. This is the default.

CS

Command server mode.

-MRC resource_cfg.mrc

The name and path of the resource registry configuration file, relative to the current directory. The resource registry configuration file resolves the aliases used in the systems.

-SERVER server_name

The server to which to deploy the system. The server specified on the -SERVER keyword must be defined in the system definition (.msd) file. The -SERVER keyword is optional. If specified, the -SERVER keyword overrides a server specified in the deployment definition script.

-USER user_name

The user name required to log in to the SFTP server to which the system is deployed. The -USER keyword is optional. The -USER keyword overrides the user name of a server specified by the -SERVER keyword or a server specified by the deployment definition script. You can use the -USER keyword together with the -PW and -IP keywords to specify a server that is not defined in the system definition (.msd) file. A server specified by the -USER, -PW and -IP keywords overrides a server specified on the -SERVER keyword or in the deployment definition script.

-PW password

The password required to log in to the SFTP server to which the system is deployed. The -PW keyword is optional. The -PW keyword overrides the password of a server specified by the -SERVER keyword or a server specified by the deployment definition script. You can use the -PW keyword together with the -USER and -IP keywords to specify a server that is not defined in the system definition (.msd) file. A server specified by the -USER, -PW and -IP keywords overrides a server specified on the -SERVER keyword or in the deployment definition script.

-IP IP_address

The IP address required to log in to the SFTP server to which the system is deployed. The -IP keyword is optional. The -IP keyword overrides the IP address of a server specified by the -SERVER keyword or a server specified by the deployment definition script. You can use the -IP keyword together with the -USER and -PW keywords to specify a server that is not defined in the system definition (.msd) file. A server specified by the -USER, -PW and -IP keywords overrides a server specified on the -SERVER keyword or in the deployment definition script.

-R [results_file_location] [results_file_name]

Logs the result of each **ssftpdeploy** operation for each system to a results file. In addition to the status of the deployment, the results log includes map processing details such as number of successful builds. This keyword is optional. If you omit it, the results are written to the default MSEsftpdeploy.txt file in the current directory. If you omit the results file name and location, the -R keyword creates a results file with path where the system definition (.msd) file is located. You can specify the name or location of the results file in the following ways:

Variable	Results file location and name
directory_name	directory_name\MSEsftpdeploy.txt
full_path*.err	full_path\MSEsftpdeploy.err
full_path\results_filename	full_path\results_filename

-LOG [log_file_location] [log_file_name]

Enables logging of all **ssftpdeploy** processes to a log file. This keyword is optional. If you omit it, the log messages are written to the default MSEsftpdeploy.log file in the current directory. You can specify the name or location of the log file in the following ways:

Variable	Log file location and name
directory_name	directory_name\MSEsftpdeploy.log
full_path*.err	full_path\MSEsftpdeploy.err
full_path\log_filename	full_path\log_filename

-VERIFYHOSTKEY

Restricts SSH communication to remote hosts that are defined in the %userprofile%\ssh\known_hosts file on the local server. This keyword is optional. If you omit it, the local server accepts any public key from a remote SSH hosts to establish SSH communication.

-H

Displays the **ssftpdeploy** command help to the console.

Examples

The examples use the following common artifacts:

Server:	tx-bld-cloud1
System definition (.msd) file:	decypher_big.msd
System name:	System1
System deployment definition script:	tx-bld-cloud-def1
Resource registry configuration file:	decypher.mrc

Use the SSH protocol to deploy System1 to the tx-bld-cloud1 server:

```
ssftpdeploy -SERVER tx-bld-cloud1 decypher_big.msd System1 tx-bld-cloud-def1
```

Use the SSH protocol to deploy System1 to the tx-bld-cloud1 server. Override the IP address and password of the server, and use the specified resource registry configuration file to resolve aliases in System1:

```
ssftpdeploy -SERVER tx-bld-cloud1 -MRC decypher.mrc -IP 10.134.55.79 -PW hclpnp123 decypher_big.msd System1 tx-bld-cloud-def1
```

Use the SSH protocol to deploy System1 to the tx-bld-cloud1 server. Override the IP address and password of the server. Use the specified resource registry configuration file to resolve aliases in System1. Verify that the SSH key of the remote host is defined in the %userprofile%\ssh\known_hosts file on the local server:

```
ssftpdeploy -SERVER tx-bld-cloud1 -MRC decypher.mrc -IP 10.134.55.79 -PW hclpnp123 -VERIFYHOSTKEY decypher_big.msd System1 tx-bld-cloud-def1
```

msdimport utility command

The `msdimport` utility command is used to import an xml file to a system definition (`.msd`) source file in a system from the command line, outside of the Integration Flow Designer.

The `msdimport` utility command returns 0 if the import is successful, and 1 if not. It is a batch operation that can also be used for automation.

See "[Troubleshooting](#)" for details about capturing and evaluating the return code resulting from running this command.

- [**msdimport syntax summary**](#)

This is the syntax that is used with the `msdimport` utility command.

- [**msdimport utility command options**](#)

These are the command options that can be used with the `msdimport` utility command.

- [**msdimport command line help**](#)

You can access command line help for the `msdimport` utility command.

- [**msdimport command usage**](#)

You can use the `msdimport` command to import an XML file to the map source file from the command line.

msdimport syntax summary

This is the syntax that is used with the `msdimport` utility command.

`msdimport` is the name of the utility command used to import xml files to system source files.

The `.xml` file is a required field and the file name of the xml source file. If the full path of the xml source file name is not specified, the `msdimport` utility command will search for the xml source file in the current directory.

The following is the syntax of the `msdimport` utility command:

```
msdimport <.xml file>
[-NO]
[-O <.msd file name/location>]
  [-ASVR [<server names>]] [-ASYS [<system names>]]
  [-OSYS] [-OSVR] [-REFSYS] [-REFSVR]
  [-LOG [. | <log file name/location>]
  [-FAIL] [-VERBOSE] [-APPEND]
  |-NOLOG]
```

None of the options in the `msdimport` utility command are case sensitive.

Typing `msdimport` with no options will display the command summary on the console.

msdimport utility command options

These are the command options that can be used with the `msdimport` utility command.

The following command options are available with the `msdimport` utility command:

• -APPEND msdimport Option	• -O msdimport Option
• -ASVR msdimport Option	• -OSVR msdimport Option
• -ASYS msdimport Option	• -OSYS msdimport Option
• -FAIL msdimport Option	• -REFSVR msdimport Option
• -LOG msdimport Option	• -REFSYS msdimport Option
• -NO msdimport Option	• -VERBOSE msdimport Option
• -NOLOG msdimport Option	

- [**-APPEND msdimport option**](#)

The -APPEND option can be used with the `msdimport` utility command.

- [**-ASVR msdimport option**](#)

The -ASVR option can be used with the `msdimport` utility command.

- [**-ASYS msdimport option**](#)

The -ASYS option can be used with the `msdimport` utility command.

- [**-FAIL msdimport option**](#)

The -FAIL option can be used with the `msdimport` utility command.

- [**-LOG msdimport option**](#)

The -LOG option can be used with the `msdimport` utility command.

- [**-NO msdimport option**](#)

The -NO option can be used with the `msdimport` utility command.

- **[-NOLOG msdimport option](#)**
The -NOLOG option can be used with the msdimport utility command.
- **[-O msdimport option](#)**
The -O option can be used with the msdimport utility command.
- **[-OSVR msdimport option](#)**
The -OSVR option can be used with the msdimport utility command.
- **[-OSYS msdimport option](#)**
The -OSYS option can be used with the msdimport utility command.
- **[-REFSVR msdimport option](#)**
The -REFSVR option can be used with the msdimport utility command.
- **[-REFSYS msdimport option](#)**
The -REFSYS option can be used with the msdimport utility command.
- **[-VERBOSE msdimport option](#)**
The -VERBOSE option can be used with the msdimport utility command.

-APPEND msdimport option

The -APPEND option can be used with the msdimport utility command.

The append (-APPEND) option of the `msdimport` utility command is used to indicate that the current `msdimport` log messages should be appended to the existing `msdimport` log file if it already exists. If the `msdimport` log append (-APPEND) option is selected, the `msdimport` log messages will be appended to the existing file if it already exists; otherwise, a new file will be created. If the (-APPEND) option is not selected, the `msdimport` log file will be created if no log file exists, or it will be overwritten if the log file does exist.

-ASVR msdimport option

The -ASVR option can be used with the msdimport utility command.

The server (-ASVR) option of the `msdimport` utility command is used to import only servers from the xml file. If the option is used alone, the `msdimport` utility command will import all the servers from the xml file. If the option is followed by a string of server names separated by commas, for example, -ASVR server1,server2,server3, the `msdimport` utility command will import those specific servers from the xml file.

-ASYS msdimport option

The -ASYS option can be used with the msdimport utility command.

The system (-ASYS) option of the `msdimport` utility command is used to import only systems from the xml file. If the option is used alone, the `msdimport` utility command will import all the systems from the xml file. If the option is followed by a string of system names separated by commas, for example, -ASVR system1,system2,system3, the `msdimport` utility command will import those specific systems from the xml file.

Any combination of the -ASYS and -ASVR options used with the `msdimport` utility command will be executed as separate individual options.

-FAIL msdimport option

The -FAIL option can be used with the msdimport utility command.

The fail (-FAIL) option of the `msdimport` utility command is used to indicate that only the systems that had errors during the importing should be included in the `msdimport` log file. If the log failures only (-FAIL) option is selected, the `msdimport` log will contain only the systems that failed importing. If the log failures only (-FAIL) option is not selected in the command line, all the imported results will be written to the `msdimport` log file.

-LOG msdimport option

The -LOG option can be used with the msdimport utility command.

The log (-LOG) option of the `msdimport` utility command is used to enable logging. The log file will be produced showing the result of the imported system source file. (-LOG) has an optional parameter to specify the location or file name, or both, for the `msdimport` log file. The default file name for the `msdimport` log will be **system_source_file_name_import.log**. The default location for the `msdimport` log will be the directory where the system definition (**.msd**) source file is located.

The file name or the location parameter, if specified with the (-LOG) option of the `msdimport` utility command, could be the name of the directory where the `msdimport` log file will be written, the full file path specification in which a wildcard can be used to represent the xml source file, or the full path in which the complete filename can be specified. If the . is specified with the (-LOG) option, the `msdimport` log will be written to the default location (directory where the xml file is located) using the default naming convention. If a wildcard is specified in the full path, for example, `msdimport myxmlfile.xml -LOG C:\MyDev\ImportResults\2003-10-03_*.results`, the `msdimport` log file will be produced in the **C:\MyDev\ImportResults** directory with the **2003-10-03_myxmlfile.results** naming convention. If the complete file name is specified, for example, `msdimport myxmlfile.xml -LOG C:\MyDev\ImportResults\myxmlimport_results.txt`, the **myxmlimport_results.txt** will be produced in the **C:\MyDev\ImportResults** directory.

If the `msdimport` utility command is unable to create the `msdimport` log file, the importing of the system source file will terminate and the Fatal error - Could not start importing the system source file, unable to create the log file fatal error message will display on the console. If nothing is specified in the

command line for the log, by default, the command line assumes the (-LOG) option is included in the command line, and writes the log messages to the console.

-NO msdimport option

The -NO option can be used with the msdimport utility command.

The no overwrite (-NO) option of the msdimport utility command is used to indicate that an existing imported system definition (.msd) source file is not to be overwritten. If the no overwrite (-NO) option is specified in the command line of msdimport, and a file with the same name as the resulting .msd file exists, it will not be overwritten, the msdimport operation will fail, and the IFD import error - The imported system source file cannot be overwritten, file already exists message will be written to the msdimport log file, if enabled. If the (-NO) option is not specified, the file with the same name as the resulting .msd file, will be overwritten.

-NOLOG msdimport option

The -NOLOG option can be used with the msdimport utility command.

The disable log (-NOLOG) option of the msdimport utility command is used to turn off the msdimport logging capability. If the disable logging option is selected, the log file will not be produced. The (-APPEND, and -VERBOSE) optional commands of the (-LOG) option cannot be used with the disable log (-NOLOG) option. If these options are selected with the (-NOLOG) option, the command line is invalid, and the Not a valid command line - (LOG) optional commands (-APPEND), (-VERBOSE) cannot be used with disable log command (-NOLOG). message will display on the console.

-O msdimport option

The -O option can be used with the msdimport utility command.

The output (-O) option of the msdimport utility command is used to specify, through a parameter, the location or file name, or both, of the system definition (.msd) file produced by importing the xml file. The specified parameter could be the name of the directory to where the imported .msd file is written, or the full file path specification in which a wildcard can be used to represent the imported system source file, or the full path in which the complete file name can be specified. The default naming convention for the imported .msd file is `xml_file_name.msd'. If a wildcard is specified in the full path, for example, MSDImport myxmlfile.xml -O C:\MyDev\ImportResults\2003-10-03_* .msd, the resulting .msd file is produced in the C:\MyDev\ImportResults directory with the 2003-10-03_myxmlfile.msd naming convention. If the complete file name is specified, for example, msdimport myxmlfile.xml -O C:\MyDev\ImportResults\Myxmlsystems.msd, a Myxmlsystems.msd system definition (.msd) file is produced in the C:\MyDev\ImportResults directory.

-OSVR msdimport option

The -OSVR option can be used with the msdimport utility command.

The output (-OSVR) option of the msdimport utility command is used to override servers if the servers are already present in the system definition (.msd) file.

-OSYS msdimport option

The -OSYS option can be used with the msdimport utility command.

The output (-OSYS) option of the msdimport utility command is used to override systems if the systems are already present in the system definition (.msd) file.

-REFSVR msdimport option

The -REFSVR option can be used with the msdimport utility command.

The reference server (-REFSVR) option of the msdimport utility command is used to allow reference servers to be imported.

-REFSYS msdimport option

The -REFSYS option can be used with the msdimport utility command.

The reference system (-REFSYS) option of the msdimport utility command is used to allow reference systems to be imported, also. If the -REFSYS command is specified, the msdimport utility command will also import the dependent systems.

-VERBOSE msdimport option

The -VERBOSE option can be used with the msdimport utility command.

The verbose (-VERBOSE) option of the msdimport utility command is used to indicate if the verbose msdimport log should be produced. The verbose msdimport log will contain an entry for the xml file being imported, the importing start date/time, the importing end date/time, the imported system source file name, and the result of the msdimport utility command. If the verbose (-VERBOSE) option is not selected, a concise version of the msdimport log will be produced, which includes an entry for the system source file name, and importing result.

msdimport command line help

You can access command line help for the msdimport utility command.

There is help that describes the msdimport utility command usage in the command line.

To view the command line help for the msdimport utility command:

Enter msdimport at the command line prompt in the *install_dir*.

The syntax and descriptions of all the options appear on the console.

msdimport command usage

You can use the msdimport command to import an XML file to the map source file from the command line.

Use the msdimport utility command to import the xml file to a system definition (.msd) file from the command line, outside the Integration Flow Designer.

The following example shows how you can use the command:

```
install_dir> msdimport myxmlfile.xml  
-O C:\MyDev\ImportResults\myimportedsystem.msd  
-LOG -APPEND
```

When you run the example, the following result occurs:

- If the **myxmlfile.xml** xml file is imported successfully, the msdimport – completed successfully. message displays on the console.
- The imported **myimportedsystem.msd** system definition file is produced in the **C:\MyDev\ImportResults** directory.

If the **myxmlfile.log** log file does not exist, it is produced in the same directory as the system file. If the **myxmlfile.log** log file already exists, the log messages are appended to the existing file.

msdexport utility command

The msdexport utility command is used to export a system source definition (.msd) file from the command line, outside of the Integration Flow Designer.

The msdexport utility command returns 0 if the export is successful, and 1 if not. It is a batch operation that can also be used for automation.

See "[Troubleshooting](#)" for details about capturing and evaluating the return code resulting from running this command.

- [**msdexport syntax summary**](#)
This is the syntax that is used with the msdexport utility command.
- [**msdexport utility command options**](#)
These are the command options that can be used with the msdexport utility command.
- [**msdexport command line help**](#)
You can access command line help for the msdexport utility command.
- [**Using the msdexport command**](#)
You can use the msdexport command to export the system source file from the command line.

msdexport syntax summary

This is the syntax that is used with the msdexport utility command.

msdexport is the name of the utility command used to export system source files.

The .msd file is a required field and is the file name of the system source file that needs to be exported. If the full path of the system source file name is not specified, the msdexport utility command will search for the system source file in the current directory.

None of the options in the msdexport utility command are case sensitive. The following is the syntax of the msdexport utility command:

```
msdexport <.msd file>  
[-NO]  
[-O <.xml file name/location>]  
[-ASVR [<server names>]] [-ASYS [<system names>]]  
[-EPWD] [-REFSYS] [-REFSVR]  
[(-LOG [. | <log file name/location>]  
[-FAIL] [-VERBOSE] [-APPEND])  
|-NOLOG]
```

Typing `msdexport` with no options will display the command summary on the console.

msdexport utility command options

These are the command options that can be used with the `msdexport` utility command.

The following command options are available with the `msdexport` utility command:

• -APPEND msdexport Option	• -NO msdexport Option
• -ASVR msdexport Option	• -NOLOG msdexport Option
• -ASYS msdexport Option	• -O msdexport Option
• -EPWD msdexport Option	• -REFSVR msdexport Option
• -FAIL msdexport Option	• -REFSYS msdexport Option
• -LOG msdexport Option	• -VERBOSE msdexport Option

-APPEND msdexport option

The `-APPEND` option can be used with the `msdexport` utility command.

-ASVR msdexport option

The `-ASVR` option can be used with the `msdexport` utility command.

-ASYS msdexport option

The `-ASYS` option can be used with the `msdexport` utility command.

-EPWD msdexport option

The `-EPWD` option can be used with the `msdexport` utility command.

-FAIL msdexport option

The `-FAIL` option can be used with the `msdexport` utility command.

-LOG msdexport option

The `-LOG` option can be used with the `msdexport` utility command.

-NO msdexport option

The `-NO` option can be used with the `msdexport` utility command.

-NOLOG msdexport option

The `-NOLOG` option can be used with the `msdexport` utility command.

-O msdexport option

The `-O` option can be used with the `msdexport` utility command.

-REFSVR msdexport option

The `-REFSVR` option can be used with the `msdexport` utility command.

-REFSYS msdexport option

The `-REFSYS` option can be used with the `msdexport` utility command.

-VERBOSE msdexport option

The `-VERBOSE` option can be used with the `msdexport` utility command.

-APPEND msdexport option

The `-APPEND` option can be used with the `msdexport` utility command.

The `append` (`-APPEND`) option of the `msdexport` utility command is used to indicate that the current `msdexport` log messages should be appended to the existing `msdexport` log file if it already exists. If the `msdexport` log `append` (`-APPEND`) option is selected, the `msdexport` log messages will be appended to the existing file if it already exists; otherwise, a new file will be created. If the (`-APPEND`) option is not selected, the `msdexport` log file will be created if no log file exists, or it will be overwritten if the log file does exist.

-ASVR msdexport option

The `-ASVR` option can be used with the `msdexport` utility command.

The server (`-ASVR`) option of the `msdexport` utility command is used to export only servers from the system definition (`.msd`) file. If the option is used alone, the `msdexport` utility command will export all the servers from the `.msd` file. If the option is followed by a string of server names separated by commas, for example, `-ASVR server1, server2, server3`, the `msdexport` utility command will export those specific servers from the `.msd` file.

-ASYS msdexport option

The `-ASYS` option can be used with the `msdexport` utility command.

The system (`-ASYS`) option of the `msdexport` utility command is used to export only systems from system definition (`.msd`) file. If the option is used alone, the `msdexport` utility command will export all the systems from the `.msd` file. If the option is followed by a string of system names separated by commas, for example, `-ASVR system1, system 2, system 3`, the `msdexport` utility command will export those specific systems from the `.msd` file.

Any combination of the **-ASYS** and **-ASVR** options used with the `msdexport` utility command will be executed as separate individual options.

-EPWD msdexport option

The **-EPWD** option can be used with the `msdexport` utility command.

The encrypt password (**-EPWD**) option of the `msdexport` utility command is used to encrypt the password while writing to a file.

-FAIL msdexport option

The **-FAIL** option can be used with the `msdexport` utility command.

The fail (**-FAIL**) option of the `msdexport` utility command is used to indicate that only the systems that had errors during the exporting should be included in the `msdexport` log file. If the log failures only (**-FAIL**) option is selected, the `msdexport` log will contain only the systems that failed exporting. If the log failures only (**-FAIL**) option is not selected in the command line, all the exported results will be written to the `msdexport` log file.

-LOG msdexport option

The **-LOG** option can be used with the `msdexport` utility command.

The log (**-LOG**) option of the `msdexport` utility command is used to enable logging. The log file will be produced showing the result of the exported system source file. (**-LOG**) has an optional parameter to specify the location or file name, or both, for the `msdexport` log file. The default file name for the `msdexport` log will be **system_source_file_name_export.log**. The default location for the `msdexport` log will be the directory where the system definition (**.msd**) file is located.

The file name or the location parameter, if specified with the (**-LOG**) option of the `msdexport` utility command, could be the name of the directory where the `msdexport` log file will be written, the full file path specification in which a wildcard can be used to represent the xml source file, or the full path in which the complete filename can be specified. If the **.** is specified with the (**-LOG**) option, the `msdexport` log will be written to the default location (directory where the (**.msd**) file is located) using the default naming convention. If a wildcard is specified in the full path, for example, `msdexport mysystems.msd -LOG C:\MyDev\ExportResults\2003-10-03_*.results`, the `msdexport` log file will be produced in the **C:\MyDev\ExportResults** directory with the **2003-10-03_mysystems.results** naming convention. If the complete file name is specified, for example, `msdexport mysystems.mms -LOG C:\MyDev\ExportResults\mysystemsexport_results.txt`, the **mysystemsexport_results.txt** will be produced in the **C:\MyDev\ExportResults** directory.

If the `msdexport` utility command is unable to create the `msdexport` log file, the exporting of the system source file will terminate and the **Fatal error - Could not start exporting the system source file, unable to create the log file** fatal error message will display on the console. If nothing is specified in the command line for the log, by default, the command line assumes the (**-LOG**) option is included in the command line, and writes the log messages to the console.

-NO msdexport option

The **-NO** option can be used with the `msdexport` utility command.

The no overwrite (**-NO**) option of the `msdexport` utility command is used to indicate that an existing exported xml (**.xml**) file is not to be overwritten. If the no overwrite (**-NO**) option is specified in the command line of `msdexport`, and a file with the same name as the resulting **.msd** file exists, it will not be overwritten, the `msdexport` operation will fail, and the **IFD export error - The exported xml file cannot be overwritten, file already exists** message will be written to the `msdexport` log file, if enabled. If the (**-NO**) option is not specified, the file with the same name as the resulting **.msd** file, will be overwritten.

-NOLOG msdexport option

The **-NOLOG** option can be used with the `msdexport` utility command.

The disable log (**-NOLOG**) option of the `msdexport` utility command is used to turn off the `msdexport` logging capability. If the disable logging option is selected, the log file will not be produced. The optional (**-APPEND**, and **-VERBOSE**) options of the (**-LOG**) option cannot be used with the disable log (**-NOLOG**) option. If these options are selected with the (**-NOLOG**) option, the command line is invalid, and the **Not a valid command line - (LOG) optional commands (-APPEND), (-VERBOSE)** cannot be used with disable log command (**-NOLOG**). error message will display on the console.

-O msdexport option

The **-O** option can be used with the `msdexport` utility command.

The output (**-O**) option of the `msdexport` utility command is used to specify, through a parameter, the location or file name, or both, for the exported xml file produced by exporting the system source file. The specified parameter could be the name of the directory where the exported xml file is written, the full file path specification in which a wildcard can be used to represent the exported xml file, or the full path in which the complete filename can be specified. The default naming convention for the exported xml file is **system_source_file_name.xml**. If a wildcard is specified in the full path, for example, `msdexport mysystem.msd -O C:\MyDev\ExportResults\2003-10-03_*.xml`, the resulting xml file is produced in the **C:\MyDev\ExportResults** directory with the **2003-10-03_mysystem.xml** naming convention. If the complete file name is specified, for example, `msdexport mysystem.mms -O C:\MyDev\ExportResults\MysystemExportedfile.xml`, an xml file **MysystemExportedfile.xml** is produced in the **C:\MyDev\ExportResults** directory. If the export output (**-O**) option is selected, and if the file name is not specified, the command line is invalid, and the

Not a valid command line – the file name parameter is required if export command option (-O) is selected error message displays on the console.

-REFSVR msdexport option

The -REFSVR option can be used with the msdexport utility command.

The reference server (-REFSVR) option of the `msdexport` utility command is used to allow reference servers to be exported.

-REFSYS msdexport option

The -REFSYS option can be used with the msdexport utility command.

The reference system (-REFSYS) option of the `msdexport` utility command is used to allow reference systems to be exported, also. If the -REFSYS command is specified, the `msdexport` utility command will also export the dependent systems.

-VERBOSE msdexport option

The -VERBOSE option can be used with the msdexport utility command.

The verbose (-VERBOSE) option of the `msdexport` utility command is used to indicate if the verbose `msdexport` log should be produced. The verbose `msdexport` log will contain an entry for the system source file being exported, the exporting start date/time, the exporting end date/time, the exported system source file name, and the result of the `msdexport` utility command. If the verbose (-VERBOSE) option is not selected, a concise version of the `msdexport` log will be produced, which includes an entry for the system source file name, and exporting result.

msdexport command line help

You can access command line help for the msdexport utility command.

There is help that describes the `msdexport` utility command usage in the command line.

To view the command line help for the msdexport utility command:

Enter `msdexport` at the command line prompt in the `install_dir`.

The syntax and descriptions of all the options appear on the console.

Using the msdexport command

You can use the msdexport command to export the system source file from the command line.

Use the `msdexport` utility command to export the system source file from the command line, outside the Integration Flow Designer.

The following example shows how you can use the command:

```
install_dir> msdexport mysystem.mms  
-O C:\MyDev\ExportResults\mysystem.xml  
-LOG -APPEND
```

When you run the example, the following result occurs:

- If the `mysystems.mms` system source file is exported successfully, the `msdexport` – completed successfully. message displays on the console.
- The exported `mysystems.xml` xml file is produced in the `C:\MyDev\ExportResults` directory.
- If the `mysystems.log` log file does not exist, it is produced in the same directory as the system source file. If it already exists, the log messages are appended to the existing file.

Utility commands for map tuning

The map tuning utility commands are used to calculate map memory page size and count and running your maps to create output consisting of statistical data regarding the execution of your maps.

Map tuning utility commands are tools to calculate map memory page size and count and running your maps to create output consisting of statistical data regarding the execution of your maps. This data can assist you in tuning your maps to attain improved performance.

The following list briefly describes the functions of each of the map tuning utility commands:

- "["dtxpath Utility Command"](#)" - calculates suggested settings for memory page size and count for maps
- "["dtxpath Utility Command"](#)" - profiles maps and analyzes map execution behavior

- [dtxpage utility command](#)

The dtxpage utility command is used to invoke the **Page Setting Assistant for Maps** application to calculate suggested settings for memory page size and count for work files used in maps. To use this utility command, the Command Server must be installed.

- [dtxprof utility command](#)

The dtxprof utility command is used to profile maps and analyze map execution behavior from the command line, outside of the Map Designer GUI. To use this utility command, the Command Server must be installed.

dtxpage utility command

The dtxpage utility command is used to invoke the **Page Setting Assistant for Maps** application to calculate suggested settings for memory page size and count for work files used in maps. To use this utility command, the Command Server must be installed.

The dtxpage utility command returns 0 if the calculations are successful, and 1 if not successful. It is a batch operation that can also be used for automation.

The .mmc file is a required parameter and is the file name of the map for which the page size and count is being calculated. If the full path of the map file name is not specified, the dtxpage utility command will search for the map in the current directory.

The **Page Setting Assistant for Maps** application will run iterations of the specified map to calculate suggested memory page size and count settings. Because of the multiple iterations of map runs, the overall run time will be longer than a typical run. Because of the additional run time, follow these guidelines:

- use a smaller representation of the input data when you are using a map that normally takes a long time to run
- use this utility command only when there is a limited number of other processes running so that CPU utilization is not adversely affected

See "[Troubleshooting](#)" for details about capturing and evaluating the return code resulting from the execution of this utility command.

You can use the dtxpage utility command to calculate suggested settings for memory page size and count for work files used in maps that you are running with the z/OS Batch Command Server. See "[z/OS Batch Command Server dtxpage Usage](#)".

- [dtxpage syntax summary](#)

This is the syntax that is used with the dtxpage utility command.

- [dtxpage command line help](#)

You can access command line help for the dtxpage utility command.

- [dtxpage command usage](#)

You can use the dtxpage command to invoke the **Page Setting Assistant for Maps** application to calculate suggested settings for memory page size and count for work files used in maps.

dtxpage syntax summary

This is the syntax that is used with the dtxpage utility command.

dtxpage is the name of the utility command used to calculate suggested memory page size and count settings for work files.

The following is the syntax of the dtxpage utility command:

`dtxpage <.mmc file name/location>`

There are no other options besides the compiled map (.mmc) file name and location.

Typing dtxpage with no options will display the command summary on the console.

dtxpage command line help

You can access command line help for the dtxpage utility command.

There is help that describes the dtxpage utility command usage in the command line.

To view the command line help for the dtxpage utility command

1. Enter dtxpage at the command line prompt in the *install_dir*.
The syntax and descriptions of all the options appear on the console.

dtxpage command usage

You can use the dtxpage command to invoke the **Page Setting Assistant for Maps** application to calculate suggested settings for memory page size and count for work files used in maps.

The following example shows how you can use the dtxpage utility command:

`install_dir> dtxpage mymap.mmc`

When you run the example, the following result occurs:

- If the suggested settings for memory page size and count are calculated for the map file successfully, first a list of the sizes and counts for each map run by the `dtxpage` utility command displays on the console. The list also includes the time stamp for each map run.
- Following the list of the sizes and counts for each map, are the suggested settings for memory page size and count that the `dtxpage` utility command calculated.
- [dtxpage command usage with the z/OS Batch Command Server](#)
You can use the `dtxpage` command with the z/OS Batch Command Server.

dtxpage command usage with the z/OS Batch Command Server

You can use the `dtxpage` command with the z/OS Batch Command Server.

The `dtxpage` utility command is available to use with the z/OS Batch Command Server. It calculates suggested settings for memory page size and count for the work files used in a specified map.

To use the `dtxpage` utility command:

1. Make a copy of the JCL file that executes the map on which the `dtxpage` utility command performs its calculations.
2. Change the program name from **MERCATOR** or **DTXCMDSV** to **DTXPAGE** in one of the copies of the JCL file.
The **Page Analysis Usage** report is written to an output specified on the **SYSPRINT** Data Definition (**DD**) statement.

dtxprof utility command

The `dtxprof` utility command is used to profile maps and analyze map execution behavior from the command line, outside of the Map Designer GUI. To use this utility command, the Command Server must be installed.

The `dtxprof` utility command returns 0 if the Map Profiler is successful and 1 if it is not successful. It is a batch operation that can also be used for automation.

The Map Profiler is a user-configurable utility that captures and reports map execution statistics. The profiler focuses on component and mapping rules and the functions and types within those rules. The resulting information enables you to see where performance is lagging and in turn make improvements in your maps.

For example, in the profile report you see that the processing time for a `LOOKUP` function is significantly more than other functions. You can modify the rule to use a better choice, such as the `SEARCHUP` function (because in this specific case the data in the `LOOKUP` is ascending), and as a result the processing time is greatly reduced. See *Data Search Usage* in the IBM Transformation Extender *Performance Recommendations* documentation for more information about recommended usages of the different search functions to achieve your performance objectives.

See "[Troubleshooting](#)" for details about capturing and evaluating the return code resulting from the execution of this utility command.

You can use the `dtxprof` utility command to calculate suggested settings for memory page size and count for work files used in maps that you are running with the z/OS Batch Command Server. See "[z/OS Batch Command Server dtxprof Usage](#)".

- [Syntax summary for dtxprof](#)
- [Utility command options for dtxprof](#)
- [Command line help for dtxprof](#)
- [Using the dtxprof command](#)

Syntax summary for dtxprof

`dtxprof` is the name of the utility command used to analyze map execution behavior.

The output file name and compiled map name (.mmc) are required fields.

The following is the syntax of the `dtxprof` utility command:

```
dtxprof
[-f [x]]
[-t [x]]
[-fs]
[-ts]
[-d]
-o output_file
-dtx "map_name [map_options]"
```

None of the options in the `dtxprof` utility command are case sensitive. Typing `dtxprof` with no options will display the command summary on the console.

Utility command options for dtxprof

The following command options are available with the `dtxprof` utility command:

• -f dtxprof Option	• -ts dtxprof Option
• -fx dtxprof Option	• -d dtxprof Option
• -t dtxprof Option	• -o dtxprof Option

• -tx dtxprof Option	• -dtx dtxprof Option
• -fs dtxprof Option	

If you type `dtxprof` with no options, the command line help that describes the command will display on your screen.

- [-f dtxprof option](#)
- [-fx dtxprof option](#)
- [-t dtxprof option](#)
- [-tx dtxprof option](#)
- [-fs dtxprof option](#)
- [-ts dtxprof option](#)
- [-d dtxprof option](#)
- [-o dtxprof option](#)
- [-dtx dtxprof option](#)

-f dtxprof option

The function times (-f) option of the `dtxprof` utility command is used to specify that all the function processing times are to be reported in the output.

-fx dtxprof option

The function times exceeding x milliseconds (-fx) option of the `dtxprof` utility command is used to specify that all the function processing times that exceed a specified value (x) are to be reported in the output. The value must be between 0 and 250 (1/10 milliseconds).

-t dtxprof option

The type times (-t) option of the `dtxprof` utility command is used to specify that all the type processing times are to be reported in the output.

-tx dtxprof option

The type times exceeding x milliseconds (-tx) option of the `dtxprof` utility command is used to specify that all the type processing times that exceed a specified value (x) are to be reported in the output. The value must be between 0 and 250 (1/10 milliseconds).

-fs dtxprof option

The function times summary output (-fs) option of the `dtxprof` utility command is used to specify that all the function processing times are to be reported in summary form in the output.

-ts dtxprof option

The type times summary output (-ts) option of the `dtxprof` utility command is used to specify that all the type processing times are to be reported in summary form in the output.

-d dtxprof option

The comma-delimited output (-d) option of the `dtxprof` utility command is used to specify that the output format will be comma-delimited.

The default value is fixed-width so if the -d option is omitted, the output will be created in fixed-width format.

-o dtxprof option

The output file name (-o) option of the `dtxprof` utility command is used to specify the output file name (*filename*).

-dtx dtxprof option

The (-dtx) option of the `dtxprof` utility command is used to specify the compiled map name (*map_name*) with optional map options (*map_options*). Quotation marks around the map name are required.

Command line help for `dtxprof`

There is help that describes the `dtxprof` utility command usage in the command line.

To view the command line help for the `dtxprof` utility command:

Enter `dtxprof` at the command line prompt in the *install_dir*.
The syntax and descriptions of all the options appear on the console.

Using the `dtxprof` command

Use the `dtxprof` utility command to profile maps and analyze map execution behavior.

The Map Profiler can capture the following statistics:

- Amount of time spent processing component and mapping rules and subordinate functions.
- Number of times rules and functions are executed (from validation through output).
- Number of times type objects are accessed.
- Depth (nesting level) of the object being profiled.
- [Type names](#)
- [Function Times](#)
- [RUN maps](#)
- [Output examples](#)
- [For best results](#)
- [z/OS Batch Command Server `dtxprof` usage](#)

Type names

Similar to the map trace file, the type names referred to in the profile output represent the full type name in the original schema, not the name of the type object that is visible in the Map Designer.

Function Times

The processing time of any subfunctions rolls up into the processing time of the top-level function. For example, if an `EITHER` function contains an `IF` function, the recorded time for the `EITHER` function would include the processing time of the `IF` function.

RUN maps

When a parent map contains a `RUN` map, the `RUN` map is not profiled. To profile a `RUN` map, it must be done outside of the `RUN` rule. You can do this by running it as a top-level executable map.

Output examples

Full report

The following command produces a full report containing both function and type information:

```
dtxprof -f -t -o sinkpro  
-dtx "C:\IBM\WebSphere Transformation Extender 9.0.0\  
examples\general\map\sinkmap\sinkmap.mmc"
```

Individual Function Breakdown

function	depth	iterations	time	map	type
<hr/>					
validation					
VALIDATE_CARD	0	1	298	sinkmap	
VALIDATE_CARD	0	1	59	sinkmap	
<hr/>					
mapping					
BUILD_CARD	0	1	71	sinkmap	
LOOKUP	1	1	42	sinkmap	week example

TYPE_NAME	2	8	2	sinkmap	week example
BUILD_CARD	0	1	65	sinkmap	total_hours element example
NUMBERTOTEXT	1	1	44	sinkmap	total_hours element example
ADD_OP	2	1	43	sinkmap	total_hours element example
ADD_OP	3	1	37	sinkmap	total_hours element example
ADD_OP	4	1	33	sinkmap	total_hours element example
ADD_OP	5	1	28	sinkmap	total_hours element example
DIV_OP	6	1	24	sinkmap	total_hours element example
SUB_OP	7	1	22	sinkmap	total_hours element example
TEXTTONUMBER	8	1	20	sinkmap	total_hours element example
TIMETOTEXT	9	1	18	sinkmap	total_hours element example
TYPE_NAME	10	1	17	sinkmap	total_hours element example
TEXTTONUMBER	8	1	1	sinkmap	total_hours element example
TIMETOTEXT	9	1	1	sinkmap	total_hours element example
TYPE_NAME	10	1	0	sinkmap	total_hours element example
FLOAT_CONSTANT	7	1	1	sinkmap	total_hours element example
DIV_OP	6	1	4	sinkmap	total_hours element example
SUB_OP	7	1	3	sinkmap	total_hours element example
TEXTTONUMBER	8	1	1	sinkmap	total_hours element example
TIMETOTEXT	9	1	0	sinkmap	total_hours element example
TYPE_NAME	10	1	0	sinkmap	total_hours element example
FLOAT_CONSTANT	7	1	0	sinkmap	total_hours element example
DIV_OP	5	1	4	sinkmap	total_hours element example
SUB_OP	6	1	3	sinkmap	total_hours element example
TEXTTONUMBER	7	1	0	sinkmap	total_hours element example
TIMETOTEXT	8	1	0	sinkmap	total_hours element example
TYPE_NAME	9	1	0	sinkmap	total_hours element example
TEXTTONUMBER	7	1	0	sinkmap	total_hours element example
TIMETOTEXT	8	1	0	sinkmap	total_hours element example
TYPE_NAME	9	1	0	sinkmap	total_hours element example
FLOAT_CONSTANT	6	1	0	sinkmap	total_hours element example
SUB_OP	5	1	3	sinkmap	total_hours element example
TEXTTONUMBER	6	1	1	sinkmap	total_hours element example
TIMETOTEXT	7	1	0	sinkmap	total_hours element example
TYPE_NAME	8	1	0	sinkmap	total_hours element example
TEXTTONUMBER	6	1	1	sinkmap	total_hours element example
TIMETOTEXT	7	1	0	sinkmap	total_hours element example
TYPE_NAME	8	1	0	sinkmap	total_hours element example
FLOAT_CONSTANT	5	1	0	sinkmap	total_hours element example
DIV_OP	3	1	3	sinkmap	total_hours element example
SUB_OP	4	1	2	sinkmap	total_hours element example
TEXTTONUMBER	5	1	1	sinkmap	total_hours element example
TIMETOTEXT	6	1	1	sinkmap	total_hours element example
TYPE_NAME	7	1	0	sinkmap	total_hours element example
TEXTTONUMBER	5	1	1	sinkmap	total_hours element example
TIMETOTEXT	6	1	0	sinkmap	total_hours element example
TYPE_NAME	7	1	0	sinkmap	total_hours element example
FLOAT_CONSTANT	4	1	0	sinkmap	total_hours element example
BUILD_CARD	0	1	50	sinkmap	employee example
TYPE_NAME	1	1	1	sinkmap	total_hours element example
TYPE_NAME	1	1	31	sinkmap	total_hours element example

Type per Rule Breakdown

time	map	type
---	---	----
validation		
mapping		
428	sinkmap	week example
44	sinkmap	
65	sinkmap	total_hours element example
322	sinkmap	
50	sinkmap	
1	sinkmap	employee example
31	sinkmap	total_hours element example

Summary report

The following command produces a summary report containing both function and type information:

```
dtxprof -fs -ts -o sinkmappro
-dtx "examples\general\map\sinkmap\sinkmap.mmc"
```

Type per Rule Summary

iterations	time	map	type
-----	----	--	----
validation			
mapping			
1	228	sinkmap	week example
2	431	sinkmap	total_hours element example
1	1	sinkmap	employee example

Function Summary

function	iterations	time
-----	-----	-----
VALIDATE_CARD	2	3153
BUILD_CARD	3	2415
TYPE_NAME	20	65
FLOAT_CONSTANT	5	1

ADD_OP	4	167
SUB_OP	5	37
DIV_OP	5	44
LOOKUP	1	226
NUMBERTOTEXT	1	51
TIMETOTEXT	10	34
TEXTTONUMBER	10	35

For best results

- If you have a large amount of data, the profiler output could be very large or unmanageable. For this reason, it is best to profile only a subset of the original data.
- You can reduce profiler output by specifying a time limit so that nothing is reported until it exceeds the time limit that you specify.
- When running the profiler, always limit or terminate other processes running on your machine to avoid adverse results.

z/OS Batch Command Server dtxprof usage

The `dtxprof` utility command is available to use with the z/OS Batch Command Server. It profiles maps and analyzes map execution behavior.

To use the `dtxprof` utility command:

1. Make a copy of the JCL file that executes the map on which the `dtxprof` utility command profiles.
2. Change the program name from **MERCATOR** or **DTXCMDSV** to **DTXPROF** in one of the copies of the JCL file.
3. Specify the command line switches you want to use with the `dtxprof` utility command in the **PARM** field on the **EXEC** statement of the JCL.
If the **PARM** field length exceeds the 100-character limit, place the **DTXCMDSV** map command line in a file and specify `-dtx "-@ddname"` in the **PARM** field along with the `dtxprof` command line you want to use.
4. Add **PROF1** and **PROF2** Data Definition (**DD**) cards.

dtxprof JCL example

The following example JCL code shows how you can use the `dtxprof` utility command with the z/OS Batch Command Server.

```
//STEP1 EXEC PGM=DTXPROF,REGION=0M,
// PARM='-f -t -fs -ts -o DD:PROFOUT -dtx "-@CMD" '
...
...
//CMD      DD *
      REVERSE /VX0D,X0A REVERSEI /VX0D,X0A REVERSEO
/*
//PROF1    DD DSN=&&PROF01,DCB=(RECFM=FBS,LRECL=23476,BLKSIZE=23476),
//           UNIT=3390,SPACE=(CYL,(0100,050),RLSE),
//           DISP=(NEW,DELETE,DELETE)
//PROF2    DD DSN=&&PROF02,DCB=(RECFM=FBS,LRECL=23476,BLKSIZE=23476),
//           UNIT=3390,SPACE=(CYL,(0100,050),RLSE),
//           DISP=(NEW,DELETE,DELETE)
//PROFOUT DD SYSOUT=*
...
...
```

`-@ddname` identifies the Data Definition (**DD**) name that points to the file containing the **DTXCMDSV** map command line.

Utility command for dtxany2xml

The `dtxany2xml` utility is used to produce a map that can transform any input data into XML output.

The `dtxany2xml` utility performs the following tasks during execution:

- Exports the input schema as an XML Schema
- Creates an XML schema from the generated XML Schema
- Creates a map source file containing two maps:
 - The first map has the name that you specified in the command line.
 - The second map has the same name as the first map but with `_r` appended to the end of it.

Use the second map only when the schema that you specified in the first parameter of the `dtxany2xml` utility command was imported from a **COBOL** copybook.

The `dtxany2xml` utility command returns 0 if the transformation into XML output is successful and 1 if it is not successful. It is a batch operation and can also be used for automation.

See "["Troubleshooting"](#)" for details about capturing and evaluating the return code resulting from the execution of this utility command.

- [dtxany2xml syntax summary](#)
The `dtxany2xml` utility command produces a map that can transform any input data into XML output.
- [dtxany2xml command line help](#)
There is help that describes the `dtxany2xml` utility command usage in the command line.

- [dtxany2xml command usage](#)

Use the **dtxany2xml** command to produce a map that can transform any input data into XML output.

dtxany2xml syntax summary

The **dtxany2xml** utility command produces a map that can transform any input data into XML output.

It creates one map that can transform any input data into XML output, and a second map that can transform the XML that the first map produced into output data in a format described by the schema that you specified.

Use the second map only when the schema that you specified in the first parameter of the **dtxany2xml** utility command was imported from a COBOL copybook.

Use the following syntax for the **dtxany2xml** utility command:

```
dtxany2xml <.mtt file name/location>
    <exported root type path>
    <input file name/location>
    <output file name/location>
    <output map name>
    [<-schema <schema_file_name/location>]
    [<log file name/location>]
```

The **dtxany2xml** utility generates the default schema name *orig_schema_name_exported_type_name.xsd*. If a schema with that name already exists, the utility overwrites it.

You can use the **-schema** keyword to specify a unique name and path for the schema. The **-schema** keyword is optional and positional.

dtxany2xml command line help

There is help that describes the **dtxany2xml** utility command usage in the command line.

To view the command line help for the **dtxany2xml** utility command:

1. Enter **dtxany2xml** at the command line prompt of the installation directory. The syntax and command descriptions are displayed.

dtxany2xml command usage

Use the **dtxany2xml** command to produce a map that can transform any input data into XML output.

This procedure assumes that you know the schema and type name that models the XML output you want to produce. It also assumes that if you are using the second map, you know the format of the schema that was imported from a **COBOL** copybook.

1. Open a command prompt and change to the product installation directory.
2. Use the following command format. The names and paths of the schema file and the log file are optional. All other command elements are required.
`dtxany2xml <schema (.mtt) name and path> <root type path to export> <input file name and path> <new output file name (.xml) and path> <new map name> [<-schema <schema_file_name/location>][<log file name and path >]`

When the utility runs successfully, you can find the expected files in the directory that you selected. No "success" message is displayed. A log file is generated only when the process does not complete successfully.

- [Scenario: dtxany2xml usage](#)

The following example shows how you can use the **dtxany2xml** utility command:

Scenario: dtxany2xml usage

The following example shows how you can use the **dtxany2xml** utility command:

This exercise uses sample files that are shipped with the product.

To create a map that can transform any input data into XML output:

1. From a command prompt, change your directory location:
`cd c:/Program Files/IBM/WebSphere Transformation Extender 8.1/ examples/general/states`
2. Enter the following command:
`dtxany2xml states.mtt Data:Input:UnitedStates sts.txt xml_output.xml XML_map XML_log.log`

This command produces the following results:

- A new XML Schema – named using the schema name plus the selected type name (*states_UnitedStates.xsd*).
- A new map source file (*XML_map.mms*) that contains two maps:
 - first map (*XML_map*)
 - The input card references the original schema (*states.mtt*) and input file (*sts.txt*).
 - The output card references the newly created XML schema (*states_UnitedStates.mtt*).
 - The type is **Doc XSD**.
 - The output file is *xml_output.xml*.

- second map (XML_map_r)
 - The input card references the newly created XML schema (states_UnitedStates.mtt) and file (xml_output.xml).
 - The output card references the original schema (states.mtt).
 - The type is **Data Input UnitedStates**.
 - The output file is sts.txt.
- Use the second map only when the schema that you specified in the first parameter of the **dtxany2xml** utility command was imported from a **COBOL** copybook.

- The log file (XML_log.log) is created only when an error occurs.
3. Using Map Designer, open *install_dir\examples\general\states\ XML_map.mms*.
 4. Build and run XML_map.

The input (sts.txt) is converted to XML output (xml_output.xml). The input (xml_output.xml), which was created by the first map, is converted back to text output (sts.txt).

Utility command for map trace-file conversion

The **dtxmaptrace** utility converts a map trace file from binary format to text format.

Syntax summary

```
dtxmaptrace [[-summaryonly | -nosummary] -o output_file] [-toascii] binary_trace.mtr
```

You can display a summary of the converted trace messages on the computer screen, or save the full set of converted trace messages to a file, with or without the summary.

Table 1. Trace output options for the **dtxmaptrace** utility

Trace message format	Trace message destination	Summary keyword	Output file
Summary only	screen	null	null
Summary only	file	-summaryonly	-o output_file
All trace messages	screen	Not supported (with or without summary)	Not applicable
All trace messages without summary	file	-nosummary	-o output_file
All trace messages with summary	file	null	-o output_file

-summaryonly

Produces a concise summary of the trace messages in outline format. This keyword is optional. If you omit it, the output file includes all trace messages, preceded by the message summary.

-nosummary

Includes all trace messages in the output file. This keyword is optional. If you omit it, the output file includes all trace messages, preceded by the message summary.

-o *output_file*

The name of the converted trace file. The output file name must be different from the binary trace file name. This keyword is optional. If you omit it, the **dtxmaptrace** utility prints summary-only trace information to the computer screen.

-toascii

Use the **-toascii** option on a non-z/OS® system to convert a binary trace file that was created on a z/OS system. When the z/OS trace file is converted to text, the **-toascii** option changes the character encoding from EBCDIC to ASCII, so that the file is readable on the non-z/OS system.

binary_trace.mtr

The name of the input binary trace file to be converted. This keyword is required and positional. It must be the last option on the **dtxmaptrace** command.

Utility command for Resource Registry

The Resource Registry utility command updates resources and virtual servers in resource name (.mrn) files by using the command line.

The utility command for this application is:

- ResourceRegistryHelper.bat on Windows and
- ResourceRegistryHelper.sh on UNIX

Use the Resource Registry utility command to make the following updates to .mrn files:

- Add resources and virtual servers to .mrn files.
- Modify resources and virtual servers in .mrn files.
- Remove resources and virtual servers from .mrn files.

- [Resource Registry syntax summary](#)

This is the syntax that is used with the Resource Registry utility command.

- [Resource registry command usage](#)

These examples use the Resource Registry utility commands to add, remove, or change resources in a resource name (.mrn) file. The examples use ResourceRegistryHelper.bat for the Windows platform. For UNIX, use ResourceRegistryHelper.sh.

Resource Registry syntax summary

This is the syntax that is used with the Resource Registry utility command.

Use the ResourceRegistryHelper.bat utility command on Windows systems or the ResourceRegistryHelper.sh on UNIX systems to update resources and virtual servers in resource name (.mrn) files. The command returns 0 if the update is successful, and 1 if it is not successful. It is a batch operation that you can use for automation.

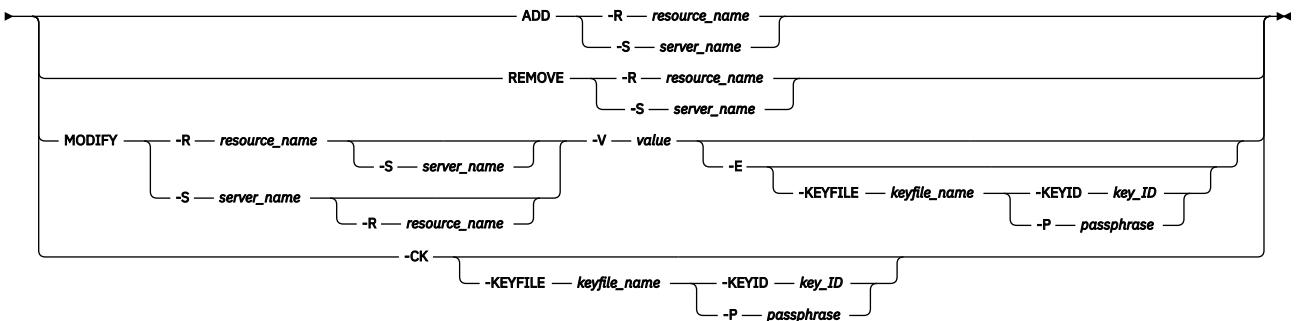
To display help for the command, enter `ResourceRegistryHelper.bat` at the command line prompt in the IBM Transformation Extender installation directory.

The options in the Resource Registry utility command are not case-sensitive.

```
resourceRegistryhelper.bat [path/file_name.mrn]
-A { ADD {-R resource_name | -S virtual_server_name}
| REMOVE {-R resource_name | -S virtual_server_name}
| MODIFY {-R resource_name | -S virtual_server_name | -R resource_name -S virtual_server_name}
-V value
[-E [-KEYFILE keyfile_path [-KEYID keyID | -P passphrase]]]
| -CK [-E [-KEYFILE keyfile_path [-KEYID keyID | -P passphrase]]]}
```

The command is shown in railroad track format below:

`>> resourceRegistryhelper.bat — path/file_name.mrn — -A —>`



Keyword

Description

- A The action (-A) option of the Resource Registry utility command indicates the action (MODIFY, ADD or REMOVE) to perform on a specified resource or virtual server in the resource name file (.mrn) file.
- ADD Adds a specified resource or virtual server to the resource name (.mrn) file.
- REMOVE Removes a specified resource or virtual server from the .mrn file.
- MODIFY Changes the value of a specified resource, virtual server, or both resource and virtual server in the .mrn file. The -V parameter is required with the MODIFY command.
 - If you specify only the resource name (-R), the command changes the value of that resource across all servers.
 - If you specify only the server name (-S), the command changes all of that server's resources to the new value.
 - If you specify both the resource name and the server name, the command changes the resource value only for the specified server.
- CK Changes the master key that encrypts the values in the specified .mrn file to a different master key. You can specify an existing master key by using the -KEYID option, or generate a new master key with a custom passphrase or a random passphrase.
- R *resource_name* Specifies the name of the resource that you want to add, remove, or change.
- S *virtual_server_name* Specifies the name of the virtual server that you want to add, remove, or change.
- V *value* Specifies the new value of the resource or virtual server that you are changing.
- E Encrypts the specified resource or virtual server value in the .mrn file. The -E option is valid only on the MODIFY command. When you omit the -E option, the resource or virtual server name is not encrypted. This is the default.
- KEYFILE *keyfile_path* Specifies the name and path of the master key (.mkf) file, relative to the .mrn file. The .mkf file stores Resource Registry encryption keys. -KEYFILE is optional. If you omit it, the command uses the default itx.mkf file in the .mrn file directory as the source of existing master keys or to store a new master key. To generate a new random master key, specify the -KEYFILE option without the -KEYID or the -P options.
- KEYID *key_ID* The numeric identifier of an existing key that the MODIFY command is to use to encrypt a resource value. The key identifier is optional. If you do not specify the -KEYID option or the -P option, the command generates a new encryption key with a random passphrase. You cannot regenerate a key that was created with a random passphrase.
- P *passphrase* A 32-byte passphrase that is to identify a new encryption key. Any character restrictions?The passphrase is optional. If you omit both the -KEYID option and the -P option, the command generates a new encryption key with a random passphrase. You cannot regenerate a key that was created with a random passphrase.

Resource registry command usage

These examples use the Resource Registry utility commands to add, remove, or change resources in a resource name (.mrn) file. The examples use `ResourceRegistryHelper.bat` for the Windows platform. For UNIX, use `ResourceRegistryHelper.sh`.

Modify examples

These examples modify a resource or virtual server in the modifyingresource.mrn file.

- Change the value of the **test** resource to *c:/server/location*:

```
install_dir\resourceregistryhelper.bat modifyingresource.mrn
-A modify
-R test
-V c:/server/location
```

- Encrypt the value of the **test** resource. Use encryption key 3257 located in the itx.mkf master key file in the .mrn directory:

```
install_dir\resourceregistryhelper.bat modifyingresource.mrn
-A modify
-R test
-E -KEYFILE itx.mkf -KEYID 3257
```

- Change the resource value of the **aix** virtual server to */server/location*. Remove the current encryption and re-encrypt the value with a new encryption key. The key is generated with a random passphrase and stored in the TempAIXServers.mkf master key file :

```
install_dir\resourceregistryhelper.bat modifyingresource.mrn
-A modify
-S aix
-V /server/location
-CK -KEYFILE TempAIXServers.mkf
```

Add examples

These examples add a resource or virtual server to the addingintoresource.mrn resource name file.

- Add the **test** resource to the .mrn file:

```
install_dir\resourceregistryhelper.bat addingintoresource.mrn
-A add
-R test
```

- Add the **aix** virtual server to the .mrn file:

```
install_dir\resourceregistryhelper.bat addingintoresource.mrn
-A add
-S aix
```

Remove examples

These examples remove a resource or virtual server from the removinginresource.mrn resource name file.

- Remove the **test** resource from the .mrn file:

```
install_dir\resourceregistryhelper.bat removinginresource.mrn
-A remove
-R test
```

- Remove the **aix** virtual server from the .mrn file:

```
install_dir\resourceregistryhelper.bat removinginresource.mrn
-A remove
-S aix
```

TX programming interface introduction

The TX programming interface includes language-specific methods and objects that provide common functionality across multiple environments.

TX programming interface overview

The TX programming interface is an object-oriented approach that enables applications to invoke IBM Transformation Extender maps, thus removing reliance on command lines.

For more information about the TX PI client interfaces, see the Javadocs by opening the package-summary.html file located in
install_dir\docs\dtxpi\java\com\ibm\websphere\dtx\dtxpi.

- [**TX programming interface methods**](#)

The TX programming interface map, card, and adapter object properties define the properties that can be set for each object. Use these methods to GET or SET the object properties.

- [**TX programming interface object overview**](#)

The TX programming interface is based on map, card, adapter, connection, and stream objects.

- [**TX programming interface function overview**](#)

Use the TX programming interface to load, run and control maps.

TX programming interface methods

The TX programming interface map, card, and adapter object properties define the properties that can be set for each object. Use these methods to GET or SET the object properties.

- [**GetAllPropertiesXML**](#)
Retrieves all object properties in XML document.
 - [**GetBinaryProperty**](#)
This method gets a pointer to the specified binary property of the object. The data is not copied by this method.
 - [**GetCountProperty**](#)
This method returns the number of defined property values.
 - [**GetIntegerProperty**](#)
This method gets the specified integral property of the object.
 - [**GetTextProperty**](#)
This method gets a pointer to the specified text property of the object. The data is not copied by this method.
 - [**GetPropertiesXML**](#)
Retrieves requested property values as an XML document. Properties are specified in the form of an XML document.
 - [**SetBinaryProperty**](#)
This method sets the specified binary property on the object.
 - [**SetIntegerProperty**](#)
This method sets the specified integral property on the object.
 - [**SetTextProperty**](#)
This method sets the specified text property on the object.
 - [**SetPropertiesXML**](#)
Sets properties specified in XML document. Properties are specified in the form of an XML document.
-

GetAllPropertiesXML

Retrieves all object properties in XML document.

Properties are specified in the form of an XML document.

The XML schema for this XML document is txpi_properties.xsd, located in the installation directory. Both the request document, which specifies which properties to return, and the returned document comply to this XML schema.

GetBinaryProperty

This method gets a pointer to the specified binary property of the object. The data is not copied by this method.

GetCountProperty

This method returns the number of defined property values.

This is the maximum index value for indexed properties. For most properties, this method will return a count of 1.

GetIntegerProperty

This method gets the specified integral property of the object.

GetTextProperty

This method gets a pointer to the specified text property of the object. The data is not copied by this method.

GetPropertiesXML

Retrieves requested property values as an XML document. Properties are specified in the form of an XML document.

The XML schema for this XML document is txpi_properties.xsd located in the installation directory. Both the request document, which specifies which properties to return, and the returned document comply to this XML schema.

SetBinaryProperty

This method sets the specified binary property on the object.

SetIntegerProperty

This method sets the specified integral property on the object.

SetTextProperty

This method sets the specified text property on the object.

SetPropertiesXML

Sets properties specified in XML document. Properties are specified in the form of an XML document.

The XML schema for this XML document is txpi_properties.xsd located in the installation directory.

TX programming interface object overview

The TX programming interface is based on map, card, adapter, connection, and stream objects.

Objects are used for source and target overrides for map execution.

- If no card overrides are required, only the map object is referenced. All other objects are created internally.
- If card overrides are required, the card and adapter objects are referenced.

Overrides are accomplished using the defined property interface.

Adapter objects interface with the adapter API. The TX programming interface is based on IBM Transformation Extender objects.

Object	Datatype	Object description
Map	HMPIMAP	An object that represents an instance of a map in the program memory. Methods allow the map properties to be modified and for the map to be executed and its execution controlled.
Card	HMPICARD	An object that represents an input or output card of a map in program memory. Defined methods allow the card properties to be modified.
Adapter	HMPIADAPT	An object that represents a resource adapter. Defined methods allow the adapter properties to be modified.
Connection	HMPICONNECT	An object that represents a connection to a resource. When a map executes, a Connection object is associated with one or more Adapter objects.
Stream	HMPISTREAM	An object that provides a means to pass data to, or get data from, a map. Stream objects eliminate the need for large, contiguous, memory buffers for passing data directly to a map object.

- [TX programming interface object hierarchy](#)

The TX programming interface objects have a relationship to each other defined by a specific hierarchy.

- [TX programming interface properties](#)

The TX programming interface methods provide a means to query or modify properties of objects.

- [TX programming interface map object](#)

The TX programming interface creates a map object instance from the byte array representing a compiled map file. The hierarchies of map, card and adapter objects are created by this method.

- [Card object overview](#)

The card object is an abstraction of a card in a map.

- [Adapter object overview](#)

An adapter object is created for each card when a map instance object is created.

TX programming interface object hierarchy

The TX programming interface objects have a relationship to each other defined by a specific hierarchy.

Under the map object is the card object and under the card object is the adapter object. Each card object always has an adapter object associated with it. A map object has a number of card objects. The number is dependent on the number of input and output cards in a map.

TX programming interface properties

The TX programming interface methods provide a means to query or modify properties of objects.

Each of these method calls has an index parameter, used in cases where there might be more than one instance of a property method (for example, multiple input card objects for a map).

For properties that are not indexed, the index should be specified as zero (0).

TX programming interface map object

The TX programming interface creates a map object instance from the byte array representing a compiled map file. The hierarchies of map, card and adapter objects are created by this method.

The map object is created when a map is loaded from a compiled map (.mmc) file or from memory. The TX programming interface caches maps. Therefore, if multiple map objects are created for the same map, the map file is read only once.

A map object is reentrant; that is, after a map object has been created, the map can be run an infinite number of times without any need for the map to be reloaded.

Multiple threads cannot reference the same map object concurrently; If multiple threads need to execute the same map concurrently, a separate map object should be obtained for each thread.

The identifier is used to identify the specific map. This might be the name of the map or any other identifier that the application will use. Subsequent calls to this method with the same identifier will not introduce additional processing of the compiled map. Therefore, after the initial call to this method, the memory consumed by compiledMap can be freed.

There are several properties and methods supported by the map object.

- [**Map object properties**](#)
Properties can be set and their values can be obtained by executing the methods defined by the map object.
- [**Map object methods**](#)
The map object methods are defined by the map object.

Map object properties

Properties can be set and their values can be obtained by executing the methods defined by the map object.

The map object supports several property types.

- [**General map properties**](#)
The general properties for map objects provide high-level information about the map object.
- [**Map trace properties**](#)
The trace properties for map objects allow you to set the trace settings for map execution.
- [**Map workspace properties**](#)
The workspace properties for map objects allow you to define the workspace settings for map execution.
- [**Map sliding century properties**](#)
Use the sliding century properties for map objects to define the sliding century settings for map execution.
- [**Map custom validation properties**](#)
Use the custom validation properties for map objects to define custom validation settings for map execution.
- [**Map audit properties**](#)
The audit properties for map objects allow you to set the audit settings for map execution.
- [**Map retry properties**](#)
The map retry properties for map objects allow you to define the map retry settings for map execution.

General map properties

The general properties for map objects provide high-level information about the map object.

- [**MPIP_MAP_DESTROY_OBJECT_POOL**](#)
If the value is set to TRUE, the objects related to the map instance will be destroyed after the map is executed.
- [**MPIP_MAP_INSTANCE**](#)
This is a unique instance number for the map object.
- [**MPIP_MAP_MAP_NAME**](#)
This is the fully qualified name of the map for the map object.
- [**MPIP_OBJECT_ERROR_CODE**](#)
The error code from the last operation on the map object.
- [**MPIP_OBJECT_ERROR_MSG**](#)
The error message from the last operation on the map object.

MPIP_MAP_DESTROY_OBJECT_POOL

If the value is set to TRUE, the objects related to the map instance will be destroyed after the map is executed.

The default value is FALSE.

Type	Values
Integer	MPI_TRUE MPI_FALSE

MPIP_MAP_INSTANCE

This is a unique instance number for the map object.

Type	Values
Integer	None

MPIP_MAP_MAP_NAME

This is the fully qualified name of the map for the map object.

Type	Values
Text	None

MPIP_OBJECT_ERROR_CODE

The error code from the last operation on the map object.

Type	Values
Integer	None

MPIP_OBJECT_ERROR_MSG

The error message from the last operation on the map object.

Type	Values
Text	None

Map trace properties

The trace properties for map objects allow you to set the trace settings for map execution.

- [**MPIP_MAP_TRACE_DIRECTORY**](#)
The directory for the trace file is either map or custom.
- [**MPIP_MAP_TRACE_DIRECTORY_CUSTOM_VALUE**](#)
This is the directory for the trace file if MPIP_MAP_TRACE_DIRECTORY is set to MPI_DIRECTORY_CUSTOM.
- [**MPIP_MAP_TRACE_FILENAME**](#)
The trace file name can be the default name (*mapname.mtr*), a custom name, or a unique, system-generated name (*prefix_mapname_processIDtime_number_hostname.mtr*), where *prefix* is Mer for executable maps or Run for run maps.
- [**MPIP_MAP_TRACE_FILENAME_CUSTOM_VALUE**](#)
This is the filename for the trace file if MPIP_MAP_TRACE_FILENAME is set to MPI_FILENAME_CUSTOM.
- [**MPIP_MAP_TRACE_FORMAT**](#)
This property creates the map trace file in text format (the default) or binary format.
- [**MPIP_MAP_TRACE_INPUT_CARD_END**](#)
This is the number of the last card to trace if MPIP_MAP_TRACE_INPUT_CONTENT is MPI_TRACE_RANGE.
- [**MPIP_MAP_TRACE_INPUT_CARD_NUMBER**](#)
This is the number of the card to trace if MPIP_MAP_TRACE_INPUT_CONTENT is MPI_TRACE_CARD.
- [**MPIP_MAP_TRACE_INPUT_CARD_START**](#)
This is the number of the first card to trace if MPIP_MAP_TRACE_INPUT_CONTENT is MPI_TRACE_RANGE.
- [**MPIP_MAP_TRACE_INPUT_CONTENT**](#)
This determines the scope of the input tracing.
- [**MPIP_MAP_TRACE_LOCATION**](#)
Create the trace information in a file or memory.
- [**MPIP_MAP_TRACE_RULES_CARD_NUMBER**](#)
This specifies the card number if MPIP_MAP_TRACE_RULES_CONTENT is set to MPI_TRACE_CARD.
- [**MPIP_MAP_TRACE_RULES_CONTENT**](#)
This turns the trace rules content on or off.
- [**MPIP_MAP_TRACE_SUMMARY_CONTENT**](#)
Controls whether the trace file contains a summary of the validation status of each input and the build status of each output.
- [**MPIP_MAP_TRACE_SWITCH**](#)
This is the master switch for map trace.

MPIP_MAP_TRACE_DIRECTORY

The directory for the trace file is either map or custom.

Type	Values
Integer	MPI_DIRECTORY_MAP MPI_DIRECTORY_CUSTOM

MPIP_MAP_TRACE_DIRECTORY_CUSTOM_VALUE

This is the directory for the trace file if MPIP_MAP_TRACE_DIRECTORY is set to MPI_DIRECTORY_CUSTOM.

Type	Values
Text	None

MPIP_MAP_TRACE_FILENAME

The trace file name can be the default name (*mapname.mtr*), a custom name, or a unique, system-generated name (*prefix_mapname_processIDtime_number_hostname.mtr*), where *prefix* is Mer for executable maps or Run for run maps.

Type	Values
Integer	<ul style="list-style-type: none">• MPI_FILENAME_DEFAULT• MPI_FILENAME_CUSTOM• MPI_FILENAME_UNIQUE

MPIP_MAP_TRACE_FILENAME_CUSTOM_VALUE

This is the filename for the trace file if MPIP_MAP_TRACE_FILENAME is set to MPI_FILENAME_CUSTOM.

Type	Values
Text	None

MPIP_MAP_TRACE_FORMAT

This property creates the map trace file in text format (the default) or binary format.

Type	Values
Integer	<ul style="list-style-type: none">• MPIP_MAP_TRACE_FORMAT_TEXT• MPIP_MAP_TRACE_FORMAT_BINARY

MPIP_MAP_TRACE_INPUT_CARD_END

This is the number of the last card to trace if MPIP_MAP_TRACE_INPUT_CONTENT is MPI_TRACE_RANGE.

Type	Values
Integer	2, ...

MPIP_MAP_TRACE_INPUT_CARD_NUMBER

This is the number of the card to trace if MPIP_MAP_TRACE_INPUT_CONTENT is MPI_TRACE_CARD.

Type	Values
Integer	1, ...

MPIP_MAP_TRACE_INPUT_CARD_START

This is the number of the first card to trace if MPIP_MAP_TRACE_INPUT_CONTENT is MPI_TRACE_RANGE.

Type	Values
Integer	1, ...

MPIP_MAP_TRACE_INPUT_CONTENT

This determines the scope of the input tracing.

Type	Values
Integer	MPI_TRACE_OFF
	MPI_TRACE_ALL
	MPI_TRACE_CARD
	MPI_TRACE_RANGE

MPIP_MAP_TRACE_LOCATION

Create the trace information in a file or memory.

Type	Values
Integer	MPI_LOCATION_FILE
	MPI_LOCATION_MEMORY

MPIP_MAP_TRACE_RULES_CARD_NUMBER

This specifies the card number if MPIP_MAP_TRACE_RULES_CONTENT is set to MPI_TRACE_CARD.

Type	Values
Integer	1, ...

MPIP_MAP_TRACE_RULES_CONTENT

This turns the trace rules content on or off.

Type	Values
Integer	MPI_TRACE_OFF
	MPI_TRACE_ALL
	MPI_TRACE_CARD
	MPI_TRACE_RANGE

MPIP_MAP_TRACE_SUMMARY_CONTENT

Controls whether the trace file contains a summary of the validation status of each input and the build status of each output.

Type	Values
Integer	<ul style="list-style-type: none">MPI_SWITCH_ONMPI_SWITCH_OFF

MPIP_MAP_TRACE_SWITCH

This is the master switch for map trace.

Turn trace on or off. When off, all other trace properties are ignored.

Type	Values
Integer	<ul style="list-style-type: none">MPI_SWITCH_ONMPI_SWITCH_OFF

Map workspace properties

The workspace properties for map objects allow you to define the workspace settings for map execution.

- MPIP_MAP_WORKSPACE_DIRECTORY**

The directory for the workspace file is either map or custom.

- MPIP_MAP_WORKSPACE_DIRECTORY_CUSTOM**

This is the directory for the workspace file if MPIP_MAP_WORKSPACE_DIRECTORY is set to MPI_DIRECTORY_CUSTOM.

- **MPIP_MAP_WORKSPACE_FILE_ACTION**
If MPIP_MAP_WORKSPACE_FILE_PREFIX is set to MPI_FILE_PREFIX_MAP, this determines whether workfiles can be reused.
 - **MPIP_MAP_WORKSPACE_FILE_PREFIX**
If MPIP_MAP_WORKSPACE_LOCATION is set to MPI_LOCATION_FILE, this property determines the prefix of the file.
 - **MPIP_MAP_WORKSPACE_LOCATION**
Create the workspace in files or memory.
 - **MPIP_MAP_WORKSPACE_PAGING_COUNT**
This is the number of workspace pages to create.
 - **MPIP_MAP_WORKSPACE_PAGING_SIZE**
This is the size of each workspace page.
-

MPIP_MAP_WORKSPACE_DIRECTORY

The directory for the workspace file is either map or custom.

Type	Values
Integer	MPI_DIRECTORY_MAP
	MPI_DIRECTORY_CUSTOM

MPIP_MAP_WORKSPACE_DIRECTORY_CUSTOM

This is the directory for the workspace file if MPIP_MAP_WORKSPACE_DIRECTORY is set to MPI_DIRECTORY_CUSTOM.

Type	Values
Text	None

MPIP_MAP_WORKSPACE_FILE_ACTION

If MPIP_MAP_WORKSPACE_FILE_PREFIX is set to MPI_FILE_PREFIX_MAP, this determines whether workfiles can be reused.

Type	Values
Integer	MPI_WORK_AREA_REUSE
	MPI_WORK_AREA_DONTREUSE

MPIP_MAP_WORKSPACE_FILE_PREFIX

If MPIP_MAP_WORKSPACE_LOCATION is set to MPI_LOCATION_FILE, this property determines the prefix of the file.

Type	Values
Integer	MPI_FILE_PREFIX_MAP
	MPI_FILE_PREFIX_UNIQUE

MPIP_MAP_WORKSPACE_LOCATION

Create the workspace in files or memory.

Type	Values
Integer	MPI_LOCATION_FILE
	MPI_LOCATION_MEMORY

MPIP_MAP_WORKSPACE_PAGING_COUNT

This is the number of workspace pages to create.

Type	Values
Integer	None

MPIP_MAP_WORKSPACE_PAGING_SIZE

This is the size of each workspace page.

Type	Values
Integer	None

Map sliding century properties

Use the sliding century properties for map objects to define the sliding century settings for map execution.

- [MPIP_MAP_SLIDING_CENTURY](#)
This is the master switch for sliding century properties.
- [MPIP_MAP_SLIDING_CENTURY_CCLOOKUP](#)
This is the base year that defines the beginning of the sliding century.

MPIP_MAP_SLIDING_CENTURY

This is the master switch for sliding century properties.

Turn sliding century on or off. When off, all other sliding century properties are ignored.

Type	Values
Integer	MPI_SWITCH_ON
Integer	MPI_SWITCH_OFF

MPIP_MAP_SLIDING_CENTURY_CCLOOKUP

This is the base year that defines the beginning of the sliding century.

Type	Values
Integer	None

Map custom validation properties

Use the custom validation properties for map objects to define custom validation settings for map execution.

- [MPIP_MAP_CUSTOM_VALIDATION](#)
This is the master switch for custom validation.
- [MPIP_MAP_CV_COMPONENT_RULE](#)
Ignore or raise a validation error if component rules are violated.
- [MPIP_MAP_CV_PRESENTATION_ERROR](#)
Ignore or raise a validation error if item presentations are violated.
- [MPIP_MAP_CV_RESTRICTION_ERROR](#)
Ignore or raise a validation error if restrictions are violated.
- [MPIP_MAP_CV_SIZE_ERROR](#)
Ignore or raise a validation error if item sizes are violated.
- [MPIP_MAP_CV_VALIDATION_ERROR](#)
Stop or continue mapping if a validation error occurs.

MPIP_MAP_CUSTOM_VALIDATION

This is the master switch for custom validation.

Turn custom validation on or off. When off, all other custom validation properties are ignored.

Type	Values
Integer	MPI_SWITCH_ON
Integer	MPI_SWITCH_OFF

MPIP_MAP_CV_COMPONENT_RULE

Ignore or raise a validation error if component rules are violated.

Type	Values
Integer	MPI_ACTION_IGNORE
Integer	MPI_ACTION_VALIDATE

MPIP_MAP_CV_PRESENTATION_ERROR

Ignore or raise a validation error if item presentations are violated.

Type	Values
Integer	MPI_ACTION_IGNORE MPI_ACTION_VALIDATE MPI_ACTION_IGNORE_NO_WARNINGS

MPIP_MAP_CV_RESTRICTION_ERROR

Ignore or raise a validation error if restrictions are violated.

Type	Values
Integer	MPI_ACTION_IGNORE MPI_ACTION_VALIDATE MPI_ACTION_IGNORE_NO_WARNINGS

MPIP_MAP_CV_SIZE_ERROR

Ignore or raise a validation error if item sizes are violated.

Type	Values
Integer	MPI_ACTION_IGNORE MPI_ACTION_VALIDATE MPI_ACTION_IGNORE_NO_WARNINGS

MPIP_MAP_CV_VALIDATION_ERROR

Stop or continue mapping if a validation error occurs.

Type	Values
Integer	MPI_ACTION_IGNORE MPI_ACTION_VALIDATE MPI_ACTION_IGNORE_NO_WARNINGS

Map audit properties

The audit properties for map objects allow you to set the audit settings for map execution.

- [MPIP_MAP_AUDIT_BURST_DATA](#)
Turn data audit on or off.
- [MPIP_MAP_AUDIT_BURST_DATA_SIZE_VALIDATION](#)
Specifies whether to use two separate audit log status codes or a single code to indicate a size error or warning for data items that fail the minimum/maximum criteria when the Validation > SizeError map setting is set to Ignore.
- [MPIP_MAP_AUDIT_BURST_EXECUTION](#)
Turn execution audit on or off.
- [MPIP_MAP_AUDIT_DIRECTORY](#)
The directory for the audit file is either map or custom.
- [MPIP_MAP_AUDIT_DIRECTORY_CUSTOM_VALUE](#)
This is the directory for the audit file if MPIP_MAP_AUDIT_DIRECTORY is set to MPI_DIRECTORY_CUSTOM.
- [MPIP_MAP_AUDIT_FILENAME](#)
The filename for the audit file is either the default or a custom name.
- [MPIP_MAP_AUDIT_FILENAME_ACTION](#)
The audit file can either be appended to or created anew.
- [MPIP_MAP_AUDIT_FILENAME_CUSTOM_VALUE](#)
This is the filename for the audit file if MPIP_MAP_AUDIT_FILENAME is set to MPI_FILENAME_CUSTOM.
- [MPIP_MAP_AUDIT_LOCATION](#)
Create the audit information in a file or in memory.
- [MPIP_MAP_AUDIT_MEMORY_SIZED](#)
If MPIP_MAP_AUDIT_LOCATION is set to MPI_LOCATION_MEMORY, this determines whether the memory should be sized.
- [MPIP_MAP_AUDIT_SETTINGS_DATA](#)
Turn data settings audit on or off.
- [MPIP_MAP_AUDIT_SETTINGS_MAP](#)
Turn map settings audit on or off.

- [**MPIP_MAP_AUDIT_SUMMARY_EXECUTION**](#)
Turn execution summary audit on or off for the map object.
 - [**MPIP_MAP_AUDIT_SWITCH**](#)
This is the master switch for map audit.
-

MPIP_MAP_AUDIT_BURST_DATA

Turn data audit on or off.

Type	Values
Integer	MPI_AUDIT_ALWAYS MPI_AUDIT_NEVER MPI_AUDIT_ONERROR MPI_AUDIT_ONWARNING_ERROR

MPIP_MAP_AUDIT_BURST_DATA_SIZE_VALIDATION

Specifies whether to use two separate audit log status codes or a single code to indicate a size error or warning for data items that fail the minimum/maximum criteria when the Validation > SizeError map setting is set to Ignore.

MPI_DATAAUDIT_WRONGSIZE specifies one code while MPI_DATAAUDIT_MINMAXERR specifies two separate codes.

Type	Values
Integer	MPI_DATAAUDIT_WRONGSIZE MPI_DATAAUDIT_MINMAXERR

MPIP_MAP_AUDIT_BURST_EXECUTION

Turn execution audit on or off.

Type	Values
Integer	MPI_AUDIT_ALWAYS MPI_AUDIT_NEVER MPI_AUDIT_ONERROR MPI_AUDIT_ONWARNING_ERROR

MPIP_MAP_AUDIT_DIRECTORY

The directory for the audit file is either map or custom.

Type	Values
Integer	MPI_DIRECTORY_MAP MPI_DIRECTORY_CUSTOM

MPIP_MAP_AUDIT_DIRECTORY_CUSTOM_VALUE

This is the directory for the audit file if MPIP_MAP_AUDIT_DIRECTORY is set to MPI_DIRECTORY_CUSTOM.

Type	Values
Integer	None

MPIP_MAP_AUDIT_FILENAME

The filename for the audit file is either the default or a custom name.

Type	Values
Integer	MPI_FILENAME_DEFAULT MPI_FILENAME_CUSTOM

MPIP_MAP_AUDIT_FILENAME_ACTION

The audit file can either be appended to or created anew.

Type	Values
Integer	MPI_FILE_CREATE MPI_FILE_APPEND

MPIP_MAP_AUDIT_FILENAME_CUSTOM_VALUE

This is the filename for the audit file if MPIP_MAP_AUDIT_FILENAME is set to MPI_FILENAME_CUSTOM.

Type	Values
Text	None

MPIP_MAP_AUDIT_LOCATION

Create the audit information in a file or in memory.

Type	Values
Integer	MPI_LOCATION_FILE MPI_LOCATION_MEMORY

MPIP_MAP_AUDIT_MEMORY_SIZED

If MPIP_MAP_AUDIT_LOCATION is set to MPI_LOCATION_MEMORY, this determines whether the memory should be sized.

Type	Values
Integer	MPI_TRUE MPI_FALSE

MPIP_MAP_AUDIT_SETTINGS_DATA

Turn data settings audit on or off.

Type	Values
Integer	MPI_SWITCH_ALWAYS MPI_SWITCH_NEVER MPI_SWITCH_ONERROR MPI_SWITCH_ONWARNING_ERROR

MPIP_MAP_AUDIT_SETTINGS_MAP

Turn map settings audit on or off.

Type	Values
Integer	MPI_SWITCH_ALWAYS MPI_SWITCH_NEVER MPI_SWITCH_ONERROR MPI_SWITCH_ONWARNING_ERROR

MPIP_MAP_AUDIT_SUMMARY_EXECUTION

Turn execution summary audit on or off for the map object.

Type	Values
Integer	MPI_SWITCH_ON MPI_SWITCH_OFF MPI_SWITCH_ONERROR MPI_SWITCH_ONWARNING_ERROR

MPIP_MAP_AUDIT_SWITCH

This is the master switch for map audit.

Turn audit on or off. When off, all other audit properties are ignored.

Type	Values
Integer	MPI_SWITCH_ON
	MPI_SWITCH_OFF
	MPI_SWITCH_ONERROR
	MPI_SWITCH_OWARNING_ERROR

Map retry properties

The map retry properties for map objects allow you to define the map retry settings for map execution.

- [**MPIP_MAP_MAP_RETRY**](#)
This is the master switch for map retry.
- [**MPIP_MAP_MAP_RETRY_ATTEMPTS**](#)
This is the maximum number of attempts to attempt to retry map execution.
- [**MPIP_MAP_MAP_RETRY_INTERVAL**](#)
This is the interval in seconds after which map retries are attempted.

MPIP_MAP_MAP_RETRY

This is the master switch for map retry.

Turn map retry on or off. When off, all other map retry properties are ignored.

Type	Values
Integer	MPI_SWITCH_ON
	MPI_SWITCH_OFF

MPIP_MAP_MAP_RETRY_ATTEMPTS

This is the maximum number of attempts to attempt to retry map execution.

Type	Values
Integer	None

MPIP_MAP_MAP_RETRY_INTERVAL

This is the interval in seconds after which map retries are attempted.

Type	Values
Integer	None

Map object methods

The map object methods are defined by the map object.

The map object provides several methods.

- [**MapGetInputCardNumber**](#)
This method returns the number of input cards.
- [**MapGetInputCardObject**](#)
This method returns the handle to the specified input card.
- [**MapGetOutputCardNumber**](#)
This method returns the number of output cards.
- [**MapGetOutputCardObject**](#)
This method returns the handle to the specified output card.

MapGetInputCardNumber

This method returns the number of input cards.

MapGetInputCardObject

This method returns the handle to the specified input card.

MapGetOutputCardNumber

This method returns the number of output cards.

MapGetOutputCardObject

This method returns the handle to the specified output card.

Card object overview

The card object is an abstraction of a card in a map.

A card object is created for each card when a map object is created. This object is a container for the card properties and associates a map and an adapter object with the card object.

- [Card object properties](#)
Properties can be set and their values can be obtained by executing the methods defined by the card object.
- [Card object methods](#)
The card object methods are defined by the card object.

Card object properties

Properties can be set and their values can be obtained by executing the methods defined by the card object.

The card object supports the General and Backup property types.

- [General card properties](#)
The general properties for card objects provide high-level information about the card object.
- [Backup card properties](#)
Use the backup properties for card objects to define the backup settings of the card for map execution.

General card properties

The general properties for card objects provide high-level information about the card object.

The MPIP_CARD_MODE and MPIP_CARD_REUSE_WORK_AREA properties apply to input cards only.

- [MPIP_CARD_DIRECTION](#)
Indicates whether the card is an input or output card.
- [MPIP_CARD_DOCUMENT_VERIFICATION](#)
Determines under what circumstances to call the external document verification program when an instance of an object with an XML type property is found.
- [MPIP_CARD_METADATA_LOCATION](#)
This is the location of the metadata (XML schema or DTD).
- [MPIP_CARD_MODE](#)
MPIP_CARD_MODE is the mode of the card (input card only).
- [MPIP_CARD_NAME](#)
The name of the card.
- [MPIP_CARD_NUMBER](#)
The number of the card.
- [MPIP_CARD_OBJECT_COUNT](#)
The number of objects found in validation (if an input card) or found in build (if an output card).
- [MPIP_CARD_REUSE_WORK_AREA](#)
Determines whether the work area for the card is reused (input card only).
- [MPIP_OBJECT_ERROR_CODE](#)
The error code from the last operation on the map object.
- [MPIP_OBJECT_ERROR_MSG](#)
The error message from the last operation on the map object.

MPIP_CARD_DIRECTION

Indicates whether the card is an input or output card.

Type	Values
Integer	MPI_CARD_INPUT
	MPI_CARD_OUTPUT

MPIP_CARD_DOCUMENT_VERIFICATION

Determines under what circumstances to call the external document verification program when an instance of an object with an XML type property is found.

Type	Values
Integer	MPI_VERIFY_NEVER
	MPI_VERIFY_ALWAYS
	MPI_VERIFY_ONFAILURE
	MPI_VERIFY_ONSUCCESS
	MPI_VERIFY_WELLFORMED

MPIP_CARD_METADATA_LOCATION

This is the location of the metadata (XML schema or DTD).

Use this card object property to override the metadata location specified on the map card settings at run time.

Type	Values
Text	user-defined

MPIP_CARD_MODE

MPIP_CARD_MODE is the mode of the card (input card only).

Type	Values
Integer	MPI_MODE_INTEGRAL
	MPI_MODE_BURST

MPIP_CARD_NAME

The name of the card.

Type	Values
Text	None

MPIP_CARD_NUMBER

The number of the card.

Type	Values
Integer	1, ...

MPIP_CARD_OBJECT_COUNT

The number of objects found in validation (if an input card) or found in build (if an output card).

Type	Values
Integer	None

MPIP_CARD_REUSE_WORK_AREA

Determines whether the work area for the card is reused (input card only).

Type	Values

Type	Values
Integer	MPI_TRUE MPI_FALSE

MPIP_OBJECT_ERROR_CODE

The error code from the last operation on the map object.

Type	Values
Integer	None

MPIP_OBJECT_ERROR_MSG

The error message from the last operation on the map object.

Type	Values
Text	None

Backup card properties

Use the backup properties for card objects to define the backup settings of the card for map execution.

- [**MPIP_CARD_BACKUP**](#)
This is the master switch for turning backup on or off.
- [**MPIP_CARD_BACKUP_ACTION**](#)
Determines whether to append to an existing file or to create a new file.
- [**MPIP_CARD_BACKUP_DIRECTORY**](#)
The directory in which the backup file is located.
- [**MPIP_CARD_BACKUP_DIRECTORY_CUSTOM_VALUE**](#)
The filename for the trace file if MPIP_CARD_BACKUP_DIRECTORY is set to MPI_DIRECTORY_CUSTOM.
- [**MPIP_CARD_BACKUP_FILENAME**](#)
The name of the backup file.
- [**MPIP_CARD_BACKUP_FILENAME_CUSTOM_VALUE**](#)
The filename for the trace file if MPIP_CARD_BACKUP_FILENAME is set to MPI_FILENAME_CUSTOM.
- [**MPIP_CARD_BACKUP_FILEPATH**](#)
The path of the backup file.
- [**MPIP_CARD_BACKUP_WHEN**](#)
Determines under what circumstances to create a backup file.

MPIP_CARD_BACKUP

This is the master switch for turning backup on or off.

Turn backup on or off. When off, all other backup properties are ignored.

Type	Values
Integer	MPI_SWITCH_ON MPI_SWITCH_OFF

MPIP_CARD_BACKUP_ACTION

Determines whether to append to an existing file or to create a new file.

Type	Values
Integer	MPI_ACTION_CREATE MPI_ACTION_APPEND

MPIP_CARD_BACKUP_DIRECTORY

The directory in which the backup file is located.

Type	Values
Integer	MPI_DIRECTORY_MAP MPI_DIRECTORY_CUSTOM

MPIP_CARD_BACKUP_DIRECTORY_CUSTOM_VALUE

The filename for the trace file if MPIP_CARD_BACKUP_DIRECTORY is set to MPI_DIRECTORY_CUSTOM.

Type	Values
Text	None

MPIP_CARD_BACKUP_FILENAME

The name of the backup file.

Type	Values
Integer	MPI_FILENAME_DEFAULT MPI_FILENAME_CUSTOM MPI_FILENAME_UNIQUE

MPIP_CARD_BACKUP_FILENAME_CUSTOM_VALUE

The filename for the trace file if MPIP_CARD_BACKUP_FILENAME is set to MPI_FILENAME_CUSTOM.

Type	Values
None	None

MPIP_CARD_BACKUP_FILEPATH

The path of the backup file.

Type	Values
Text	None

MPIP_CARD_BACKUP_WHEN

Determines under what circumstances to create a backup file.

Type	Values
Integer	MPI_WHEN_ALWAYS MPI_WHEN_ONERROR

Card object methods

The card object methods are defined by the card object.

The card object provides several methods.

- [**CardGetAdapterObject**](#)
This method returns the adapter object associated with the specified card object.
- [**CardGetAdapterType**](#)
This method returns the type of the adapter object that is currently associated with the specified card object.
- [**CardGetMapObject**](#)
This method returns the map object associated with the specified card object.
- [**CardOnNotify**](#)
This method can be called to provide a notification for adapters of key events.
- [**CardOverrideAdapter**](#)
This method can be called to associate a new adapter object with the card object. This is used whenever the type of the adapter needs to be changed programmatically from the type that is compiled into the map.

CardGetAdapterObject

This method returns the adapter object associated with the specified card object.

CardGetAdapterType

This method returns the type of the adapter object that is currently associated with the specified card object.

Adapter types are listed in the adapters.xml file. The type is the 'id' attribute of each adapter entry.

CardGetMapObject

This method returns the map object associated with the specified card object.

CardOnNotify

This method can be called to provide a notification for adapters of key events.

Value	Description
MPIN_ADAPTER_GETSTART	Prior to any get calls for the card
MPIN_ADAPTER_GETSTOP	After all get calls for the card
MPIN_ADAPTER_PUTSTART	Prior to any put calls for the card
MPIN_ADAPTER_PUTSTOP	After all put calls for the card
MPIN_ADAPTER_LISTENSTART	Prior to any listen calls for the card
MPIN_ADAPTER_LISTENSTOP	After all listen calls for the card
MPIN_ADAPTER_LISTENABORT	Called to abort a non-polling listener
MPIN_ADAPTER_MAPABORT	When a map is cancelled by user action
MPIN_OBJECT_PREPARE_DESTROY	Called prior to an object being destroyed

These entry points provide the opportunity for an adapter to initialize or clean up the GET, PUT or LISTEN methods.

For example, the adapter might need to query a queue to find out some properties about the queue. This would be done in the onNotify (*_START) call. If this was done with the GET, PUT, or LISTEN methods and they were called multiple times, unnecessary work would be performed repeatedly.

onNotify is called with *_START prior to any GET, PUT, or LISTEN method and is called with *_STOP after all GET, PUT, or LISTEN calls.

For example, if a map is executing multiple bursts and within each burst, GET is called multiple times to get multiple messages, there will be only one onNotify call with MPIN_GET_START, prior to the first get. Only after all bursts and all gets within a burst, will onNotify be called with MPIN_GET_STOP to allow the adapter to clean up.

If a user cancels a map execution, onNotify will be called with MPIN_ADAPTER_MAPABORT. The GET or PUT methods should be aborted by the adapter as soon as possible after the notification.

The MPIN_ADAPTER_LISTENABORT notification is delivered only to adapters that have defined listenerBlocks as TRUE. In this scenario, onNotify is called on a separate thread from the listen call. For all other notifications, the call is made on the connection thread. When an adapter receives this notification, it should abort the LISTEN method using whatever mechanism or API that is supported by the adapter.

CardOverrideAdapter

This method can be called to associate a new adapter object with the card object. This is used whenever the type of the adapter needs to be changed programmatically from the type that is compiled into the map.

The new type can be specified either by an alias string (for example, DATABASE or STREAM) or by using a numeric adapter type. If the alias is given as NULL, iAdapterType is assumed to be set to a valid adapter type.

The adapter types are listed in the adapters.xml file. The 'id' attribute of each adapter entry represents the adapter name. Additional types are listed in CardOverrideAdapter.

Adapter object overview

An adapter object is created for each card when a map instance object is created.

This adapter object is used for custom adapter implementations.

- [Adapter object properties](#)
Properties are defined for all adapter objects, though each adapter can also GET and SET its own adapter-specific properties (for example, QueueManagerName for MQSeries®).
- [Adapter object methods](#)
The adapter object methods are defined by the adapter object.

Adapter object properties

Properties are defined for all adapter objects, though each adapter can also GET and SET its own adapter-specific properties (for example, QueueManagerName for MQSeries®).

The adapter object supports several property types.

General, On Success, On Failure, Retry, Scope, and Warning properties apply to both source and target adapter objects.

Aggregation applies only to source adapter objects.

- [**General adapter properties**](#)

These properties provide high-level information about the adapter object.

- [**Adapter retry properties**](#)

Use the adapter retry properties for adapter objects to define the settings for the retry settings for map execution.

- [**Adapter on failure properties**](#)

Use the MPIP_ADAPTER_ON_FAILURE property for adapter objects to define the settings for the OnFailure transactional settings for map execution.

- [**Adapter on success properties**](#)

Use the MPIP_ADAPTER_ON_SUCCESS property for adapter objects to define the settings for the OnSuccess transactional settings for map execution.

- [**Adapter scope properties**](#)

Use the MPIP_ADAPTER_SCOPE property for adapter objects to define the settings for the scope of the transaction for map execution.

- [**Adapter warning properties**](#)

Use the MPIP_ADAPTER_WARNINGS property for adapter objects to define the settings for action on adapter warnings for map execution.

- [**Adapter aggregation properties**](#)

Use the aggregation properties for adapter objects to define the settings that determine the number of messages that are returned from an input adapter for map execution.

General adapter properties

These properties provide high-level information about the adapter object.

- [**MPIP_ADAPTER_ABBREV**](#)

The alias of the adapter.

- [**MPIP_ADAPTER_COMMANDLINE**](#)

The command line that determines the connection parameters and the behavior of the adapter associated with the adapter object.

- [**MPIP_ADAPTER_CONTEXT**](#)

The context in which the adapter is being called.

- [**MPIP_ADAPTER_MSG_COLLECTION**](#)

The message collection object that is used to pass event messages (MMessage objects) from a Listen call to the Launcher, and from the Launcher to a GET call.

- [**MPIP_ADAPTER_NUM_MESSAGES**](#)

This property MUST be set by an adapter before returning from a GET call. This informs the Resource Manager how many messages have been received and are being returned in the GET. This is used in association with the Quantity and Fetch Unit properties to determine whether the GET should be called again or not.

- [**MPIP_ADAPTER_PROVIDER_CODE**](#)

The return code from the resource to which the adapter communicates upon execution of the input or output adapter.

- [**MPIP_ADAPTER_PROVIDER_MSG**](#)

This is the return error message from the resource to which the adapter communicates upon execution of the input or output adapter.

- [**MPIP_ADAPTER_TYPE**](#)

The type of the adapter object.

- [**MPIP_ADAPTER_USER_DATA**](#)

This property is used to associate binary data with an adapter object and can be used for various purposes, such as to pass configuration data, specific environment settings, or possibly real time data.

- [**MPIP_ADAPTER_USESAMECONN**](#)

This property is set by an adapter in the Listen method to inform the Resource Manager that the same connection thread should be used in the subsequent corresponding GET call.

- [**MPIP_ADAPTER_WILDCARD**](#)

The value of a wildcard returned from an event.

- [**MPIP_OBJECT_ERROR_CODE**](#)

The error code from the last operation on the map object.

- [**MPIP_OBJECT_ERROR_MSG**](#)

The error message from the last operation on the map object.

MPIP_ADAPTER_ABBREV

The alias of the adapter.

Type	Values
Text	None

MPIP_ADAPTER_COMMANDLINE

The command line that determines the connection parameters and the behavior of the adapter associated with the adapter object.

Type	Values
Text	None

MPIP_ADAPTER_CONTEXT

The context in which the adapter is being called.

If not SOURCE, _SOURCE_EVENT, or _TARGET, there is no card object associated with the adapter object.

Type	Values
Integer	MPI_CONTEXT_SOURCE MPI_CONTEXT_SOURCE_EVENT MPI_CONTEXT_TARGET MPI_CONTEXT_GET MPI_CONTEXT_PUT MPI_CONTEXT_DBLOOKUP MPI_CONTEXT_DBQUERY

MPIP_ADAPTER_MSG_COLLECTION

The message collection object that is used to pass event messages (MMessage objects) from a Listen call to the Launcher, and from the Launcher to a GET call.

In a GET call, the adapter should consult the collection to see if it is being called in response to an event.

Type	Values
Object	None

MPIP_ADAPTER_NUM_MESSAGES

This property MUST be set by an adapter before returning from a GET call. This informs the Resource Manager how many messages have been received and are being returned in the GET. This is used in association with the Quantity and Fetch Unit properties to determine whether the GET should be called again or not.

If this property is set to 0, this informs the Resource Manager that there is no more data to fetch, and hence the GET function will not be called again.

Type	Values
Integer	None

MPIP_ADAPTER_PROVIDER_CODE

The return code from the resource to which the adapter communicates upon execution of the input or output adapter.

Whether this is set is determined by each individual adapter.

Type	Values
Integer	None

MPIP_ADAPTER_PROVIDER_MSG

This is the return error message from the resource to which the adapter communicates upon execution of the input or output adapter.

Whether this is set is determined by each individual adapter.

Type	Values
Text	None

MPIP_ADAPTER_TYPE

The type of the adapter object.

Type	Values
Integer	The id values from the adapters.xml file.

MPIP_ADAPTER_USER_DATA

This property is used to associate binary data with an adapter object and can be used for various purposes, such as to pass configuration data, specific environment settings, or possibly real time data.

Type	Values
Binary	None

For example, the adapter might receive a command line option that requires it to process a certain file. The contents of that file could be parsed during command line validation and placed in the adapter's MPIP_ADAPTER_USER_DATA setting. The data placed in the MPIP_ADAPTER_USER_DATA setting could then be subsequently referenced by the adapter after it is invoked to get or put data at map execution time. Another example is that the adapter might want to store certain runtime data there. While bursting or upon completion of the adapter's transaction scope, the data could be referenced for additional run time or logging purposes.

MPIP_ADAPTER_USESAMECONN

This property is set by an adapter in the Listen method to inform the Resource Manager that the same connection thread should be used in the subsequent corresponding GET call.

Type	Values
Integer	None

MPIP_ADAPTER_WILDCARD

The value of a wildcard returned from an event.

Type	Values
Text	None

MPIP_OBJECT_ERROR_CODE

The error code from the last operation on the map object.

Type	Values
Integer	None

MPIP_OBJECT_ERROR_MSG

The error message from the last operation on the map object.

Type	Values
Text	None

Adapter retry properties

Use the adapter retry properties for adapter objects to define the settings for the retry settings for map execution.

- [MPIP_ADAPTER_RETRY](#)
This is the master "switch" for all adapter retry properties.
- [MPIP_ADAPTER_RETRY_ATTEMPTS](#)
The maximum number of attempts to attempt to retry adapter execution.
- [MPIP_ADAPTER_RETRY_INTERVAL](#)
The interval, in seconds, after which adapter retries are attempted.

MPIP_ADAPTER_RETRY

This is the master "switch" for all adapter retry properties.

Turn adapter retry on or off. When off, all other adapter retry properties are ignored.

Type	Values
Integer	MPI_SWITCH_ON MPI_SWITCH_OFF

MPIP_ADAPTER_RETRY_ATTEMPTS

The maximum number of attempts to attempt to retry adapter execution.

Type	Values
Integer	None

MPIP_ADAPTER_RETRY_INTERVAL

The interval, in seconds, after which adapter retries are attempted.

Type	Values
Integer	None

Adapter on failure properties

Use the MPIP_ADAPTER_ON_FAILURE property for adapter objects to define the settings for the OnFailure transactional settings for map execution.

- [MPIP_ADAPTER_ON_FAILURE](#)

This is the action to take if the map is unsuccessful.

MPIP_ADAPTER_ON_FAILURE

This is the action to take if the map is unsuccessful.

Not all settings are applicable to all values of MPIP_ADAPTER_TYPE.

Type	Values
Integer	MPI_ACTION_COMMIT MPI_ACTION_ROLLBACK

Adapter on success properties

Use the MPIP_ADAPTER_ON_SUCCESS property for adapter objects to define the settings for the OnSuccess transactional settings for map execution.

- [MPIP_ADAPTER_ON_SUCCESS](#)

This is the action to take if the map is successful.

MPIP_ADAPTER_ON_SUCCESS

This is the action to take if the map is successful.

Not all settings are applicable to all values of MPIP_ADAPTER_TYPE.

Type	Values
Integer	For source adapter objects: MPI_ACTION_KEEP MPI_ACTION_KEEPONCONTENT MPI_ACTION_DELETE For target adapter objects: MPI_ACTION_CREATE MPI_ACTION_CREATEONCONTENT MPI_ACTION_DONT_CREATE MPI_ACTION_APPEND

Adapter scope properties

Use the MPIP_ADAPTER_SCOPE property for adapter objects to define the settings for the scope of the transaction for map execution.

- [MPIP_ADAPTER_SCOPE](#)

This is the scope of the adapter's transaction.

MPIP_ADAPTER_SCOPE

This is the scope of the adapter's transaction.

Type	Values
Integer	MPI_SCOPE_CARD
	MPI_SCOPE_BURST
	MPI_SCOPE_MAP

Adapter warning properties

Use the MPIP_ADAPTER_WARNINGS property for adapter objects to define the settings for action on adapter warnings for map execution.

- [MPIP_ADAPTER_WARNINGS](#)

If the adapter returns a warning, this determines whether the map should fail.

MPIP_ADAPTER_WARNINGS

If the adapter returns a warning, this determines whether the map should fail.

Type	Values
Integer	MPI_ACTION_IGNORE
	MPI_ACTION_FAIL

Adapter aggregation properties

Use the aggregation properties for adapter objects to define the settings that determine the number of messages that are returned from an input adapter for map execution.

These properties apply to source adapter objects only.

- [MPIP_ADAPTER_FETCH_UNIT](#)

The fetch unit for a single burst.

- [MPIP_ADAPTER_LISTEN_TIME](#)

The listen timeout (in seconds).

- [MPIP_ADAPTER_QUANTITY](#)

The quantity of messages to return for the map.

MPIP_ADAPTER_FETCH_UNIT

The fetch unit for a single burst.

Type	Values
Integer	MPI_NUMBER_SOME or a positive integer

MPIP_ADAPTER_LISTEN_TIME

The listen timeout (in seconds).

Type	Values
Integer	MPI_INFINITE or a positive integer

MPIP_ADAPTER_QUANTITY

The quantity of messages to return for the map.

Type	Values
Integer	MPI_NUMBER_SOME or a positive integer

Adapter object methods

The adapter object methods are defined by the adapter object.

The map object provides several methods.

The AdaptGetCardObject, AdaptGetInputStreamObject, and AdaptGetOutputStreamObject methods are applicable to all adapter objects.

- [**AdaptGetCardObject**](#)
Returns the card object associated with the adapter object.
 - [**AdaptGetInputStreamObject**](#)
Retrieves the input stream object.
 - [**AdaptGetOutputStreamObject**](#)
Retrieves the output stream object.
-

AdaptGetCardObject

Returns the card object associated with the adapter object.

AdaptGetInputStreamObject

Retrieves the input stream object.

AdaptGetOutputStreamObject

Retrieves the output stream object.

TX programming interface function overview

Use the TX programming interface to load, run and control maps.

This interface makes it possible for users to execute and control maps within their own programs. Benefits that this interface provides:

- Object-oriented interface
 - Ability to control when you want the TX programming interface to load the maps
 - Ability to track the progress of a map execution
 - Ability to override any properties of maps, cards, or adapter objects
 - Ability to control the execution of the map
- [**Scenario: running TX programming interface maps**](#)
Code examples of running maps are installed with the Development Kit in the *install_dir\examples\dk\dtxi* directory, where *install_dir* represents the directory in which the DK is installed.
- [**Monitor and manage map overview**](#)
These methods enable the execution of a map to be controlled from another thread, and for the progress of a map to be reported.
- [**Load and run map overview**](#)
Use the loading and running methods to load specific maps, refresh maps when the maps have changed, and run maps.
-

Scenario: running TX programming interface maps

Code examples of running maps are installed with the Development Kit in the *install_dir\examples\dk\dtxi* directory, where *install_dir* represents the directory in which the DK is installed.

Example files

These examples range from the simplest possible case to a more complicated scenario involving modification of map and adapter settings.

- example1.xxx: Running a single map
- example2.xxx: Running multiple instances of a map in parallel
- example3.xxx: Overriding a card in a map
- example4.xxx: Using a user-provided status method
- example5.xxx: Using streams to override inputs and outputs
- example6.xxx: Loading a map from a byte array
- example7.xxx: Getting and setting properties

These example files demonstrate many of the interfaces described in the documentation.

Note: The examples are available for each language for which the programming interface is available. Therefore, the filename extensions vary accordingly.

Monitor and manage map overview

These methods enable the execution of a map to be controlled from another thread, and for the progress of a map to be reported.

- **[Abort](#)**
This method aborts the map defined by the map object. The map must be in the running or paused state when this method is called.
 - **[Continue](#)**
This method continues the map defined by the map object.
 - **[Pause](#)**
This method pauses the map defined by the map object.
 - **[RegisterStatusMethod](#)**
This method allows for a user-provided status method to be registered for a particular map object.
-

Abort

This method aborts the map defined by the map object. The map must be in the running or paused state when this method is called.

This map does not destroy the map instance. Therefore, it is possible to run the map once more by calling Run. It is not possible to continue the execution of a map after Abort is called. Note that Abort must be called from a thread that is separate from the thread that is executing Run.

Continue

This method continues the map defined by the map object.

The map must be in a paused state when this method is called. If the map is not paused, the call will not change the state of the map instance. Note that Continue must be called from a thread that is separate from the thread that is executing Run.

Pause

This method pauses the map defined by the map object.

The map is paused indefinitely until Continue is executed. Note that Pause must be called from a thread that is separate from the thread that is executing Run.

RegisterStatusMethod

This method allows for a user-provided status method to be registered for a particular map object.

This status method will be invoked at critical points during the execution of the map.

If the status method returns MPI_STATE_ABORTED, execution of the map will stop immediately. After aborting a map, it is possible to start it again from the beginning using the Run method.

iNotifyWhen is a bitmask that governs when the status method will be called. It can be any combination of the values listed.

Value	Description
MPI_EVENT_START_MAP	Notify whenever a map is started.
MPI_EVENT_MAP_COMPLETE	Notify whenever a map completes.
MPI_EVENT_START_BURST	Notify whenever a burst is started.
MPI_EVENT_BURST_COMPLETE	Notify whenever a burst completes.
MPI_EVENT_START_INPUT	Notify whenever an input starts.
MPI_EVENT_INPUT_COMPLETE	Notify whenever an input completes.
MPI_EVENT_START_OUTPUT	Notify whenever an output starts.
MPI_EVENT_OUTPUT_COMPLETE	Notify whenever an output completes.
MPI_EVENT_CARD_EVENTS	Notify whenever any card is started or completes.
MPI_EVENT_BURST_EVENTS	Notify whenever a burst starts or completes.
MPI_EVENT_MAP_EVENTS	Notify whenever a map starts or completes.
MPI_EVENT_ALL_EVENTS	Notify all card, map, and burst events.

Load and run map overview

Use the loading and running methods to load specific maps, refresh maps when the maps have changed, and run maps.

- **[LoadFile](#)**
This method reads the compiled map file (.mmc) if it has not previously been read.
- **[LoadMemory](#)**
This method processes the map code (CompiledMap) if it has not been previously processed. LoadMemory then creates and populates the hierarchy of map, card, and adapter objects.
- **[Refresh](#)**
This method updates a map instance with a new map definition following an invocation of Reload.

- **[ReloadFile](#)**
This method instructs the engine to read the compile map file specified by MapName and to update its internal map cache.
 - **[ReloadMemory](#)**
This method instructs the TX programming interface to process the compiled map (CompiledMap) and to update its internal map cache.
 - **[Run](#)**
This method executes the map defined by the map object.
 - **[Unload](#)**
This method destroys the map object, and other card and adapter objects associated with it.
-

LoadFile

This method reads the compiled map file (.mmc) if it has not previously been read.

LoadFile then creates and populates the hierarchy of map, card, and adapter objects. If the map has already been read, and is stored in the map cache, a subsequent call to LoadFile creates another instance of the map.

LoadMemory

This method processes the map code (CompiledMap) if it has not been previously processed. LoadMemory then creates and populates the hierarchy of map, card, and adapter objects.

If the map has already been read, and is stored in the map cache, a subsequent call to LoadMemory creates another instance of the map. The Identifier is used to identify the specific map; this might be the name of the map or any other identifier that the application will use.

Subsequent calls to this method with the same identifier will not introduce additional processing of the compiled map. Therefore, after the initial call to LoadMemory, the memory consumed by CompiledMap can be freed.

Refresh

This method updates a map instance with a new map definition following an invocation of Reload.

If a compiled map changes during the execution of a process, the changed map will not be picked up by the TX programming interface unless an explicit call to Refresh is made.

It is recommended that you not add or remove cards while there are existing map objects referencing the map. This could lead to unpredictable behavior. If such changes are made, all map instances referencing the map must be terminated (by invoking Unload). If the only changes to maps are to the map rules, it is valid to call Reload and Refresh to pick up the changes to the map.

ReloadFile

This method instructs the engine to read the compile map file specified by MapName and to update its internal map cache.

If any map objects currently exist that refer to this map, they will continue to use the old map definition unless they execute the Refresh method. Any map, card, or adapter overrides are preserved.

If any map objects are created after this method has been invoked, they will always use the reloaded map, rather than any previous version of the map.

ReloadMemory

This method instructs the TX programming interface to process the compiled map (CompiledMap) and to update its internal map cache.

If any map objects currently exist that refer to this map, they will continue to use the old map definition unless they execute the Refresh method. Any map, card, or adapter overrides are preserved.

Run

This method executes the map defined by the map object.

It can be invoked any number of times in sequence. Note that to run multiple instances of the same map in parallel, you must load the map multiple times (by calling LoadFile or LoadMemory).

Unload

This method destroys the map object, and other card and adapter objects associated with it.

If there are no other map instances that have this map loaded, this call will flush the map from the map cache.

C API overview

The C API enables applications to invoke maps and to develop adapters to use with IBM Transformation Extender.

- [C API interface methods overview](#)

The map, card, and adapter object properties define the properties that can be set for each object. Use the C API interface methods to GET or SET the adapter object properties.

- [Loading, running, and controlling maps](#)

There are various methods you can use to load, run, and control maps when you use APIs with IBM Transformation Extender.

- [Custom adapters](#)

The custom adapters interface makes it possible for users to develop adapters with as much functionality as those provided with the product.

C API interface methods overview

The map, card, and adapter object properties define the properties that can be set for each object. Use the C API interface methods to GET or SET the adapter object properties.

- [mpiPropertyGetBinary](#)

The C API uses different interface methods, including [mpiPropertyGetBinary](#).

- [mpiPropertyGetCount](#)

The C API uses different interface methods, including [mpiPropertyGetCount](#).

- [mpiPropertyGetInteger](#)

The C API uses different interface methods, including [mpiPropertyGetInteger](#).

- [mpiPropertyGetPtr](#)

The C API uses different interface methods, including [mpiPropertyGetPtr](#).

- [mpiPropertyGetText](#)

The C API uses different interface methods, including [mpiPropertyGetText](#).

- [mpiPropertySetBinary](#)

The C API uses different interface methods, including [mpiPropertySetBinary](#).

- [mpiPropertySetInteger](#)

The C API uses different interface methods, including [mpiPropertySetInteger](#).

- [mpiPropertySetPtr](#)

The C API uses different interface methods, including [mpiPropertySetPtr](#).

- [mpiPropertySetText](#)

The C API uses different interface methods, including [mpiPropertySetText](#).

- [mpiPropertyGetAllPropertiesXML](#)

The C API uses different interface methods, including [mpiPropertyGetAllPropertiesXML](#).

- [mpiPropertyGetPropertiesXML](#)

The C API uses different interface methods, including [mpiPropertyGetPropertiesXML](#).

- [mpiPropertySetPropertiesXML](#)

The C API uses different interface methods, including [mpiPropertySetPropertiesXML](#).

mpiPropertyGetBinary

The C API uses different interface methods, including [mpiPropertyGetBinary](#).

```
MPIRC mpiPropertyGetBinary (HMPIOBJ hObject,MPIPROP iProperty,Void **ppData,size_t *pnSize)
```

Inputs

hObject

Handle of the IBM Transformation Extender object.

iProperty

Property ID

iIndex

Index of the property.

iValue

Integer value of the property.

Outputs

ppData

Pointer to the binary item.

pnSize

Size of the item.

Returns

Success status

mpiPropertyGetCount

The C API uses different interface methods, including **mpiPropertyGetCount**.

```
MPIRC mpiPropertyGetCount (HMPIOBJ hObject,MPIPROP iProperty,int *piCount)
```

Inputs

hObject
Handle of the IBM Transformation Extender object.
iProperty
Property ID

Outputs

piCount
Count of the property values.

Returns

Success status

mpiPropertyGetInteger

The C API uses different interface methods, including **mpiPropertyGetInteger**.

```
MPIRC mpiPropertyGetInteger (HMPIOBJ hObject,MPIPROP iProperty,int iIndex,int *pValue)
```

Inputs

hObject
Handle of the IBM Transformation Extender object.
iProperty
Property ID
iIndex
Index of the property.

Outputs

pValue
Integer value of the property.

Returns

Success status

mpiPropertyGetPtr

The C API uses different interface methods, including **mpiPropertyGetPtr**.

```
MPIRC mpiPropertyGetPtr (HMPIOBJ hObject,MPIPROP iProperty,int *iIndex,const void **ppValue)
```

Inputs

hObject
Handle of the IBM Transformation Extender object.
iProperty
Property ID
iIndex
Index of the property.
FALSE. Only the properties of the object are returned.

Outputs

ppValue
Pointer to the pointer value.

Returns

Success status

mpiPropertyGetText

The C API uses different interface methods, including **mpiPropertyGetText**.

```
MPIRC mpiPropertyGetText (HMPIOBJ hObject,MPIPROP iProperty,int iIndex,const char **ppText,size_t *pnLen)
```

Inputs

hObject
Handle of the IBM Transformation Extender object.
iProperty
Property ID
iIndex
Index of the property.

Outputs

ppText
Pointer to the pointer value.
pnLen
Length of the text item.

Returns

Success status.

Note: If pnLen is returned as 0, the string is null- terminated.

mpiPropertySetBinary

The C API uses different interface methods, including **mpiPropertySetBinary**.

```
MPIRC mpiPropertySetBinary (HMPIOBJ hObject,MPIPROP iProperty,int iIndex,const void *pData,size_t nSize)
```

Inputs

hObject
Handle of the IBM Transformation Extender object.
iProperty
Property ID
iIndex
Index of the property.
FALSE. Only the properties of the object are returned.

Outputs

pData
Pointer to the pointer value.
nSize
Size of the binary item.

Returns

Success status

Note: The data passed in pData is copied by this method.

mpiPropertySetInteger

The C API uses different interface methods, including **mpiPropertySetInteger**.

```
MPIRC mpiPropertySetInteger (HMPIOBJ hObject,MPIPROP iProperty,int iIndex,int iValue)
```

Inputs

hObject
Handle of the IBM Transformation Extender object.

iProperty
Property ID

iIndex
Index of the property.

iValue
Integer value of the property.

Outputs

None

Returns

Success status

mpiPropertySetPtr

The C API uses different interface methods, including **mpiPropertySetPtr**.

This method stores a pointer value as a property of the object. The data passed in pValue is not copied by this method. The application must ensure that this memory is not freed during the life of the object.

Additionally, the application must free up this memory when the object is destroyed. If the size of the data being pointed to must be stored, a separate integer property must be used.

```
MPIRC mpiPropertySetPtr (HMPIOBJ hObject,MPIPROP iProperty,int iIndex,const void *pValue)
```

Inputs

hObject
Handle of the IBM Transformation Extender object.

iProperty
Property ID

iIndex
Index of the property.

pValue
Pointer value.

Outputs

None

Returns

Success status.

mpiPropertySetText

The C API uses different interface methods, including **mpiPropertySetText**.

```
MPIRC mpiPropertySetText (HMPIOBJ hObject,MPIPROP iProperty,int iIndex,const char *pText,size_t nLen)
```

Inputs

hObject
Object handle.

iProperty
Property ID

iIndex
Property index

Outputs

pText
Pointer to the pointer value.
nLen
Length of the text item.

Returns

Success status

If the length is specified as 0, the string passed in pText is assumed to be null-terminated. The data passed in pText is copied by this method.

mpiPropertyGetAllPropertiesXML

The C API uses different interface methods, including **mpiPropertyGetAllPropertiesXML**.

```
MPIRC mpiPropertyGetAllPropertiesXML (HMPIOBJ hObject, char *pszOutXML, size_t *pnLen, int bRecurse)
```

Inputs

hObject
Object for which the properties are being requested.
pnLen
Size of the buffer passed in pszOutXML.
bRecurse
TRUE. Properties of related objects will also be returned. Specifically:

- If it is a card object, then the properties of the card and the related adapter object are returned.
- If it is a map object, then the properties of the map, all of its associated card objects, and the related adapter objects are returned.

FALSE. Only the properties of the object are returned.

Outputs

pszOutXML
Returned XML document conforming to dtxi_properties.xsd, which contains the values of the requested properties.
pnLen
Size of the XML document returned.

Returns

Success status

mpiPropertyGetPropertiesXML

The C API uses different interface methods, including **mpiPropertyGetPropertiesXML**.

```
MPIRC mpiPropertyGetPropertiesXML (HMPIOBJ hObject, int pIDs[], int iOffset, int iCount, char *pszOutXML, size_t *pnLen);
```

Inputs

hObject
Object for which the properties are being requested.
pIDs
Integer array which lists the property IDs that are to be returned. For indexed type properties all index values are returned.
iOffset
Start offset in the array of property IDs.
iCount
Number of elements of the property ID array to use.
pnLen
Size of the buffer passed in pszOutXML.

Outputs

pszOutXML
Returned XML document conforming to dtxi_properties.xsd, which contains the values of the requested properties.
pnLen
Size of the XML document returned.

Returns

Success status

mpiPropertySetPropertiesXML

The C API uses different interface methods, including **mpiPropertySetPropertiesXML**.

```
MPIIRC mpiPropertySetPropertiesXML (HMPIOBJ hObject, const char *pszInXML)
```

Inputs

hObject

Object for which the properties are set.

pszInXML

XML document conforming to dtxp1_properties.xsd which lists the properties that are to be set. It is possible to set properties on an object and on its related objects in a single call. Specifically,

- If called on a adapter object, only adapter properties can be set.
- If called on an card object, then the properties of the card and its related adapter object can be set
- If called on an map object, then the properties of the map and its related card and adapter objects can be set.

Outputs

None

Returns

Success status.

C API map object methods overview

The following methods are supported by the C API map object.

- [mpiMapGetInputCardNumber](#)
The C API uses different map object methods, including **mpiMapGetInputCardNumber**.
- [mpiMapGetOutputCardNumber](#)
The C API uses different map object methods, including **mpiMapGetOutputCardNumber**.
- [mpiMapGetInputCardObject](#)
The C API uses different map object methods, including **mpiMapGetInputCardObject**.
- [mpiMapGetOutputCardObject](#)
The C API uses different map object methods, including **mpiMapGetOutputCardObject**.
- [mpiMapGetUniqueMapInstance](#)
The C API uses different map object methods, including **mpiMapGetUniqueMapInstance**.

mpiMapGetInputCardNumber

The C API uses different map object methods, including **mpiMapGetInputCardNumber**.

```
MPIIRC mpiMapGetInputCardNumber (HMPIMAP hMap, int *piCount)
```

Inputs

hMap

Map object

Outputs

piCount

Address of an integer variable to receive the card count.

Returns

Success status

mpiMapGetOutputCardNumber

The C API uses different map object methods, including **mpiMapGetOutputCardNumber**.

```
MPIRC mpiMapGetInputCardObject (HMPIMAP hMap,int iCard,HMPICARD *hCard)
```

Inputs

hMap
Map object

Outputs

piCount
Address of an integer variable to receive the card count.

Returns

Success status

mpiMapGetInputCardObject

The C API uses different map object methods, including **mpiMapGetInputCardObject**.

```
MPIRC mpiMapGetInputCardObject (HMPIMAP hMap,int iCard,HMPICARD *hCard)
```

Inputs

hMap
Map object
iCard
Number of the input card (starting at 1).

Outputs

hCard
Card object

Returns

Success status

mpiMapGetOutputCardObject

The C API uses different map object methods, including **mpiMapGetOutputCardObject**.

```
MPIRC mpiMapGetOutputCardObject (HMPIMAP hMap,int iCard,HMPICARD *hCard)
```

Inputs

hMap
Map object
iCard
Number of the input card (starting at 1).

Outputs

hCard
Card object

Returns

Success status

mpiMapGetUniqueMapInstance

The C API uses different map object methods, including **mpiMapGetUniqueMapInstance**.

If the map instance is not set before you run the map, IBM Transformation Extender automatically assigns a unique integer as the map instance number. However, if an API user uses IBM Transformation Extender Object Pools, the unique map instance must be assigned through the API code. This method obtains the unique map instance value and sets the MPIP_MAP_INSTANCE property with this unique value.

To call this method in C++:

```
int mpiMapGetUniqueMapInstance()
```

To call this method in Java:

```
int MMap.getUniqueMapInstance()
```

Inputs

None.

Outputs

Unique Map Instance Number

Integer representing the unique instance number of the map. This method does not assign the integer number but rather returns it. The map instance number is assigned by setting the IBM Transformation Extender map integer property MPIP_MAP_INSTANCE to the value returned by this call.

Returns

Unique integer that represents the map instance.

C API card object methods overview

The following methods are supported by the C API card object.

- [mpiCardGetAdapterObject](#)
The C API uses different card object methods, including **mpiCardGetAdapterObject**.
- [mpiCardGetAdapterType](#)
The C API uses different card object methods, including **mpiCardGetAdapterType**.
- [mpiCardGetMapObject](#)
The C API uses different card object methods, including **mpiCardGetMapObject**.
- [mpiCardOverrideAdapter](#)
The C API uses different card object methods, including **mpiCardOverrideAdapter**.

mpiCardGetAdapterObject

The C API uses different card object methods, including **mpiCardGetAdapterObject**.

```
MPIIRC mpiCardGetAdapterObject (HMPICARD hCard, HMPIADAPT *hAdapter)
```

Inputs

hCard
Card object

Outputs

hAdapter
Adapter object

Returns

Success status

mpiCardGetAdapterType

The C API uses different card object methods, including **mpiCardGetAdapterType**.

```
MPIIRC mpiCardGetAdapterType (HMPICARD hCard, int *piAdapterType)
```

Inputs

hCard
Card object

Outputs

piAdapterType
Type of adapter.

Returns

Success status

Some types are not listed in this file. These are listed below, along with their reserved values:

Type	Value
Buffer	4
Stream	31

mpiCardGetMapObject

The C API uses different card object methods, including **mpiCardGetMapObject**.

```
MPIIRC mpiCardGetMapObject (HMPICARD hCard, HMPIMAP *hMap)
```

Inputs

hCard
Card object

Outputs

hmap
Map object

Returns

Success status

mpiCardOverrideAdapter

The C API uses different card object methods, including **mpiCardOverrideAdapter**.

```
MPIIRC mpiCardOverrideAdapter (HMPICARD hCard, int iAdapterType, char *pszAlias)
```

Inputs

hCard
Card object

Outputs

iAdapterType
Numeric type of the adapter.
pszAlias
Alias name for the adapter.

Returns

Success status

Adapter object methods

The following methods are supported by the adapter objects.

- [mpiAdaptGetCardObject](#)
The C API uses different adapter object methods, including **mpiAdaptGetCardObject**.
- [mpiAdaptGetInputStreamObject](#)
The C API uses different adapter object methods, including **mpiAdaptGetInputStreamObject**.
- [mpiAdaptGetOutputStreamObject](#)
The C API uses different adapter object methods, including **mpiAdaptGetOutputStreamObject**.

mpiAdaptGetCardObject

The C API uses different adapter object methods, including **mpiAdaptGetCardObject**.

```
MPIRC mpiAdaptGetCardObject (HMPIADAPT hAdapter,HMPICARD *hCard)
```

Inputs

hAdapter
Adapter object

Outputs

hCard
Card object

Returns

Success status

mpiAdaptGetInputStreamObject

The C API uses different adapter object methods, including **mpiAdaptGetInputStreamObject**.

```
MPIRC mpiAdaptGetInputStreamObject (HMPIADAPT hAdapter,HMPISTREAM *phStream)
```

Inputs

hAdapter
Adapter object

Outputs

phStream
Stream object

Returns

Success status

mpiAdaptGetOutputStreamObject

The C API uses different adapter object methods, including **mpiAdaptGetOutputStreamObject**.

```
MPIRC mpiAdaptGetOutputStreamObject (HMPIADAPT hAdapter,HMPISTREAM *phStream)
```

Inputs

hAdapter
Adapter object

Outputs

phStream
Stream object

Returns

Success status

Loading, running, and controlling maps

There are various methods you can use to load, run, and control maps when you use APIs with IBM Transformation Extender.

- [Loading and running maps](#)
Use the following methods to load specific maps, refresh maps when the maps have changed, and run maps.
- [Controlling and monitoring maps](#)
These methods allow map execution to be controlled from another thread and to report map progress.
- [Initialization and termination methods](#)
The following methods initialize the environment required for running maps and clean up the environment after the map run is complete.
- [Callbacks](#)
An API application uses callback to receive logging messages on a per-map basis or for global logging based on WebSphere Transformation Extender initialization.
- [C API examples](#)
The examples included with the C API are located in the following directory: *install_dir\examples\dk\dtspi\c*

Loading and running maps

Use the following methods to load specific maps, refresh maps when the maps have changed, and run maps.

- [mpiMapLoadFile](#)
The C API uses different methods to load, run, and control maps, including **mpiMapLoadFile**.
- [mpiMapLoadMemory](#)
The C API uses different methods to load, run, and control maps, including **mpiMapLoadMemory**.
- [mpiMapRefresh](#)
The C API uses different methods to load, run, and control maps, including **mpiMapLoadRefresh**.
- [mpiMapReloadFile](#)
The C API uses different methods to load, run, and control maps, including **mpiMapReloadFile**.
- [mpiMapReloadMemory](#)
The C API uses different methods to load, run, and control maps, including **mpiMapReloadMemory**.
- [mpiMapRun](#)
The C API uses different methods to load, run, and control maps, including **mpiMapRun**.
- [mpiMapUnload](#)
The C API uses different methods to load, run, and control maps, including **mpiMapUnload**.

mpiMapLoadFile

The C API uses different methods to load, run, and control maps, including **mpiMapLoadFile**.

```
MPIIRC mpiMapLoadFile (HMPIMAP *hMap,const char *lpszMapName)
```

Inputs

lpszMapName
Name of the compiled map file (.mmc).

Outputs

hMap
Map object

Returns

Success status

mpiMapLoadMemory

The C API uses different methods to load, run, and control maps, including **mpiMapLoadMemory**.

```
MPIIRC mpiMapLoadMemory (HMPIMAP *hMap,const char *lpszIdentifier,const char *lpszWorkingDirectory,const char *pcCompiledMap,int iSize)
```

Inputs

lpszIdentifier
Text string used to identify the compiled map. The string cannot exceed 32 characters.
lpszWorkingDirectory
Directory that should be used as the working directory for the map.
pcCompiledMap
Bytes that comprise a compiled map.
iSize
Number of bytes pointed to pcCompiledMap.

Outputs

hMap
Map object

Returns

Success status

```
HMPIMAP hMap1;
HMPIMAP hMap2;
char *pMapData =
MY_COMPILED_MAP_BYTES;
int iSize =
sizeof(MY_COMPILED_MAP_BYTES);
mpiMapLoadMemory (&hMap1, "MyMap", pMapData, iSize);
mpiMapLoadMemory (&hMap2, "MyMap", NULL, 0);
```

mpiMapRefresh

The C API uses different methods to load, run, and control maps, including **mpiMapLoadRefresh**.

```
MPIRC mpiMapRefresh (HMPIMAP hMap)
```

Inputs

hMap
Map object

Outputs

None

Returns

Success status

mpiMapReloadFile

The C API uses different methods to load, run, and control maps, including **mpiMapReloadFile**.

```
MPIRC mpiMapReloadFile (const char *lpszMapName)
```

Inputs

lpszMapName
Name of the compiled map file (.mmc).

Outputs

None

Returns

Success status

The following code demonstrates the usage of mpiMapReloadFile:

```
HMPIMAP hMap1;
HMPIMAP hMap2;
mpiMapLoadFile (&hMap1, "fred.mmc");
mpiMapLoadFile (&hMap2, "fred.mmc");
mpiMapRun (&hMap1);
mpiMapRun (&hMap2);
/* Now the program detects that the map file fred.mmc has changed */
mpiMapReloadFile ("fred.mmc");
/* hMap1 will pick up the updates and run with the 'new' map */
mpiMapRefresh (hMap1);
mpiMapRun (&hMap1);
/* HMAP2 did not refresh will run with the 'old' map */
mpiMapRun (&hMap2);
```

mpiMapReloadMemory

The C API uses different methods to load, run, and control maps, including **mpiMapReloadMemory**.

```
MPIIRC mpiMapReloadMemory (const char *lpszIdentifier, const char *WorkingDirectory, const char *pcCompiledMap, int iSize)
```

Inputs

lpszIdentifier
Text string used to identify the compiled map.
WorkingDirectory
Directory from which relative paths are determined.
pcCompiledMap
Bytes that comprise a compiled map.
iSize
Number of bytes pointed to pcCompiledMap.

Outputs

None

Returns

Success status

mpiMapRun

The C API uses different methods to load, run, and control maps, including **mpiMapRun**.

```
MPIIRC mpiMapRun (HMPIMAP hMap)
```

Inputs

hMap
Map object

Outputs

None

Returns

Success status

mpiMapUnload

The C API uses different methods to load, run, and control maps, including **mpiMapUnload**.

```
MPIIRC mpiMapUnload (HMPIMAP *hMap)
```

Inputs

hMap
Map object

Outputs

None

Returns

Success status

Controlling and monitoring maps

These methods allow map execution to be controlled from another thread and to report map progress.

- [**mpiMapAbort**](#)
The C API uses different methods to control and monitor maps, including **mpiMapAbort**.
- [**mpiMapContinue**](#)
The C API uses different methods to control and monitor maps, including **mpiMapContinue**.
- [**mpiMapPause**](#)
The C API uses different methods to control and monitor maps, including **mpiMapPause**.
- [**mpiMapRegisterStatusMethod**](#)
The C API uses different methods to control and monitor maps, including **mpiMapRegisterStatusMethod**.

mpiMapAbort

The C API uses different methods to control and monitor maps, including **mpiMapAbort**.

```
MPIRC mpiMapAbort (HMPIMAP hMap)
```

Inputs

hMap
Map object

Outputs

None

Returns

Success status

mpiMapContinue

The C API uses different methods to control and monitor maps, including **mpiMapContinue**.

```
MPIRC mpiMapContinue (HMPIMAP hMap)
```

Inputs

hMap
Map object

Outputs

None

Returns

Success status

mpiMapPause

The C API uses different methods to control and monitor maps, including **mpiMapPause**.

```
MPIRC mpiMapPause (HMPIMAP hMap)
```

Inputs

hMap
Map object

Outputs

None

Returns

Success status

mpiMapRegisterStatusMethod

The C API uses different methods to control and monitor maps, including **mpiMapRegisterStatusMethod**.

```
MPIRC mpiMapRegisterStatusMethod  
(HMPIMAP hMap MPICTRLMTD lpfnControl, int iNotifyWhen)
```

Inputs

hMap
 Map object
lpfnControl
 Address of the user-provided status method
iNotifyWhen
 Bitmask indicating the events of which the registrar wished to be notified

Outputs

None

Returns

Success status

Initialization and termination methods

The following methods initialize the environment required for running maps and clean up the environment after the map run is complete.

- [mpiInitAPI](#)
The C API uses different methods to initialize and terminate map execution, including **mpiInitAPI**.
- [mpiInitAPIex](#)
The C API uses different methods to initialize and terminate map execution, including **mpiInitAPIex**.
- [mpiTermAPI](#)
The C API uses different methods to initialize and terminate map execution, including **mpiTermAPI**.

mpiInitAPI

The C API uses different methods to initialize and terminate map execution, including **mpiInitAPI**.

This method initializes the IBM Transformation Extender environment. It should be called prior to any other call and only once per process. The lpszConfigFile can be passed in as NULL as long as name resolution is not required.

```
MPIRC mpiInitAPI (char *lpszConfigFile)
```

Inputs

lpszConfigFile
 Name of the configuration file used to determine the resource files used.

Outputs

None

Returns

Success status

Related reference

- [mpiInitAPIex](#)

mpiInitAPIex

The C API uses different methods to initialize and terminate map execution, including **mpiInitAPIex**.

This method initializes the WebSphere Transformation Extender environment. Call this method only once per process, before any other call. Both the Resource Registry configuration (.mrc) file and a log function callback can be passed to the **mpiInitAPIex** method.

```
MPIIRC mpiInitAPIex(const char *lpszResourceConfigFile, MPI_LOGFNCALLBACK logFnCB);
```

Related reference

- [mpiInitAPI](#)

mpiTermAPI

The C API uses different methods to initialize and terminate map execution, including [mpiTermAPI](#).

This method cleans up the IBM Transformation Extender environment. This method can be called after any other call, but only called one time for each process.

```
MPIIRC mpiTermAPI (void)
```

Inputs

None

Outputs

None

Returns

Success status

Callbacks

An API application uses callback to receive logging messages on a per-map basis or for global logging based on WebSphere Transformation Extender initialization.

- [MPI_TRACE_CALLBACK_PROC](#)
An API application can use **MPI_TRACE_CALLBACK_PROC** to receive logging messages such as I/O operations and Resource Manager operations on a per-map instance basis.
- [MPI_LOGFNCALLBACK](#)
An API application can use **MPI_LOGFNCALLBACK** as a global process API callback. **MPI_LOGFNCALLBACK** is registered during the Transformation Extender API initialization, and includes Connection Manager and Resource Registry logging. An API application can use both **MPI_TRACE_CALLBACK_PROC** and **MPI_LOGFNCALLBACK** in order to log WebSphere Transformation Extender I/O, Resource Manager, Connection Manager operations and Resource Registry messages to a central logging facility.

MPI_TRACE_CALLBACK_PROC

An API application can use **MPI_TRACE_CALLBACK_PROC** to receive logging messages such as I/O operations and Resource Manager operations on a per-map instance basis.

Related reference

- [MPI_LOGFNCALLBACK](#)
- [mpiInitAPI](#)
- [mpiInitAPIex](#)

MPI_LOGFNCALLBACK

An API application can use **MPI_LOGFNCALLBACK** as a global process API callback. **MPI_LOGFNCALLBACK** is registered during the Transformation Extender API initialization, and includes Connection Manager and Resource Registry logging. An API application can use both **MPI_TRACE_CALLBACK_PROC** and **MPI_LOGFNCALLBACK** in order to log WebSphere Transformation Extender I/O, Resource Manager, Connection Manager operations and Resource Registry messages to a central logging facility.

Related reference

- [MPI_TRACE_CALLBACK_PROC](#)
- [mpiInitAPI](#)
- [mpiInitAPIex](#)

C API examples

The examples included with the C API are located in the following directory: *install_dir\examples\dk\dtspi\c*

- [Running a single map](#)

This example shows the simplest way to load and run a map. After the map is loaded, it can be run any number of times.

- [Running multiple instances of a map in parallel](#)

This example shows how to create multiple instances of map objects and how to run these map instances in parallel.

- [Overriding a card in a map](#)

This example shows how to set properties of card objects to override the settings of the compiled map.

- [Using a user-provided status method](#)

This example shows how to provide a status method to the C API and how to receive notification of the progress of a map execution.

- [Using streams to override inputs and outputs](#)

This example shows how to pass data to an input card of a map and how to get data from an output card of a map (the sending of a single buffer to the API).

- [Loading a map from a byte array](#)

This example shows how to load a map from a byte array.

- [Getting and setting properties](#)

This example shows how to GET and SET properties.

- [Developing a custom adapter](#)

To develop a custom adapter, please see the **filesamp.c** example, which demonstrates all of the interfaces that are described in this documentation. The example is well-commented and should be consulted to aid in the understanding of the adapter interface.

Running a single map

This example shows the simplest way to load and run a map. After the map is loaded, it can be run any number of times.

```
#include "dtspi.h"
#define CHECK_ERR(rc) if(MPIRC_FAILED(rc)) { printf("Last error is: %s\n",mpiErrorGetText(rc)); return -1; }

int main()
{
const char      *szMsg;
int             iRC;
MPIRC          rc;
HMPIMAP        hMap;

rc = mpiInitAPI(NULL);
CHECK_ERR(rc);

rc = mpiMapLoadFile (&hMap, "test1.mmc");
CHECK_ERR(rc);

rc = mpiMapRun (hMap);
CHECK_ERR(rc);

rc = mpiPropertyGetText (hMap, MPIP_OBJECT_ERROR_MSG, 0, &szMsg, NULL);
CHECK_ERR(rc);

rc = mpiPropertyGetInteger (hMap, MPIP_OBJECT_ERROR_CODE, 0, &iRC);
CHECK_ERR(rc);
printf("Map status: %s (%d)\n", szMsg, iRC);

rc = mpiMapUnload (hMap);
CHECK_ERR(rc);

rc = mpiTermAPI();
CHECK_ERR(rc);
return 0;
}
```

Running multiple instances of a map in parallel

This example shows how to create multiple instances of map objects and how to run these map instances in parallel.

```
#include "dtspi.h"

#ifndef WIN32
#include <windows.h>
#include <process.h>

#define CRITICAL_SECTION CRITICAL_SECTION
#define INITIALIZE_CRITICAL_SECTION(x) InitializeCriticalSection(x)
#define ENTER_CRITICAL_SECTION(x) EnterCriticalSection(x)
#define LEAVE_CRITICAL_SECTION(x) LeaveCriticalSection(x)
#define DELETE_CRITICAL_SECTION(x) DeleteCriticalSection(x)
#define CREATE_THREAD(a,b) _beginthread((a),0,(b))
#define SLEEP(x) Sleep(x)

void RunMap(HMPIMAP hMap);

#else /* UNIX defines */
#include <pthread.h>
#include <unistd.h>
#include <stropts.h>
#include <poll.h>
```

```

pthread_t tid;

#define CRITICAL_SECTION pthread_mutex_t
#define INITIALIZE_CRITICAL_SECTION(x) pthread_mutex_init((x),NULL)
#define ENTER_CRITICAL_SECTION(x) pthread_mutex_lock(x)
#define LEAVE_CRITICAL_SECTION(x) pthread_mutex_unlock(x)
#define DELETE_CRITICAL_SECTION(x) pthread_mutex_destroy(x)
#define CREATE_THREAD(a,b) pthread_create(&tid,NULL,
(void* (*)(void*)) (a),(void*) (b))
#define SLEEP(x) poll(NULL,NULL,(x))

void* RunMap(HMPIMAP hMap);
#endif
CRITICAL_SECTION crit_sec;
int num_maps;

#define CHECK_ERR(rc) if(MPIRC_FAILED(rc))
{ printf("Last error is: %s\n",mpiErrorGetText(rc)); return -1; }

int main()
{
const char *szMsg;
int iRC;
HMPIMAP hMap1;
HMPIMAP hMap2;
MPIRC rc;

INITIALIZE_CRITICAL_SECTION( &crit_sec);

rc = mpiInitAPI(NULL);
CHECK_ERR(rc);

rc = mpiMapLoadFile (&hMap1, "test2.mmc");
CHECK_ERR(rc);
rc = mpiMapLoadFile (&hMap2, "test2.mmc");
/* This 2nd call will not read the MMC file.*/
CHECK_ERR(rc);

num_maps = 1;
CREATE_THREAD(RunMap,hMap1);

ENTER_CRITICAL_SECTION(&crit_sec);
num_maps++;
LEAVE_CRITICAL_SECTION(&crit_sec);

CREATE_THREAD(RunMap,hMap2);

/* Wait for map threads to complete */
while(num_maps)
SLEEP(100);

rc = mpiPropertyGetText (hMap1, MPIP_OBJECT_ERROR_MSG, 0, &szMsg, NULL);
CHECK_ERR(rc);
rc = mpiPropertyGetInteger (hMap1, MPIP_OBJECT_ERROR_CODE, 0, &iRC);
CHECK_ERR(rc);
printf("Map status: %s (%d)\n", szMsg, iRC);

rc = mpiPropertyGetText (hMap2, MPIP_OBJECT_ERROR_MSG, 0, &szMsg, NULL);
CHECK_ERR(rc);
rc = mpiPropertyGetInteger (hMap2, MPIP_OBJECT_ERROR_CODE, 0, &iRC);
CHECK_ERR(rc);
printf("Map status: %s (%d)\n", szMsg, iRC);

rc = mpiMapUnload (hMap1);
CHECK_ERR(rc);
rc = mpiMapUnload (hMap2);
CHECK_ERR(rc);

rc = mpiTermAPI();
CHECK_ERR(rc);

DELETE_CRITICAL_SECTION(&crit_sec);
return 0;
}

#endif WIN32
void RunMap(HMPIMAP hMap)
#else /* UNIX version of the function */
void* RunMap(HMPIMAP hMap)
#endif
{
MPIRC rc;

rc = mpiMapRun (hMap);

ENTER_CRITICAL_SECTION(&crit_sec);
num_maps--;
LEAVE_CRITICAL_SECTION(&crit_sec);

#endif WIN32
return;
#else
return NULL;
#endif
}

```

```
}
```

Overriding a card in a map

This example shows how to set properties of card objects to override the settings of the compiled map.

Note: This example is written to be used with the GZIP adapter ,which is not available on all platforms. If necessary, override the GZIP adapter type in this example with a substitute adapter to correctly run the example.

```
#include "dtxpi.h"

#define CHECK_ERR(rc) if(MPIRC_FAILED(rc))  
{ printf("Last error is: %s\n",mpiErrorGetText(rc));  
return -1;}

int main()  
{  
const char    *szMsg;  
int          iRC;  
MPIRC rc;  
HMPIMAP      hMap;  
HMPLICARD    hInputCard1;  
HMPIADAPT    hAdapter;

rc = mpiInitAPI(NULL);  
CHECK_ERR(rc);
rc = mpiMapLoadFile (&hMap, "test3.mmc");
CHECK_ERR(rc);

/* Turn on burst and summary execution audit on the map instance */
rc = mpiPropertySetInteger (hMap, MPIP_MAP_AUDIT_SWITCH, 0,
    MPI_SWITCH_ON);
CHECK_ERR(rc);
rc = mpiPropertySetInteger (hMap, MPIP_MAP_AUDIT_BURST_EXECUTION, 0,
    MPI_SWITCH_ON);
CHECK_ERR(rc);
rc = mpiPropertySetInteger (hMap, MPIP_MAP_AUDIT_SUMMARY_EXECUTION, 0,
    MPI_SWITCH_ON);
CHECK_ERR(rc);

/* Get the adapter object handle */
rc = mpiMapGetInputCardObject (hMap, 1, &hInputCard1);
CHECK_ERR(rc);
rc = mpiCardOverrideAdapter(hInputCard1,"GZIP",0);
CHECK_ERR(rc);
rc = mpiCardGetAdapterObject (hInputCard1, &hAdapter);
CHECK_ERR(rc);

/* set a command line for the adapter */
rc = mpiPropertySetText(hAdapter, MPIP_ADAPTER_COMMANDLINE, 0,
    "-FILE Input.gz",
0);
CHECK_ERR(rc);

rc = mpiMapRun (hMap);
CHECK_ERR(rc);

rc = mpiPropertyGetText (hMap, MPIP_OBJECT_ERROR_MSG, 0, &szMsg, NULL);
CHECK_ERR(rc);
rc = mpiPropertyGetInteger (hMap, MPIP_OBJECT_ERROR_CODE, 0, &iRC);
CHECK_ERR(rc);
printf("Map status: %s (%d)\n", szMsg, iRC);

rc = mpiMapUnload (hMap);
CHECK_ERR(rc);
rc = mpiTermAPI();
CHECK_ERR(rc);
return 0;
}
```

Using a user-provided status method

This example shows how to provide a status method to the C API and how to receive notification of the progress of a map execution.

```
#include "dtxpi.h"  
#define CHECK_ERR(rc) if(MPIRC_FAILED(rc))  
{ printf("Last error is: %s\n",mpiErrorGetText(rc)); return -1;}

void StatusMethod(HMPIMAP hMap, int iEventID, int iBurstNum, int iCardNum)  
{  
    switch(iEventID)  
    {  
        case MPI_EVENT_START_INPUT:  
            printf("Event received = MPI_EVENT_START_INPUT \n");  
            break;  
        case MPI_EVENT_INPUT_COMPLETE:
```

```

        printf("Event received = MPI_EVENT_INPUT_COMPLETE \n");
        break;
    case MPI_EVENT_START_OUTPUT:
        printf("Event received = MPI_EVENT_START_OUTPUT \n");
        break;
    case MPI_EVENT_OUTPUT_COMPLETE:
        printf("Event received = MPI_EVENT_OUTPUT_COMPLETE \n");
        break;
    case MPI_EVENT_START_BURST:
        printf("Event received = MPI_EVENT_START_BURST \n");
        break;
    case MPI_EVENT_BURST_COMPLETE:
        printf("Event received = MPI_EVENT_BURST_COMPLETE \n");
        break;
    case MPI_EVENT_START_MAP:
        printf("Event received = MPI_EVENT_START_MAP \n");
        break;
    case MPI_EVENT_MAP_COMPLETE:
        printf("Event received = MPI_EVENT_MAP_COMPLETE \n");
        break;
    default:
        printf("Unknown event !!!");
    }
}

int main()
{
    const char      *szMsg;
    int             iRC;
    size_t          tlen;
    MPIRC          rc;
    HMPIMAP         hMap;

    rc = mpiInitAPI(NULL);
    CHECK_ERR(rc);
    rc = mpiMapLoadFile (&hMap, "test4.mmc");
    CHECK_ERR(rc);

    rc = mpiMapRegisterStatusMethod (hMap, StatusMethod, MPI_EVENT_ALL);
    CHECK_ERR(rc);

    rc = mpiMapRun (hMap);
    CHECK_ERR(rc);

    rc = mpiPropertyGetText (hMap, MPIP_OBJECT_ERROR_MSG, 0, &szMsg, &tlen);
    CHECK_ERR(rc);
    rc = mpiPropertyGetInteger (hMap, MPIP_OBJECT_ERROR_CODE, 0, &iRC);
    CHECK_ERR(rc);

    printf("Map status: %s (%d)\n";
    rc = mpiMapUnload (hMap);
    CHECK_ERR(rc);
    rc = mpiTermAPI();
    CHECK_ERR(rc);
    return 0;
}

```

Using streams to override inputs and outputs

This example shows how to pass data to an input card of a map and how to get data from an output card of a map (the sending of a single buffer to the API).

If the data is received in a number of smaller pieces, it is acceptable, and indeed efficient, to write the data to the stream in small pieces.

```

#include "dtxpi.h"

#define CHECK_ERR(rc) if(MPIRC_FAILED(rc)) \
{ printf("Last error is: %s\n",mpiErrorGetText(rc)); return -1; }

int main()
{
const char      *szMsg;
char szInputBuffer[] = "This is my input data";
int             iRC;
int             bIsEnd;
void            *pData;
size_t          nSizeOfData;
size_t          nLen;
HMPIMAP         hMap;
HMPIADAPT       hAdapter;
HMPICARD        hCard;
HMPISTREAM      hStream;
HMPISTRAMPAGE   hPage;
MPIRC          rc;

rc = mpiInitAPI(NULL);
CHECK_ERR(rc);
rc = mpiMapLoadFile (&hMap, "test5.mmc");
CHECK_ERR(rc);

/* Get the adapter object handle for input card #1*/
rc = mpiMapGetInputCardObject (hMap, 1, &hCard);

```

```

CHECK_ERR(rc);

/* Override the adapter in input card #1 to be a stream */
rc = mpiCardOverrideAdapter(hCard,NULL, MPI_ADAPTER_TYPE_STREAM);
CHECK_ERR(rc);

rc = mpiCardGetAdapterObject (hCard, &hAdapter);
CHECK_ERR(rc);
/* Get the handle to the stream object */
rc = mpiPropertyGetObject (hAdapter, MPI_ADAPTER_DATA_FROM_ADAPTER,
 0, &hStream);
CHECK_ERR(rc);

/* Send a single large page */
rc = mpiStreamWrite (hStream, szInputBuffer, strlen(szInputBuffer));
CHECK_ERR(rc);

/* Get the adapter object handle for output card #2*/
rc = mpiMapGetOutputCardObject (hMap, 2, &hCard);
CHECK_ERR(rc);

/* Override the adapter in output card #2 to be a stream */
rc = mpiCardOverrideAdapter(hCard,NULL, MPI_ADAPTER_TYPE_STREAM);
CHECK_ERR(rc);

rc = mpiMapRun (hMap);
CHECK_ERR(rc);

rc = mpiPropertyGetText(hMap, MPI_OBJECT_ERROR_MSG, 0,
  &szMsg, &nLen);
CHECK_ERR(rc);
rc = mpiPropertyGetInteger (hMap, MPI_OBJECT_ERROR_CODE, 0, &iRC);
CHECK_ERR(rc);
printf("Map status: %s (%d)\n", szMsg, iRC);

/* Get the adapter object handle for output card #2*/
rc = mpiCardGetAdapterObject (hCard, &hAdapter);
CHECK_ERR(rc);
/* Get the handle to the stream object */
rc = mpiPropertyGetObject (hAdapter, MPI_ADAPTER_DATA_TO_ADAPTER, 0,
  &hStream);
CHECK_ERR(rc);

/* Get the data in pieces from the stream */
mpiStreamSeek(hStream,0,SEEK_SET);
while (1)
{
rc = mpiStreamIsEnd (hStream, &bIsEnd);
CHECK_ERR(rc);
/*Clean and Break*/
if (bIsEnd)
{
rc = mpiStreamSetSize(hStream, 0);
break;
}
rc = mpiStreamReadPage (hStream, &hPage);
CHECK_ERR(rc);
rc = mpiStreamPageGetInfo (hPage, &pData, &nSizeOfData);
CHECK_ERR(rc);
fwrite(pData,1,nSizeOfData,stdout);
/*           DoSomethingWithData (pData, nSizeOfData); */
}

rc = mpiMapUnload (hMap);

CHECK_ERR(rc);
rc = mpiTermAPI();
CHECK_ERR(rc);
return 0;

```

Loading a map from a byte array

This example shows how to load a map from a byte array.

```

#include "dtxpi.h"

#define CHECK_ERR(rc) if(MPIRC_FAILED(rc)) { printf \
("Last error is: %s\n",mpiErrorGetText(rc)); return -1; }

int main()
{
  const char      *szMsg;
  char            *szMMCBuffer;
  char            *szInputBuffer;
  int             iRC;
  int             bIsEnd;
  void            *pData;
  size_t          nSizeOfData;
  size_t          nLen;
  HMPIMAP         hMap;
  HMPIADAPT       hAdapter;
  HMPICARD        hCard;

```

```

HMPIRSTREAM      hStream;
HMPIRSTREAMPAGE hPage;
MPIRC            rc;
FILE             *fp;

rc = mpiInitAPI(NULL);
CHECK_ERR(rc);

/* Load a local map file */
fp = fopen("test6.mmc", "rb");
fseek( fp, 0, SEEK_END );
nSizeOfData = ftell( fp );
szMMCBuffer = (char *)malloc( nSizeOfData );
fseek( fp, 0, SEEK_SET );
fread( szMMCBuffer, sizeof(char), nSizeOfData, fp );
fclose( fp );

rc = mpiMapLoadMemory (&hMap, "test6", NULL, szMMCBuffer,
nSizeOfData);
CHECK_ERR(rc);

/* Get the adapter object handle for input card #1*/
rc = mpiMapGetInputCardObject (hMap, 1, &hCard);
CHECK_ERR(rc);

/* Override the adapter in input card #1 to be a stream */
rc = mpiCardOverrideAdapter(hCard,NULL, MPI_ADAPTER_TYPE_STREAM);
CHECK_ERR(rc);

/* Read the local data file */
fp = fopen("Input.txt", "r");
fseek( fp, 0, SEEK_END );
nSizeOfData = ftell( fp ); szInputBuffer = (char *)malloc( nSizeOfData );
fseek( fp, 0, SEEK_SET );
fread( szInputBuffer, sizeof(char), nSizeOfData, fp );
fclose( fp );

/* Get the handle to the adapter object */
rc = mpiCardGetAdapterObject (hCard, &hAdapter);
CHECK_ERR(rc);

/* Get the handle to the stream object */
rc = mpiPropertyGetObject (hAdapter, MPIP_ADAPTER_DATA_FROM_ADAPT,
0, &hStream);
CHECK_ERR(rc);

/* Send a single large page */
rc = mpiStreamWrite (hStream, szInputBuffer, strlen(szInputBuffer));
CHECK_ERR(rc);

/* Get the adapter object handle for output card #1*/
rc = mpiMapGetOutputCardObject (hMap, 1, &hCard);
CHECK_ERR(rc);

/* Override the adapter in output card #1 to be a stream */
rc = mpiCardOverrideAdapter(hCard,NULL, MPI_ADAPTER_TYPE_STREAM);
CHECK_ERR(rc);

rc = mpiMapRun (hMap);
CHECK_ERR(rc);

rc = mpiPropertyGetText(hMap, MPIP_OBJECT_ERROR_MSG, 0, &szMsg,&nLen);
CHECK_ERR(rc);
rc = mpiPropertyGetInteger (hMap, MPIP_OBJECT_ERROR_CODE, 0, &iRC);
CHECK_ERR(rc);
printf("Map status: %s (%d)\n", szMsg, iRC);

/* Get the adapter object handle for output card #1*/
rc = mpiCardGetAdapterObject (hCard, &hAdapter);
CHECK_ERR(rc);
/* Get the handle to the stream object */
rc = mpiPropertyGetObject (hAdapter, MPIP_ADAPTER_DATA_TO_ADAPT,
0, &hStream);
CHECK_ERR(rc);

/* Get the data in pieces from the stream */
mpiStreamSeek(hStream,0,SEEK_SET);
while (1)
{
  rc = mpiStreamIsEnd (hStream, &bIsEnd);
  CHECK_ERR(rc);
  /*Clean and Break*/
  if (bIsEnd)
  {
    rc = mpiStreamSetSize(hStream, 0);
    break;
  }
  rc = mpiStreamReadPage (hStream, &hPage);
  CHECK_ERR(rc);
  rc = mpiStreamPageGetInfo (hPage, &pData, &nSizeOfData);
  CHECK_ERR(rc);
  write(pData,1,nSizeOfData,stdout);

  /* DoSomethingWithData (pData, nSizeOfData); */
  /*
  */
}

```

```

rc = mpiMapUnload (hMap);
CHECK_ERR(rc);
rc = mpiTermAPI();
CHECK_ERR(rc);
return 0;
}

```

Getting and setting properties

This example shows how to GET and SET properties.

```

#include "dtxpi.h"

#define MAX_BUFFER_LEN 131072
#define CHECK_ERR(rc) if(MPIRC_FAILED(rc)) \
{ printf("Last error is: %s\n",mpiErrorGetText(rc)); \
return -1; }

int main()
{
    const char      *szMsg;
    const char      *szCommandLine;
    char            szOutXML[MAX_BUFFER_LEN];
    char            buf[4];
    size_t          iLen;
    int             ids[1];
    int             iRC;
    MPIRC          rc;
    HMPIMAP         hMap;
    HMPICARD        hCard;
    HMPIADAPT       hAdapter;

    rc = mpiInitAPI(NULL);
    CHECK_ERR(rc);

    rc = mpiMapLoadFile (&hMap, "test7.mmc");
    CHECK_ERR(rc);

    /* Get the adapter object handle for input card #1*/
    rc = mpiMapGetInputCardObject (hMap, 1, &hCard);
    CHECK_ERR(rc);

    /* Get the handle to the adapter object */
    rc = mpiCardGetAdapterObject (hCard, &hAdapter);
    CHECK_ERR(rc);

    /* Get the adapter command line in raw format */
    rc = mpiPropertyGetText (hAdapter, MPIP_ADAPTER_COMMANDLINE, 0, ,
                           &szCommandLine &iLen);

    CHECK_ERR(rc);
    printf("The adapter command line is....\n%s\n\n", szCommandLine);

    /* Get it in XML format */
    iLen = sizeof(szOutXML);
    ids[0] = MPIP_ADAPTER_COMMANDLINE;
    rc = mpiPropertyGetPropertiesXML(hAdapter, ids, 0, 1, &szOutXML[0], &iLen);
    CHECK_ERR(rc);
    printf("The adapter command line in XML is....\n%s\n\n", szOutXML);

    /* Modify the command line using XML */

    memset(szOutXML, 0, iLen);
    strcat(szOutXML, "<Adapter>");
    strcat(szOutXML, "<CommandLine id=\"\">");
    sprintf(buf, "%d", MPIP_ADAPTER_COMMANDLINE);
    strcat(szOutXML, buf);
    strcat(szOutXML, "\" type=\"text\"");
    strcat(szOutXML, "Input2.txt");
    strcat(szOutXML, "</CommandLine>");
    strcat(szOutXML, "/Adapter>");
    rc = mpiPropertySetPropertiesXML(hAdapter, szOutXML);
    CHECK_ERR(rc);

    /* Print all of the Adapter's properties in XML format */
    memset(szOutXML, 0, iLen);
    rc = mpiProperty GetAllPropertiesXML(hAdapter, &szOutXML[0], &iLen, 0);
    CHECK_ERR(rc);
    printf("The modified, complete set of adapter properties is....
\n%s\n\n", szOutXML);

    /* Clean up */
    rc = mpiMapUnload (hMap);
    CHECK_ERR(rc);
    rc = mpiTermAPI();
    CHECK_ERR(rc);
    return 0;
}

```

Developing a custom adapter

To develop a custom adapter, please see the **filesamp.c** example, which demonstrates all of the interfaces that are described in this documentation. The example is well-commented and should be consulted to aid in the understanding of the adapter interface.

Custom adapters

The custom adapters interface makes it possible for users to develop adapters with as much functionality as those provided with the product.

- [Developing custom adapters](#)

The custom adapters interface makes it possible for users to develop adapters with as much functionality as those provided with the product.

- [Adapter registration and the library virtual table](#)

The first step required to develop an adapter is to add the adapter to the adapters.xml registration file.

- [Initializing the adapter](#)

InitializeLibrary provides an entry point by which any global structures, critical sections, or lists that are required by the adapter can be initialized.

- [Point to adapter methods](#)

After your adapter is loaded, you call the virtual table (vtable) structure to obtain pointers to the methods.

- [Adapter registration file](#)

The adapters.xml registration file contains a single tag for each adapter, including a number of attributes to provide information about the adapter.

- [Additional adapter methods](#)

There are a number of additional methods that are part of the Transformation Extender Programming Interface library that are of particular relevance to adapter developers.

- [Passing data to and from maps](#)

Data is passed between adapters and the engine through stream objects. From the adapter perspective, there is no need to create these stream objects; they are already created when either the Get or Put method is called.

- [Listener scenario overview](#)

These are listener scenario examples that demonstrate the flow of calls for input events.

- [Threads, connections, and transactions overview](#)

It is important to understand how threads, connections, and transactions interact.

Developing custom adapters

The custom adapters interface makes it possible for users to develop adapters with as much functionality as those provided with the product.

The C API provides:

- The ability to integrate adapters into IBM Transformation Extender products using the XML registration file.
- The ability to participate in connection pooling.
- The ability to provide an event listener.
- An object-oriented interface.

Adapter information originates in the adapters.xml registration file and the vtable adapter library.

Adapter registration and the library virtual table

The first step required to develop an adapter is to add the adapter to the adapters.xml registration file.

The following line is an example of how to add an adapter to the adapters.xml file:

```
<UserAdapter name="File Sample" alias="SFILE"
id="home_markdown_jenkins_workspace_Transform_in_SSVSD8_11.0.1_com.ibm.websphere.dtx.capi.doc_creating_and_initializing_object_201" type="file"
library="filesamp" vendor="abc"/>
```

This contains one key field: the name of the library (library="filesamp"). Note that name must not contain a .DLL, .SO, .SL, or any other library extension.

The adapter must export the library virtual table, which must be given the same name as the library, with _config appended. For example, filesamp_config.

The library virtual table contains a number of pointers to methods and configuration information for the adapter. Its structure is:

```
struct MPI_LIBRARY_VTABLE
{
    size_t nSize; /* size of this structure */
    MPIRC (*pfInitializeLibrary)(void);
    MPIRC (*pfDestroyLibrary)(void);
    MPIRC (*pfCreateAdapterInstance)(HMPPIADAPT *phAdapter, HMPICARD hCard);
    MPIRC (*pfDestroyAdapterInstance)(HMPPIADAPT hAdapter);
    MPIRC (*pfCreateConnectionInstance)(HMPICONNECT *phConnection);
    MPIRC (*pfDestroyConnectionInstance)(HMPICONNECT hConnection);
    MPIRC (*pfCombinedListen)(HMPPIADAPTCOLL hColl, HMPICONNECT hConn);

    char lpszCommandMask; /* Mask of command line arguments defining connection */
}
int nTransMode; /* single, multiple or non-transactional */
```

```

int nBurstType;      /* Unit of work - message or logical */
int bListener;       /* Does the adapter support listener interface */
int bListenerBlocks; /* Does the listener block */
int bListenerTrans;  /* Is the listener transactional? */
int bRetries;        /* Does the adapter support retries */

/* Source settings */
int bSourceManage;   /* Should RM manage the source? */
int bSourceWarnings; /* Does the adapter return warnings on sources? */
int nSourceScopes;   /* Which adapter Scope settings are supported for source?
*/
int nSourceOnSuccess; /* Which OnSuccess settings are supported for source?
*/
int nSourceOnFailure; /* Which OnFailure settings are supported for source?
*/
int nDefSourceScope;  /* Default Scope setting for source */
int nDefSourceOnSuccess; /* Default OnSuccess setting for source */
int nDefSourceOnFailure; /* Default OnFailure setting for source */

/* Target settings */
int bTargetManage;   /* Should RM manage the target? */
int bTargetWarnings; /* Does the adapter return warnings on targets? */
int nTargetScopes;   /* Which adapter scope settings are supported for target?
*/
int nTargetOnSuccess; /* Which OnSuccess settings are supported for target?
*/
int nTargetOnFailure; /* Which OnFailure settings are supported for target?
*/
int nDefTargetScope;  /* Default Scope setting for target */
int nDefTargetOnSuccess; /* Default OnSuccess setting for target */
int nDefTargetOnFailure; /* Default OnFailure setting for target */
};

typedef struct MPI_LIBRARY_VTABLE MPI_LIBRARY_VTABLE;

```

Initializing the adapter

InitializeLibrary provides an entry point by which any global structures, critical sections, or lists that are required by the adapter can be initialized.

After the adapter has been loaded, the InitializeLibrary method is called. This method provides an entry point whereby any global structures, critical sections, lists, and so on required by the adapter can be initialized.

The DestroyLibrary method should be provided to clean up these same data structures.

- [InitializeLibrary](#)

The InitializeLibrary method initializes any memory, data structures, or critical sections that are required by the adapter.

- [DestroyLibrary](#)

The DestoyLibrary method frees any memory, data structures, critical sections and other resources allocated by the adapter (typically those created by the InitializeLibrary method). It is called only once after any other adapter method invocations.

InitializeLibrary

The InitializeLibrary method initializes any memory, data structures, or critical sections that are required by the adapter.

This method is called only once before any other adapter method invocations other than those contained within the property interface at design time. It is not called at design time.

MPIRC InitializeLibrary (void)

Inputs

None

Outputs

None

Returns

Success status

DestroyLibrary

The DestoyLibrary method frees any memory, data structures, critical sections and other resources allocated by the adapter (typically those created by the InitializeLibrary method). It is called only once after any other adapter method invocations.

This method is not called at design time.

```
MPIRC_DestroyLibrary (void)
```

Inputs

None

Outputs

None

Returns

Success status

Point to adapter methods

After your adapter is loaded, you call the virtual table (vtable) structure to obtain pointers to the methods.

Tip: These method names are used in the context of this documentation. However, they are user-definable and can be changed.

Adapter registration file

The **adapters.xml** registration file contains a single tag for each adapter, including a number of attributes to provide information about the adapter.

```
<UserAdapter name="File Sample" alias="SFILE"
id="_home_markdown_jenkins_workspace_Transform_in_SSVSD8_11.0.1_com.ibm.websphere.dtx.capi.doc_adapter_registration_file_201"
type="file"
library="filesamp" vendor="abc"/>
```

Attribute	Description
name	Name of the adapter used in the adapter list in the Map Designer and Information Flow Designer (for example, Database or MQSeries® (server)).
alias	Abbreviated name of the adapter that is used in GET and PUT functions as well as command line overrides (for example, DB for database or MQS for MQSeries).
id	Identification type of the adapter. For adapters developed by IBM®, these numbers are between 0 and 200. For external, user-developed adapters, the id numbers should be >= 200.
type	Class type of the adapter. For example, database (db) or messaging (msg).
library	Name of the DLL or shared library. The DLL/SO/SL file extension should not be provided. In addition, for UNIX adapters, the name should not be prefixed with lib. This will be added when accessing the adapter.
vendor	Name of the adapter manufacturer. The vendor name is displayed along with the adapter name in the Map Designer.

Adapters.xml example

```
<Adapters>
  <M4Adapters>
    <M4Adapter name="File" alias="FILE"
id="_home_markdown_jenkins_workspace_Transform_in_SSVSD8_11.0.1_com.ibm.websphere.dtx.capi.doc_adapter_registration_file_0"
type="file"
library="m4file"/>
    <M4Adapter name="Buffer" alias="BUF"
id="_home_markdown_jenkins_workspace_Transform_in_SSVSD8_11.0.1_com.ibm.websphere.dtx.capi.doc_adapter_registration_file_4"
type="buff" />
    <M4Adapter name="Database" alias="DB"
id="_home_markdown_jenkins_workspace_Transform_in_SSVSD8_11.0.1_com.ibm.websphere.dtx.capi.doc_adapter_registration_file_5"
type="db"
library="dbutil"/>
    <M4Adapter name="Application" alias="APP"
id="_home_markdown_jenkins_workspace_Transform_in_SSVSD8_11.0.1_com.ibm.websphere.dtx.capi.doc_adapter_registration_file_6"
type="app" />
  </M4Adapters>
  <UserAdapters>
    <UserAdapter name="File Sample" alias="SFILE"
id="_home_markdown_jenkins_workspace_Transform_in_SSVSD8_11.0.1_com.ibm.websphere.dtx.capi.doc_adapter_registration_file_201"
type="file"
library="filesamp" vendor="abc"/>
  </UserAdapters>
</Adapters>
```

Virtual table adapter library

The virtual table (vtable) adapter library contains configuration information about the adapter, including the names of functions, transactional capabilities, and so on. This vtable is exported from the DLL/shared library of each adapter.

Virtual table library values

The virtual table (vtable) adapter library contains configuration information about the adapter, including the names of functions, transactional capabilities, and so on. This vtable is exported from the DLL/shared library of each adapter.

Field	Permitted Values and Description
nSize	=sizeof (MPI_LIBRARY_VTABLE) Ensures that the correct structure definition is being used.
nTransMode	MPI_TRANSACTIONS_SINGLE One transaction per connection. MPI_TRANSACTIONS_MULTIPLE Multiple transactions per connection. MPI_TRANSACTIONS_NONE The adapter does not support transactions.
nBurstType	TRUE The adapter supports the listener interface. FALSE The adapter does not support the Listener interface
bListenerBlocks	TRUE The LISTEN function of the adapter blocks indefinitely and requires an external abort to request termination. FALSE The LISTEN function of the adapter does not block.
bListenerTrans	TRUE The listener is transactional and requires a BeginTransaction call. FALSE The listener is not transactional.
bRetries	TRUE The adapter supports retries. FALSE The adapter does not support retries.
bSourceWarningS	TRUE The adapter can return warning codes from a source. FALSE The adapter does not return warning codes from a source.
nSourceScopes	MPI_SCOPE_MAP MPI_SCOPE_BURST MPI_SCOPE_CARD Values can be ORed together to provide a list of Scope settings that are permissible for a source.
nSourceOnSuccess	MPI_ACTION_COMMIT MPI_ACTION_ROLLBACK Values can be ORed together to provide a list of OnFailure settings that are permissible for a source.
nSourceOnFailure	MPI_ACTION_COMMIT MPI_ACTION_ROLLBACK Values can be ORed together to provide a list of OnFailure settings that are permissible for a source.
nDefSourceScope	Default value for Scope.
nDefSourceOnSuccess	Default value for OnSuccess.
nDefSourceOnFailure	TRUE The Resource Manager should manage targets to ensure there is no conflict. FALSE The Resource Manager should not manage targets.
bTargetWarnings	TRUE The adapter can return warning codes from a target. FALSE The adapter does not return warning codes from a target.
nTargetScopes	MPI_SCOPE_MAP MPI_SCOPE_BURST MPI_SCOPE_CARD Values can be ORed together to provide a list of Scope settings that are permissible for a target.
nTargetOnSuccess	MPI_ACTION_CREATE MPI_ACTION_CREATEONCONTENT MPI_ACTION_DONT_CREATE MPI_ACTION_APPEND (file only) MPI_ACTION_UPDATE (file only) Values can be ORed together to provide a list of OnSuccess settings that are permissible for a target.
nTargetOnFailure	MPI_ACTION_COMMIT MPI_ACTION_ROLLBACK Values can be ORed together to provide a list of OnFailure settings that are permissible for a target.
nDefTargetScope	Default value for Scope.
nDefTargetOnSuccess	Default value for OnSuccess.
nDefTargetOnFailure	Default value for OnFailure.

Virtual table example structure

The virtual table (vtable) adapter library contains configuration information about the adapter, including the names of functions, transactional capabilities, and so on. This vtable is exported from the DLL/shared library of each adapter. This is an example of the vtable structure.

```
EXPORT MPI_LIBRARY_VTABLE M4MQS_config =
{
sizeof(MPI_LIBRARY_VTABLE),
InitializeLibrary, /* InitializeLibrary */
DestroyLibrary, /* Destroy Library */
CreateAdapterInstance, /* CreateAdapterInstance */
DestroyAdapterInstance, /* DestroyAdapterInstance */
CreateConnectionInstance, /* CreateConnectionInstance */
DestroyConnectionInstance L, /* DestroyConnectionInstance */
NULL, /* CombinedListen */
"-QN % -MID %" /* Command line mask */
MPI_TRANSACTIONS_SINGLE, /* Transactional mode */
MPI_UNIOTOFWORK_MESSAGE, /* Unit of work */
TRUE, /* Listener interface supported */
FALSE, /* Listener does not block */
FALSE, /* Listener is not transactional */
TRUE, /* Retries */

/** Source Settings ***/
FALSE, /* Resource manage source */
TRUE, /* Warnings */
MPI_SCOPE_MAP | MPI_SCOPE_BURST | MPI_SCOPE_CARD, /* Source scopes */
MPI_ACTION_KEEPONCONTENT | MPI_ACTION_KEEP |
MPI_ACTION_DELETE, /* Source OnSuccess*/
MPI_ACTION_ROLLBACK | MPI_ACTION_COMMIT, /* Source OnFailure */
MPI_SCOPE_MAP, /* Default scope */
MPI_ACTION_DELETE, /* Default OnSuccess */
```

```

MPI_ACTION_ROLLBACK, /* Default OnFailure */
/** Target Settings **/

FALSE, /* Resource manage target */
TRUE, /* Warnings */
MPI_SCOPE_MAP | MPI_SCOPE_BURST | MPI_SCOPE_CARD, /* Target scopes */
MPI_ACTION_CREATEONCONTENT | MPI_ACTION_CREATE, /* Target OnSuccess */
MPI_ACTION_ROLLBACK | MPI_ACTION_COMMIT, /* Target OnFailure */
MPI_SCOPE_MAP, /* Default scope */
MPI_ACTION_CREATE, /* Default OnSuccess */
MPI_ACTION_ROLLBACK /* Default OnFailure */
};

```

Methods provided by an adapter

Various methods must be provided by an adapter.

The following types of methods must be provided by an adapter. Some of these methods are mandatory, while others are optional.

- Library initialization and termination methods
- Adapter instance creation and destruction
- Connection instance creation and destruction
- Source/target interface
- Property interface
- Connection interface
- Listener interface

To see if the adapter method is required or optional, see the following table.

Method	Required (Yes/No)
InitializeLibrary	N
DestroyLibrary	N
CreateAdapterInstance	Y
DestroyAdapterInstance	Y
CreateConnectionInstance	Y
DestroyConnectionInstance	Y
CombinedListen	N ³
Connect	Y
Disconnect	Y
Get	N ¹
Put	N ¹
OnNotify	N ¹
BeginTransaction	N ²
EndTransaction	N ²
Listen	N ³
ValidateProperties	Y
CompareConnection	Y
ValidateConnection	N
CompareWatches	N ^{3,4}

^{1,2,3,4}

¹ Some adapters do not support both Source and Target interfaces, and therefore might only provide a Get or Put method.

² This method is not required if the resource is non-transactional.

³ This method is required for only those adapters that support the Listen interface.

⁴ CompareWatches is required only if CombinedListen is provided by the adapter. If Listen is provided, CompareWatches is not needed.

Adapter methods used post-instance creation

After you create an instance of the adapter and connection objects, the map can use these objects to get data, put data, or to listen for events. The adapter virtual table, MPI_ADAPTER_VTABLE, points to a number of methods that are provided by the adapter.

The methods in the following table can be used after instances of the adapter and connection objects have been created.

Method	Description
BeginTransaction	Start a transaction
EndTransaction	End a transaction
Get	Get data for a source resource
Put	Send data to a target resource
Listen	Listen for an event
ValidateProperties	Check the validity of the adapter's properties
CompareConnection	Determine whether an existing connection can be reused
ValidateConnection	Ensure an existing connection is still alive
CompareWatches	Determine whether two watches are equivalent
OnNotify	This is called to notify the adapter of key events

There is an additional method, CombinedListen, that works with collections of adapters and is not, therefore, a method of the adapter object.

Adapter instance overview

When a map executes, there are a number of objects created (for example, map and card). Each card has an associated adapter instance. The adapter instance contains a number of properties that determine the behavior of the adapter. The number and type of these properties are defined through the mpiObjectNewPropColl method.

When an instance of your adapter is required, the CreateAdapterInstance method is called. This method must call the mpiAdapterCreate method to create the physical object and associate the adapter virtual table with the object. The adapter virtual table has the following structure:

```
struct MPI_ADAPTER_VTABLE
{
    size_t nSize; /* size of this structure */
    MPIIRC (*pfCompareWatches) (HMPIDADAPT hAdapter1, HMPIDADAPT hAdapter2,
                                MPICOMP *peComp);
    MPIIRC (*pfCompareResources) (HMPIDADAPT hAdapter1, HMPIDADAPT hAdapter2,
                                  MPICOMP *peComp);
    MPIIRC (*pfGet) (HMPIDADAPT hAdapter, HMPICONNECT hConnection);
    MPIIRC (*pfPut) (HMPIDADAPT hAdapter, HMPICONNECT hConnection);
    MPIIRC (*pfBeginTransaction) (HMPIDADAPT hAdapter, HMPICONNECT hConnection);
    MPIIRC (*pfEndTransaction) (HMPIDADAPT hAdapter, HMPICONNECT hConnection,
                               MPITRANS eAction);
    MPIIRC (*pfListen) (HMPIDADAPT hAdapter, HMPICONNECT hConnection);
    MPIIRC (*pfValidateProperties) (HMPIDADAPT hAdapter);
    MPIIRC (*pfValidateConnection) (HMPIDADAPT hAdapter, HMPICONNECT hConnection);
    MPIIRC (*pfCompareConnection) (HMPIDADAPT hAdapter, HMPICONNECT hConnection,
                                  MPICOMP *peComp);
    MPIIRC (*pfOnNotify) (HMPIOBJ hObject, int iId, int iParam, HMPIOBJ hParam);
};

typedef struct MPI_ADAPTER_VTABLE MPI_ADAPTER_VTABLE;
```

CreateAdapterInstance example

```
MPI_ADAPTER_VTABLE AdapterTable = { /* Initialize */ };
/* Define the adapter properties */
enum
{
    MYP_USERNAME = MPI_PROPBASE_USER,
    MYP_PASSWORD,
    MYP_SERVER
};

const static int AdapterProperties[] =
{
    MPI_PROP_TYPE_TEXT, /* MYP_USERNAME */
    MPI_PROP_TYPE_TEXT, /* MYP_PASSWORD */
    MPI_PROP_TYPE_TEXT /* MYP_SERVER */
};
#define NUM_ADAPTER_PROPERTIES (sizeof(AdapterProperties)/sizeof(int))

MPIIRC CreateAdapterInstance (HMPIDADAPT *phAdapter, HMPICARD hCard)
{
    MPIIRC rc;

    rc = mpiAdapterCreate(phAdapter, hCard, &AdapterTable);

    if (rc == MPI_SUCCESS)
    {
        /* Define the number and type of adapter properties */
        rc = mpiObjectNewPropColl(*phAdapter,
                                 MPI_PROPBASE_USER,
                                 NUM_ADAPTER_PROPERTIES,
                                 AdapterProperties);

        /* Do any initialization of instance of adapter object */
    }
    return rc;
}
```

DestroyAdapterInstance example

The DestroyAdapterInstance method should do the reverse, that is, it should delete the adapter object and free up any data structures that were created by the create method. The following code example deletes an object.

```
MPIIRC DestroyAdapterInstance (HMPIDADAPT hAdapter)
{
    /* Free up any memory or data structures use by the adapter instance */
    return (mpiObjectDestroy(hAdapter));
}
```

- [Adapter instance creation and destruction](#)

The following methods are called to create an instance of an adapter object, and to destroy this object.

Adapter instance creation and destruction

The following methods are called to create an instance of an adapter object, and to destroy this object.

- [CreateAdapterInstance](#)
This CreateAdapterInstance method creates an adapter object and initializes it.
- [DestroyAdapterInstance](#)
The DestroyAdapterInstance method cleans up and destroys an adapter object.

CreateAdapterInstance

This CreateAdapterInstance method creates an adapter object and initializes it.

```
MPIRC CreateAdapterInstance  
(HMPIDADAPT *phAdapter)
```

Inputs

None

Outputs

phAdapter
Adapter handle

Returns

Success status

DestroyAdapterInstance

The DestroyAdapterInstance method cleans up and destroys an adapter object.

Inputs

```
MPIRC DestroyAdapterInstance  
(HMPIDADAPT hAdapter)
```

phAdapter
Adapter handle

Outputs

None

Returns

Success status

Connection instance overview

Instances of connection objects are created in the same way as adapter instances. When a new connection is required, a connection instance is created.

When an instance of a connection is required, the CreateConnectionInstance method is called. This method must call the mpiConnectionCreate method to create the physical object and associate the connection virtual table with the object.

The connection virtual table has the following structure:

```
struct MPI_CONNECTION_VTABLE  
{  
size_t nSize; /* size of this structure */  
MPIRC (*pfConnect) (HMPICONNECT hConnection, HMPIDADAPT hAdapter);  
MPIRC (*pfDisconnect) (HMPICONNECT hConnection);  
};  
typedef struct MPI_CONNECTION_VTABLE MPI_CONNECTION_VTABLE;
```

CreateConnectionInstance example

The following code is an example of creating a connection instance.

```
MPI_CONNECTION_VTABLE ConnectionTable = { /* Initialize */ };  
  
MPIRC CreateConnectionInstance(HMPICONNECT *phConnection)  
{  
MPIRC rc;  
rc = mpiConnectionCreate(phConnection, &ConnectionTable);
```

```

if (rc == MPI_SUCCESS)
{
/* Define the number and type of adapter properties */
rc = mpiObjectNewPropColl(*phConnection,
MPI_PROPBASE_USER,
NUM_CONNECTION_PROPERTIES,
ConnectionProperties);

/* Do any initialization of instance of connection object */
}
return rc;
}

```

DestroyConnectionInstance example

The DestroyConnectionInstance method should do the reverse, that is, it should delete the connection object and free up any data structures that were created by the create method.

The following code is an example of deleting an object.

```

MPIRC
DestroyConnectionInstance(HMPICONNECT hConnection)
{
/* Free up any memory or data structures used by the connection instance */

return (mpiObjectDestroy(hConnection));
}

```

- [Connection instance creation and destruction](#)

The CreateConnectionInstance and DestroyConnectionInstance methods are called to create an instance of the adapter connection object, and to destroy this object.

-
- [CreateConnectionInstance](#)
The CreateConnectionInstance method creates a connection object and initializes it.
 - [DestroyConnectionInstance](#)
The DestroyConnectionInstace method cleans up and destroys a connection object.
-

CreateConnectionInstance

The CreateConnectionInstance method creates a connection object and initializes it.

```

MPIRC
CreateConnectionInstance
(HMPICONNECT *phConnection)

```

Inputs

None

Outputs

phConnection
Connection handle

Returns

Success status

DestroyConnectionInstance

The DestroyConnectionInstace method cleans up and destroys a connection object.

```

MPIRC
DestroyConnectionInstance (HMPICONNECT hConnection)

```

Inputs

hConnection
Connection handle

Outputs

None

Returns

Success status

Connection interface overview

The connection interface methods are called to connect to a resource and to control connection pooling.

- **[Connect interface method](#)**

The Connect method is called to connect to the resource specified by the adapter object. Context information returned from the connection, such as handles, cursors or other contextual information, is returned by setting properties of the connection object.

- **[Disconnect interface method](#)**

The Disconnect method disconnects from the resource referenced by the connection object. The connection context is obtained by getting the properties of the connection object that were set in the Connect call.

- **[CompareConnection interface method](#)**

The CompareConnection method is called by the Resource Manager to determine whether a connection can be reused.

- **[ValidateConnection interface method](#)**

The ValidateConnection method is called by the Resource Manager on an idle connection to determine whether a connection is still valid. It is invoked when the period specified in the Launcher configuration file expires. The method should ping the connection to determine whether it is still valid, using whatever mechanism the resource supports to do this.

Connect interface method

The Connect method is called to connect to the resource specified by the adapter object. Context information returned from the connection, such as handles, cursors or other contextual information, is returned by setting properties of the connection object.

In addition, this method should copy properties that are relevant for determining whether a connection can be shared (CompareConnection method). For example, if a connection can be shared when the datasource name matches, the datasource property should be obtained from the adapter object and set in the connection object.

```
MPIRC Connect  
(HMPICONNECT hConnection, HMPIADAPT hAdapter)
```

Inputs

hConnection
 Connection handle
hAdapter
 Adapter handle

Outputs

None

Returns

Success status

Disconnect interface method

The Disconnect method disconnects from the resource referenced by the connection object. The connection context is obtained by getting the properties of the connection object that were set in the Connect call.

Inputs

```
MPIRC Disconnect (HMPICONNECT hConnection)  
  
hConnection  
    Connection handle
```

Outputs

None

Returns

Success status

CompareConnection interface method

The CompareConnection method is called by the Resource Manager to determine whether a connection can be reused.

It is called in two cases:

- To determine whether multiple adapter references within the same map can share a connection.
- To determine whether a connection that is currently inactive can be used by a map.

This method can be logically equivalent to CompareWatches.

```
MPIRC CompareConnection (HMPIDADAPT hAdapter, HMPICONNECT hConnection,  
                           MPICOMP *peComp)
```

Inputs

hAdapter
Adapter handle for the potentially new connection.
hConnection
Connection handle for the existing connection.

Outputs

peComp

MPI_CMP_EQUAL
 Connection corresponds.
MPI_CMP_DIFFER
 Connection does not correspond.
MPI_CMP_LESS
 Connection is less than adapter.
MPI_CMP_GREATER
 Connection is greater than adapter.

Returns

Success status

The adapter consults the properties for the adapter object and the connection object and determines whether the properties correspond such that the connection might be reused for the adapter object. The MPIP_CONNECTION_INUSE property determines whether the connection object is currently being used in a map. If this property is FALSE, the connection is currently idle.

Besides adapter-specific properties, other properties might be important in determining whether a connection can be reused. For example, MPIP_ADAPTER_ON_FAILURE determines whether to rollback or commit at the end of the transaction. If this is set to MPI_ACTION_COMMIT for one adapter object, but MPI_ACTION_ROLLBACK for another, it is most unlikely that the connection should be shared if the connection is active.

If the sense of ordering in comparisons can be made, the number of comparisons can be reduced because every existing connection does not have to be compared when searching for a valid connection to use.

In the following code example, a resource is identified by QueueManager.

```
mpiPropertyGetText(hAdapter, MYPROP_Q_MGR, 0, &pAdaptQMgr, &iLen);  
mpiPropertyGetText(hConnection, MYPROP_Q_MGR, 0, &pConnQMgr, &iLen);  
rc = strcmp(pAdaptQMgr, pConnQMgr);  
if (rc == 0)  
    return MPI_CMP_EQUAL;  
else if (rc < 0)  
    return MPI_CMP_LESS;  
else  
    return MPI_CMP_GREATER;
```

ValidateConnection interface method

The ValidateConnection method is called by the Resource Manager on an idle connection to determine whether a connection is still valid. It is invoked when the period specified in the Launcher configuration file expires. The method should ping the connection to determine whether it is still valid, using whatever mechanism the resource supports to do this.

```
MPIRC ValidateConnection (HMPIDADAPT  
                           hAdapter, HCONNECTION hConnection)
```

Inputs

hAdapter
Adapter handle
hConnection

Connection handle

Outputs

None

Returns

MPIIRC_SUCCESS

Connection is good.

MPIIRC_BAD_CONNECTION

Connection is no longer good.

This method is optional. If not provided, the connection is assumed to be invalid when its expiration timeout occurs. This timeout is currently defined in the **config.yaml** configuration file.

Library initialization and termination methods

The InitializeLibrary and DestroyLibrary custom adapter methods are called to perform one-time initialization tasks and to clean up from these tasks.

Listener interface overview

The Listener interface uses the Listen, CombinedListen, and CompareWatches methods to listen for single or multiple events from the resource.

There are two distinct ways to implement a listener:

- A listener listens for a single event. If there are other listeners for the adapter, those also listen for single events. If there is any overlap in what is being listened for this, it is handled by the Resource Manager.
For example, there might be two listeners listening on the same queue, both of which might report the same event to the Resource Manager. For this scenario, the adapter should provide a Listen method. In this instance, there is no need to provide a CompareWatches method.
- A listener listens for multiple events. If it is not possible to have more than one listener listening on the same queue or for the same file and so on, or if it is inefficient to do so, a single listener should be provided that handles multiple events.
For this scenario, the adapter should provide a CombinedListen method. If CompareWatches returns MPI_CMP_EQUAL, the watches that were returned as being equal are passed to the CombinedListen method.

Only one of these methods should be provided, depending entirely on whether multiple separate threads can listen for the same resource.

- [Listen method](#)
The Listen method listens for events and returns when an event has occurred.
 - [CombinedListen method](#)
The CombinedListen method is generally the same as the Listen method. The difference is that the CombinedListen method listens for multiple watches.
 - [CompareWatches method](#)
The CompareWatches method compares two watches to see whether they are equivalent. This method is equivalent to CompareConnection method.
-

Listen method

The Listen method listens for events and returns when an event has occurred.

For example, a message has been received, a database update has occurred, and so on. In addition, should a connection become unusable, the Listen method should return a status indicating this.

```
MPIIRC Listen (HMPIDADAPT  
hAdapter,HCONNECTION hConnection)
```

Inputs

hConnection

Connection handle

Outputs

None

Returns

MPIIRC_BAD_CONNECTION

Connection failed other success status

In the vtable library, there is a bListenerBlocks configuration option that specifies whether the listener is interruptible. To be interruptible means that the function that the adapter calls to listen for an event must be able to be interrupted by a call from another thread.

Generally, unless the API provides a break call, the adapter is non-interruptible. If the adapter is non-interruptible, the Listen method should never execute for more than a few seconds. After being called, the adapter makes a call to its resource to discover new events. If, after a few seconds, no event has occurred, the Listen method should return.

Returning back to the Resource Manager allows the Resource Manager to determine whether to invoke Listen again, the most common scenario, or to shutdown the watch either because the Launcher has asked for the watch to be deleted, or the Launcher is in the process of shutting down. In the latter cases, the Resource Manager will invoke Disconnect to clean up the watch.

If the Listen method for the adapter is interrupted, it does not need to return to the Resource Manager every few seconds if no event occurs, although it must return whenever an event occurs.

Instead, whenever a listener must be terminated by the Resource Manager, the OnNotify method will be called with a notification code of MPIN_ADAPTER_LISTENABORT on another thread from the listener. Within this call, the adapter should execute the break function on the listener's connection context to cause it to terminate and return control back to the Resource Manager.

If one or more events occur, the Listen method should return immediately, adding the event data to the message collection. For example,

```
MyGetMessageID(&pMID);  
mpiPropertyGetObject(hAdapter,  
MPIP_ADAPTER_MSG_COLLECTION, 0, &hMsgColl);  
mpiMsgCollNewMsg(hMsgColl, pMID, strlen(pMID),  
NULL, 0, NULL, NULL);
```

- [Listen method attributes](#)

The Listen method listens for events and returns when an event has occurred. The Listen method uses the attributes Resource ID, Wildcard, and Userdata.

Listen method attributes

The Listen method listens for events and returns when an event has occurred. The Listen method uses the attributes Resource ID, Wildcard, and Userdata.

A message that is an event contains one to three of the following attributes that you set when a message is added to the collection.

Resource ID

An identifier that uniquely identifies the message, file or buffer (for example, message ID, memory address, and file handle). The Resource ID is used in the Get method to actually retrieve the physical data.

Wildcard

If a property of the adapter contains a wildcard, the actual value that represents the wildcard is specified as an attribute of the message. The mpiExtractWildcard method can be called to get the value of the wildcard.

Userdata

The userdata attribute is provided so that any other data can be associated with the message. Some adapters actually get the data for the event during the Listen call. For example, messaging systems that do not permit non-destructive browsing. For these, the message data should be associated with the userdata attribute.

If userdata is provided, the adapter must provide a method to free up the memory used by the userdata. This method is of type MPI_PF_FREEUSERDATA which is defined as:

```
typedef void (*MPI_PF_FREEUSERDATA)(void *);
```

CombinedListen method

The CombinedListen method is generally the same as the Listen method. The difference is that the CombinedListen method listens for multiple watches.

```
MPIRC_CombinedListen (HMPIADAPTCOLL  
hAdapterColl, HCONNECTION hConnection)
```

Inputs

hAdapterColl
Adapter collection handle
hConnection
Connection handle

Outputs

None

Returns

MPIRC_EVENT
Event occurred
MPIRC_NO_EVENT
No event occurred
MPIRC_BAD_CONNECTION
Connection failed other success status

This method receives an adapter collection object, which is a container for adapter objects. Notice that there is only one connection object because all watches are accessing the same resource. To determine the number of adapter objects in the collection, the following method should be invoked:

```
mpiAdaptCollGetCount (hAdapterColl, &iCount);
```

Upon detection of an event, CombinedListen must determine the watches for which the event applies. For example, the combined listener is looking for the following files:

- Watch #1: A*.txt
- Watch #2: ABC*.txt

If a file is discovered with the name AXYZ.txt, the event is applicable to the first watch only. However, if a file with name of ABCXYZ.txt is discovered, this would be applicable to both Watch 1 and 2.

Having determined the watches to which the event applies, a message should be added to the MPIP_ADAPTER_MSG_COLLECTION property for each corresponding adapter object (in the same way as described in the Listen method). Note that in the above example, the value of the wildcard would differ in the two message collections.

Between successive calls to CombinedListen, a watch might be added or removed. This means that it is necessary for the adapter to check the number of adapter objects in the collection on each invocation. If the number of adapter objects changes between invocations, the adapter might need to do additional processing. For example, to adjust filtering of messages accordingly.

CompareWatches method

The CompareWatches method compares two watches to see whether they are equivalent. This method is equivalent to CompareConnection method.

```
MPIRC CompareWatches (HMPIADAPT hAdapter1, HMPIADAPT hAdapter2, MPICOMP *peComp)
```

Inputs

hAdapter1
Adapter handle
hAdapter2
Adapter handle

Outputs

peComp

MPI_CMP_EQUAL
 Watches correspond
MPI_CMP_DIFFER
 Watches do not correspond
MPI_CMP_LESS
 Watch1 is less than Watch2
MPI_CMP_GREATER
 Watch1 is greater than Watch2

Returns

Success status

If the adapter does not combine listeners, that is provides Listen as opposed to the CombinedListen method, there is no need to provide a CompareWatches method.

If the method returns MPI_CMP_EQUAL, only one listener thread is created to handle multiple equivalent watches, and the CombinedListen method will be called to handle the multiple combined watches.

If wildcards are part of the command, it might not be possible to combine watches. For example, if there are two commands, to the same queue but one has MID="FRED" and the other MID="*", in some instances, the resource is the same, but not in others.

In this instance, the adapter might handle both events, returning MPI_CMP_EQUAL from this call, or two separate listeners might be spawned, returning MPI_CMP_DIFFER, MPI_CMP_LESS or MPI_CMP_GREATER from this method. The determination of the path to follow can be determined by performance reasons or limitation in the resource's API.

If the sense of ordering in comparisons can be made, the number of comparisons can be reduced because every existing watch does not have to be compared when searching for a valid connection to use.

The following example shows a resource identified by QueueManager:

```
mpiPropertyGetText(hAdapter1, MYPROP_Q_MGR, 0, &pAdapt1QMgr, &iLen);  
mpiPropertyGetText(hAdapter2, MYPROP_Q_MGR, 0, &pAdapt2QMgr, &iLen);  
rc = strcmp(pAdapt1QMgr, pAdapt2QMgr);  
if (rc == 0)  
    return MPI_CMP_EQUAL;  
else if (rc < 0)  
    return MPI_CMP_LESS;  
else  
    return MPI_CMP_GREATER;
```

Method invocation examples

The method invocation examples demonstrate the sequence of calls made to an adapter by Launcher.

- [Get and put method invocation examples](#)

These are examples of the sequence of calls that occur for Get or Put within a map.

- [Quantity, FetchUnit, and listen time method invocation examples](#)

These are examples of the sequence of calls that occur for Quantity, FetchUnit, or Listen Time method within a map.

Get and put method invocation examples

These are examples of the sequence of calls that occur for Get or Put within a map.

It demonstrates sharing connections, both active (the connection is already being used in the current map) and inactive (the connection is not being used by any current map).

Sequence of calls

1. The map calls **ValidateProperties**.
2. The map determines if an active connection exists.

For inactive connections

1. The map calls **CompareConnections**.
2. The map determines if there is a match.
3. If there is a match, the map calls:
 - **BeginTransaction**
 - **OnNotify(Start)**
4. If there is no match, the map calls:
 - **Connect**
 - **BeginTransaction**
 - **OnNotify(Start)**
5. The map then calls **Get** or **Put** for every message available to process.

For active connections

1. The map calls **CompareConnections**.
2. The map determines if there is a match.
3. If there is a match, the map calls:
 - **OnNotify(Start)**
4. If there is no match, the map calls:
 - **Connect**
 - **BeginTransaction**
 - **OnNotify(Start)**
5. The map then calls **Get** or **Put** for every message available to process.

No more messages to process

1. If there are no more messages to process, the map calls:
 - **OnNotify(Stop)**
 - **EndTransaction**
 - **Disconnect**

The Disconnect call is not made immediately after the EndTransaction, but only after the process has terminated or the connection expiration timeout occurs. The description shows the scenario in which the Scope setting is set to Map. Even if there are multiple bursts, there is only one call to BeginTransaction and one to EndTransaction. If the Scope setting was instead set to Burst, BeginTransaction would be called at the start of each burst, and EndTransaction at the end of the burst. This scenario for adapters is for allowing only one transaction for each connection. For an adapter to allow multiple transactions on a connection, the BeginTransaction call is made even if the CompareConnection call returns MPI_CMP_EQUAL.

Quantity, FetchUnit, and listen time method invocation examples

These are examples of the sequence of calls that occur for Quantity, FetchUnit, or Listen Time method within a map.

For an input card that is not a source event, the Resource Manager might call the Get method more than once if it determines there is more data to be returned. The Resource Manager interprets two special command line options that can be specified in the adapter's command line:

- -QTY: number of messages to be processed by the map
- -LSN: amount of time to listen for messages for each burst of the map

In addition to these command options, the FetchUnit card setting determines how many messages are returned from each burst. Therefore, if the FetchUnit is greater than one (1) or the -QTY (quantity) command is provided on the adapter command line, the Get method will be called more than once.

Carefully adhere to the following rules to ensure that the Get method is called correctly. Note that "X messages" means X messages per burst of the map. For example, if the map has multiple bursts, then each burst will fetch the FetchUnit and decrement this number from the quantity. Therefore, if FetchUnit = 3 and QTY = 7, X is 3 (burst #1), 3 (burst #2), and 1 (burst #3).

- When X messages, -LSN Y (where Y > 0) The Get method is called repeatedly until either: The adapter has been listening for Y seconds (cumulatively across all Get calls for the burst). X messages have been returned for the burst. Note that returning zero (0) messages from a Get call does not mark the end of the data. The Get will be called again as long as the elapsed time does not exceed Y and the total number of messages does not exceed X.
- When X messages, -LSN 0
 - The purpose of -LSN 0 is to fetch all data that is immediately available, such as all messages that are sitting on a queue, without ever waiting for data to arrive. Therefore, whenever an absence of data is detected, it marks the end of the input.
- The Get method is called repeatedly until either:
 - The adapter returns no messages from a call to Get if (MPIP_ADAPTER_NUM_MESSAGES) is 0. The adapter will not be called again for this burst (contrast with -LSN Y where Y > 0)
 - X messages have been returned for the burst
- X messages, -LSN not specified
 - If -LSN is not specified, this implies listen forever (an infinite time). Therefore, the Get will be called repeatedly until -QTY messages have been received.
- -QTY S, -LSN not specified
 - This is an invalid configuration because it means get all of the messages that are returned in an infinite amount of time, (that is, the map will never complete). The Resource Manager will detect this and return an error at the point of property validation.

On each call to Get, the adapter must set the MPIP_ADAPTER_NUM_MESSAGES property to the number of messages being returned.

If an adapter recognizes that there is no more data (for example, a database query has retrieved all of the rows in the result set), the adapter should return MPIRC_W_END_OF_DATA. The adapter will not be called back for more data.

Source and target interface overview

Source and target interface methods are called to get data from a resource, to put data to a resource, and to control a transaction while developing custom adapters.

- [Get method overview](#)
The Get method is called to obtain data from a resource for input to a map.
- [Put method overview](#)
The Put method is called to send data to a resource for output from a map.
- [OnNotify method overview](#)
The OnNotify method is called to notify an adapter about the progress of the map as related to the adapter.
- [BeginTransaction method overview](#)
The BeginTransaction method is called to explicitly start a transaction.
- [EndTransaction method overview](#)
The EndTransaction method is called at the end of the transaction to complete the transaction.

Get method overview

The Get method is called to obtain data from a resource for input to a map.

The Get method is called for an input card, GET function, or for a DBLOOKUP and DBQUERY if the source is a database. The call accesses the resource for which the hConnection handle is associated and returns data. This data is passed to the map server through a stream object.

```
MPIRC Get (HMPIADAPT hAdapter, HMPICONNECT  
hConnection)
```

Inputs

hAdapter	Adapter handle
hConnection	Connection handle

Outputs

None

Returns

Success status

If the Get method is invoked as a result of an input event (the card acts as a watch for the Launcher and a Listen method has returned an event), the Get method should process the event. To determine whether the call is in the context of an event, see the MPIP_ADAPTER_CONTEXT property. For example:

```
mpiPropertyGetInteger(hAdapter, MPIP_ADAPTER_CONTEXT,  
&iContext);
```

This will return MPI_CONTEXT_SOURCE_EVENT if the card is an input event.

If Get is not being called from an event, the method should get a single unit of work from the input (for example, a result set from a database query, a file, or a message from a queue), and return the unit of work through a stream object associated with the MPIP_ADAPTER_DATA_FROM_ADAPT property. For example:

```
mpiPropertyGetObject (hAdapter,  
MPIP_ADAPTER_DATA_FROM_ADAPT, 0, &hStream);  
mpiStreamNewPage(hStream, &hPage, iSize);
```

```
mpiStreamPageGetInfo(hPage, &pData, &iSize);
get data into pData
mpiStreamWritePage(hStream, hPage);
```

Quantity or FetchUnit values do not affect the adapter; this is handled by the Resource Manager. Each call to Get should return a single unit of work. If Get is called from the GET mapping function (the MPIP_ADAPTER_CONTEXT will be set to MPI_CONTEXT_GET), data might be passed to the Get call from the third parameter in the GET mapping function. This applies to adapters that require data to be sent out in the form of a request, such that the corresponding reply is returned from the Get call.

To determine whether any data has been passed, the adapter should get the MPIP_ADAPTER_DATA_TO_ADAPT property. If this returns an MPIRC_NULL_OBJECT error, it means that no data was passed to the Get. If data was passed, use the stream object to get the data.

If Get is called as the result of an input event (the MPIP_ADAPTER_CONTEXT property is MPI_CONTEXT_SOURCE_EVENT), information about the event or events is obtained from the MPIP_ADAPTER_MSG_COLLECTION property by repeatedly calling mpiMsgCollGetNext until there are no more messages in the collection. These messages have previously been added to a collection by the adapter's Listen method.

Put method overview

The Put method is called to send data to a resource for output from a map.

The method is called for an output card or PUT function. The call accesses the resource for which the hConnection handle is associated and sends data to the resource. This data is passed from the map server through a stream object.

```
MPIRC Put (HMPIADAPT hAdapter,
HMPICONNECT hConnection)
```

Inputs

hAdapter
Adapter handle
hConnection
Connection handle

Outputs

None

Returns

Success status

OnNotify method overview

The OnNotify method is called to notify an adapter about the progress of the map as related to the adapter.

```
MPIRC OnNotify (HMPIOBJ hAdapter, int
iEvent,int iParam,HMPIOBJ hConnection)
```

Inputs

hAdapter
Adapter handle
iEvent
Event ID
iParam
[ignore this parameter]
hConnection
Connection handle

Outputs

None

Returns

Success status

The OnNotify method is called at the following times and with the following IDs:

Notification	Description
MPIN_ADAPTER_GETSTART	Prior to any Get calls for the card
MPIN_ADAPTER_GETSTOP	After all Get calls for the card
MPIN_ADAPTER_PUTSTART	Prior to any Put calls for the card

Notification	Description
MPIN_ADAPTER_PUTSTOP	After all Put calls for the card
MPIN_ADAPTER_LISTENSTART	Prior to any Listen calls for the card
MPIN_ADAPTER_LISTENSTOP	After all Listen calls for the card
MPIN_ADAPTER_LISTENABORT	Called to abort a non-polling listener
MPIN_ADAPTER_MAPABORT	When a map is cancelled by user action

These entry points provide the opportunity for the adapter to initialize or clean up the Get, Put, or Listen method. For example, the adapter might need to query a queue to find out some properties about the queue. This would be done in the OnNotify (*_START) call. If it was done in the Get, Put, or Listen and these methods were called multiple times, unnecessary work would be performed repeatedly.

OnNotify is called with *_START prior to any Get, Put, or Listen, and is called with *_STOP after all Get, Put, or Listen calls. For example, if a map is executing multiple bursts and within each burst, Get is called multiple times to get multiple messages, there will be only one OnNotify call with MPIN_GET_START, prior to the first Get. Only after all bursts and all Gets within a burst, will OnNotify be called with MPIN_GET_STOP to allow the adapter to clean up.

If a user cancels a map execution, OnNotify will be called with MPIN_MAP_ABORT. The Get or Put methods should be aborted by the adapter as soon as possible after the notification.

The MPIN_ADAPTER_LISTENABORT notification is delivered only to adapters that have defined bListenerBlocks as TRUE in the vtable library. In this scenario, OnNotify is called on a separate thread from the Listen call. For all other notifications, the call is made on the connection thread.

When an adapter receives this notification, it should abort the Listen method using whatever mechanism or API that is supported by the adapter.

BeginTransaction method overview

The BeginTransaction method is called to explicitly start a transaction.

This method is optional because some resources are non-transactional, and for others, a transaction is implicitly started when the connection is established.

```
MPIRC BeginTransaction (HMPIDADPT
hAdapter, HMPICONNECT hConnection)
```

Inputs

hAdapter
Adapter handle
hConnection
Connection handle

Outputs

None

Returns

Success status

This method is always invoked following a Connect call and before any Get or Put calls. If the adapter operates in the mode in which the listener and map connection threads are combined, BeginTransaction is called prior to the Listen call.

EndTransaction method overview

The EndTransaction method is called at the end of the transaction to complete the transaction.

eAction will have the value MPI_ACTION_COMMIT or MPI_ACTION_ROLLBACK to indicate the course of action that the adapter should take. Note that the end of the transaction might be the end of the card, the burst or the map.

```
MPIRC EndTransaction (HMPIDADPT
hAdapter, HMPICONNECT hConnection, MPITRANS eAction)
```

Inputs

hAdapter
Adapter handle
hConnection
Connection handle
eAction
Indicates whether to commit or rollback the transaction.

Outputs

None

Returns

Success status

If there are multiple cards sharing a connection and the adapter allows for only one transaction per connection, the hAdapter passed to EndTransaction is the hAdapter of the card that corresponds to the one passed to the BeginTransaction call.

Property interface overview

The ValidateProperties custom adapter method is called to validate the adapter's properties.

- [ValidateProperties method overview](#)

The ValidateProperties method is called to ensure that all required properties are set and that there are no mutually exclusive property settings.

ValidateProperties method overview

The ValidateProperties method is called to ensure that all required properties are set and that there are no mutually exclusive property settings.

MPIRC ValidateProperties (HMPIADAPT hAdapter)

Inputs

hAdapter
Adapter handle

Outputs

None

Returns

Success status

The adapter properties are not set by most modes of execution. Therefore, it is necessary for the adapter to get the command line property, to parse it, and to set the properties within this method.

To get the command line property, issue the following property call:

```
mpiPropertyGetText(hAdapter, MPIP_ADAPTER_COMMANDLINE, 0, &lpszCmdLine, &nLen)
```

After the command line has been fetched, it should be parsed and a number of mpiPropertySet* calls should be made. In the process, the options should be validated to ensure that they are consistent and mutually compatible.

To store additional properties or other information for the adapter, set the MPIP_ADAPTER_USER_DATA property in the adapter object. In subsequent calls, the user data can be retrieved by calling mpiGetPropertyBinaryPointer. If the MPIP_ADAPTER_USER_DATA property is set, the adapter must free this memory in the DestroyAdapterInstance call.

C API return codes

There are three types of C API return codes: success, warning, or error.

- Success: The call succeeded. The only success code is MPIRC_SUCCESS.
- Warning: The call succeeded, but some condition arose that might be of concern. For example, MPIRC_W_NO_DATA.
- Error: The call failed.

The following return codes, which are defined in the dtspi.h header file along with many others, should be used by an adapter.

Return code	Description
MPIRC_SUCCESS	This operation was successful.
MPIRC_W_NO_DATA	The adapter has no data to return from a Get.
MPIRC_W_END_OF_DATA	All data has been retrieved from the input.
MPIRC_E_BAD_CONNECTION	The connection became unusable.

In addition to predefined codes, an adapter might provide its own return codes. Error and warning codes returned should be defined using the following macros:

- MAKEUSERWARNING(code) - sets the code as a warning code
- MAKEUSERERROR(code) - sets the code as an error code

Example

```
#define MY_E_CONNECT_FAILED  
MAKEUSERERROR(1)  
  
#define MY_E_BAD_PARMS  
MAKEUSERERROR(2)
```

```

#define MY_E_NO_MEMORY
MAKEUSERERROR(3)

#define MY_W_NO_MSG_AVAIL
MAKEUSERWARNING(1)

#define MY_W_TIMED_OUT
MAKEUSERWARNING(2)

```

Zero (0) should not be used as the parameter of MAKEUSERERROR or MAKEUSERWARNING.

These codes must then be returned as properties of the object.

Example

```
mpiPropertySetInteger(hAdapter,
MPIP_OBJECT_ERROR_CODE, 0, rc);
```

Return codes from the resource to which the adapter connects might also be returned, (for example, a SQL State or code from an R/3 API) by setting the MPIP_ADAPTER_PROVIDER_CODE property. The return code from the adapter methods determines program flow. If an error occurs in an adapter method, an error code must be returned in the return code. The MPIP_OBJECT_ERROR_CODE property does not determine flow, but is used to maintain error information related to various objects. This information can then be displayed in the trace and audit logs.

Additional adapter methods

There are a number of additional methods that are part of the Transformation Extender Programming Interface library that are of particular relevance to adapter developers.

- [Wildcard methods overview](#)

These methods are used to perform wildcard matching and determine the value for which a wildcard stands.

- [mpiMapReportStatus method overview](#)

The mpiMapReportStatus method allows an adapter to display a short message in the map status window, and to determine whether the map has been cancelled by a user action.

Wildcard methods overview

These methods are used to perform wildcard matching and determine the value for which a wildcard stands.

- [mpiCompareWildcard method overview](#)

The mpiCompareWildcard method compares a string (passed in pActual) with a wildcard pattern (pPattern) to determine whether it matches.

- [mpiExtractWildcard method overview](#)

The mpiExtractWildcard method compares a string (passed in pActual) with a wildcard pattern (pPattern) to see whether it matches. If it does, the string that substitutes for the wildcard value is returned in the buffer provided by the caller in pBuf.

mpiCompareWildcard method overview

The mpiCompareWildcard method compares a string (passed in pActual) with a wildcard pattern (pPattern) to determine whether it matches.

For example, if pPattern is A*123 and pActual is ABCD123, the return status will be MPI_CMP_EQUAL because pActual does match the pPattern string.

```
MPIRC mpiCompareWildcard (const char
*pPattern,int nPatternLen,const char
*pActual,int nActualLen)
```

Inputs

pPattern
Wildcard pattern
nPatternLen
Length of the pPattern string.
pActual
Actual data which might match the wildcard.
nActualLen
Length of the actual data.

Outputs

pBuf
A buffer to receive the wildcard value if one exists.
pLen
Size of the buffer on input and the length of the wildcard value on output.

Returns

MPI_CMP_EQUAL
Wildcard matches.
MPI_CMP_LESS
Wildcard does not match.

mpiExtractWildcard method overview

The mpiExtractWildcard method compares a string (passed in pActual) with a wildcard pattern (pPattern) to see whether it matches. If it does, the string that substitutes for the wildcard value is returned in the buffer provided by the caller in pBuf.

For example, if pPattern is A*123 and pActual is ABCD123, the value returned in pBuf will be BCD.

```
MPIRC mpiExtractWildcard (const char
    *pPattern,int nPatternLen,const char
    *pActual,int nActualLen,char *pBuf,int
    *pLen)
```

Inputs

pPattern
Wildcard pattern
nPatternLen
Length of the pPattern string
pActual
Actual data which might match the wildcard
nActualLen
Length of the actual data

Outputs

pBuf
A buffer to receive the wildcard value if one exists
pLen
Size of the buffer on input and the length of the wildcard value on output

Returns

MPI_CMP_EQUAL
Wildcard matches
MPI_CMP_LESS
Wildcard does not match

mpiMapReportStatus method overview

The mpiMapReportStatus method allows an adapter to display a short message in the map status window, and to determine whether the map has been cancelled by a user action.

mpiMapReportStatus is used to display a message in the map status window.

If an adapter attempts to determine the state of a map, that is, to see whether the map has been cancelled, it will receive an MPIN_MAP_ABORT notification through its OnNotify method. On receipt of this notification, the adapter should abort any current activity as soon as possible.

This little command copies things.

Passing data to and from maps

Data is passed between adapters and the engine through stream objects. From the adapter perspective, there is no need to create these stream objects; they are already created when either the Get or Put method is called.

Data is stored within the stream objects in pages which might be of variable size. The adapter is responsible for writing pages to the stream (in a Get), or reading pages from the stream (in a Put).

- [Receiving data from a map](#)
Use the Put method to receive data from a map through stream objects. There are two different ways of accomplishing this. The method chosen depends on the manner in which the data has to be sent to the resource.
- [Sending data to a map](#)
Use a Get method to return (send) data to the engine through stream objects. There are a number of different ways to accomplish this task. The method chosen depends on the manner in which the resource returns data to the adapter.

Receiving data from a map

Use the Put method to receive data from a map through stream objects. There are two different ways of accomplishing this. The method chosen depends on the manner in which the data has to be sent to the resource.

- The data must be passed to the resource as a single, contiguous buffer.

In this example, the buffer is obtained from the stream and passed to the resource:

```
mpiStreamGetSize (hStream, &nSizeOfData);

pData = malloc(nSizeOfData);

mpiStreamRead (hStream, pData, nSizeOfData,
&iBytesRead);
```

This copies the data from the stream page(s) to the buffer.

- The data can be passed to the resource as a number of small buffers.

In this example, each page is obtained from the stream and is then passed to the resource:

```
while(TRUE)
{
mpiStreamIsEnd (hStream, &bIsEnd);
if (bIsEnd)
break;
mpiStreamReadPage (hStream, &hPage);
mpiStreamPageGetInfo (hPage, &pData, &nSizeOfPage, &nSizeOfData);
MySendData (pData, nSizeOfData);
}
```

There might be scenarios where it is acceptable to send data to the resources in pieces, but the size of the pieces might not correspond to the size of a page. As an example, consider database rows. It is possible that a row of data spans across page boundaries. The adapter must expect and be able to handle this.

Sending data to a map

Use a Get method to return (send) data to the engine through stream objects. There are a number of different ways to accomplish this task. The method chosen depends on the manner in which the resource returns data to the adapter.

- A memory buffer is returned from the resource that can be passed to the engine.

In this scenario, only a single call to mpiStreamWrite is required:

```
mpiStreamWrite (hStream, pData, nSizeOfData);
```

This will create a page, copy the data to the page and append the page to the stream.

- The adapter allocates a single buffer that is then populated by the resource.

If the adapter is receiving data into a buffer that it has allocated, the buffer should be allocated through the stream interface as shown in the following example:

```
/* Create a page */
mpiStreamNewPage (hStream, &hPage, nSizeOfBuffer);

/* Get the address of the buffer */
mpiStreamPageGetInfo (hPage, &pData, &nSizeOfBuffer);

/* Get the data into the buffer */
MyGetData (pData, &ActualDataSize);

/* Add the page to the stream */
mpiStreamWritePageEx (hStream, hPage, nActualDataSize);
```

This will create a page and append the page to the stream. This is the preferable way to handle this kind of resource because it requires no memory copies or reallocations.

- The adapter gets data in pieces from the resource.

This is identical to the above scenario except that it is allocated repeatedly. Each call to mpiStreamWritePage appends a page to the stream.

Listener scenario overview

These are listener scenario examples that demonstrate the flow of calls for input events.

To determine the model that best fits the requirements of the resource, follow these examples to understand the flow of calls for input events as they summarize the available alternatives.

- Adapters Process Similar Events
- Resource Manager Processes Similar Events
- Listener and Map Connection Threads are Combined

The meaning of the following similar events is that the connection is identical in both cases, but the event itself might differ. For example, consider:

```
"QueueName = fred, MessageID = 100"
```

and

```
"QueueName = fred, MessageID = 200"
```

If the API for the resource allows the Message ID to be provided to a Get call, such that only the message with the specified Message ID is returned, it is probably easiest to create multiple threads, one getting messages with ID 100; the other messages with ID 200. This is the model used in the Resource Manager Processes Similar Events example.

However, if the API cannot do this selection or it is not possible to have multiple connections to the same resource, the adapter must handle the filtering of messages itself. In the latter scenario, a single Listen method will have to handle multiple events.

- [**Listener scenario: adapters process similar events**](#)

This is an example scenario in which the adapter combines watches and provides CombinedListen and CompareWatches methods.

- [**Listener scenario: Resource Manager processes similar events**](#)

This is an example of a Listener scenario in which the adapter does not, or can not, combine watches.

- [**Listener scenario: Listener and map connection threads are combined**](#)

This is an example scenario in which an adapter requires that all activities related to an input event are carried out within the same thread.

Listener scenario: adapters process similar events

This is an example scenario in which the adapter combines watches and provides CombinedListen and CompareWatches methods.

This example describes a scenario where the adapter combines watches and provides CombinedListen and CompareWatches methods. The adapter in this example is an input to two maps and is marked as an input event for both. The following events occur at initialization:

1. The Launcher starts and reads the system file (.msl) and calls the Resource Manager, passing the details of two watches for the adapter.
2. The Resource Manager calls the ValidateProperties method of the adapter with Watch 1 data.
3. The Resource Manager calls the ValidateProperties method of the adapter with Watch 2 data.
4. The Resource Manager calls the CompareWatches method with adapter objects from the above calls. Because the adapter will handle the events itself, it returns information indicating that the watches are the same (MPI_CMP_EQUAL).
5. The Resource Manager creates a thread (T1) and calls the Connect method with the adapter object (from Step 2). Resource Manager has an arbitrary choice. Because the adapter has indicated that the watches are equivalent, it should not matter which adapter object is used to determine the connection parameters.
6. (T1): The Resource Manager calls the OnNotify (MPIN_LISTEN_START) method and then the CombinedListen method with both of the adapter objects from Steps 2 and 3 in an HAdapterCollection.
7. The Resource Manager notifies the Launcher that the listener started successfully.
Now an event occurs.
8. (T1): The CombinedListen method identifies whether the adapter objects have been triggered and adds messages to the collection objects of the adapters.
9. (T1): The Resource Manager notifies the Launcher passing the appropriate adapter object.
10. (T1): The Resource Manager calls the CombinedListen method again.
11. The map runs and the Resource Manager is called for input.

Listener scenario: Resource Manager processes similar events

This is an example of a Listener scenario in which the adapter does not, or can not, combine watches.

This example describes a scenario where the adapter does not, or cannot, combine watches itself. This means that the adapter provides a Listen method, and does not provide CombinedListen or CompareWatches methods. The adapter in this example is an input to two maps and is marked as an input event for both. The following occurs at initialization:

1. The Launcher starts and reads the Launcher system file (.msl) and calls the Resource Manager, passing the details of two watches for the adapter.
2. The Resource Manager calls the ValidateProperties method of the adapter with Watch 1 data.
3. The Resource Manager calls the ValidateProperties method of the adapter with Watch 2 data.
4. The Resource Manager creates a thread (T1) and calls the Connect method of the adapter object from Step 2.
5. (T1): The Resource Manager calls the OnNotify (MPIN_LISTEN_START) and then the Listen method with Watch 1 details.
6. The Resource Manager creates a thread (T2) and calls the Connect method of the adapter object from Step 3.
7. (T2): Resource Manager calls the OnNotify (MPIN_LISTEN_START) and then the Listen method with Watch 2 details.
8. The Resource Manager notifies the Launcher that the listeners started successfully. Now an event occurs.
9. (T1): The Listen method returns with an event, returning a message. (T1): The Resource Manager notifies the Launcher passing the message collection object.
10. (T1): The Resource Manager calls the Listen method again.
11. The map runs and the Resource Manager is called for input. However, the connection goes bad on Watch 2.
12. (T2): The Listen method returns MPIRC_BAD_CONNECTION.
13. (T2): The Resource Manager notifies the Launcher, which logs the error and notifies interested network management applications using SNMP. The Resource Manager then calls the OnNotify (MPI_LISTEN_STOP) method and the Disconnect method to clean up the connection.
14. (T2): The Resource Manager calls the Connect method again with the adapter object.
15. (T2): The Resource Manager calls the OnNotify (MPI_LISTEN_START) method and then the Listen method.

Listener scenario: Listener and map connection threads are combined

This is an example scenario in which an adapter requires that all activities related to an input event are carried out within the same thread.

This example describes a scenario where an adapter is requiring that all activities related to an input event are carried out within that same thread (for example, SAP BW).

1. A watch is to be added. The Resource Manager sees from the vtable adapter library that bListenThread is TRUE. It gets the listener thread count from the **config.yaml** configuration file. (Call this n).
2. The Resource Manager calls the ValidateProperties method with watch data.
3. The Resource Manager starts n threads, each of which calls the Connect, BeginTransaction, OnNotify (MPIN_LISTEN_START), and Listen methods.
4. When a Listen method returns messages in the message collection object, the Resource Manager calls the Launcher passing the connection context.
5. The Launcher calls the Resource Manager to initiate the Get method, passing the connection context back to the Resource Manager.
6. The Resource Manager communicates with the thread corresponding to the connection context, calling its Get method.
7. Possibly, other Get or Put methods are called during the processing of this map.
8. The EndTransaction method is called at the appropriate time.

9. When the map completes, the Resource Manager first calls the BeginTransaction method and then the Listen method once more on the same thread.

Threads, connections, and transactions overview

It is important to understand how threads, connections, and transactions interact.

You must separate threads for adapter connections and map logic. This allows connections to remain open even after maps complete. The Resource Manager maintains a pool of connection threads and associates one or more with a map thread during the execution of a map.

There is always a one-to-one relationship between a connection and a thread; one connection is associated with one thread. Multiple calls associated with one connection all occur on a single thread, that is, if the Connect method is called from thread n, all subsequent Listen, Get, Put, BeginTransaction, EndTransaction, and Disconnect calls for that connection will also be called from thread n.

Adapters fit into one of two categories:

- Those that have a one-to-one relationship between a connection and a transaction.
- Those that allow for multiple, concurrent transactions on the same connection.

The determination as to the category into which an adapter falls is normally made by the implementation of the transaction mechanism in the resource to which the adapter interfaces.

The adapter defines the mode in which it is operating by setting the nTransMode value in the vtable library to one of the following values:

- MPI_TRANSACTIONS_NONE
- MPI_TRANSACTIONS_SINGLE
- MPI_TRANSACTIONS_MULTIPLE

This parameter takes affects when the BeginTransaction and EndTransaction methods are called as follows:

- For adapters allowing for a single transaction per connection, there is only one call to BeginTransaction and EndTransaction.
- For adapters allowing for multiple transactions per connection, there will be multiple calls.
- For adapters with a transaction mode of NONE, there will be no calls to BeginTransaction or EndTransaction.

The following example shows the sequence of calls for a map that has two output cards, both with Scope setting set to Map, which can operate within the same connection:

Single transaction per connection	Multiple transactions per connection
Connect	Connect
BeginTransaction (1)	BeginTransaction (1)
Put (1)	Put (1)
CompareConnection --> MPI_CMP_EQUAL	CompareConnection --> MPI_CMP_EQUAL
Put (2)	BeginTransaction (2)
EndTransaction (1)	Put (2)
Disconnect	EndTransaction (1) EndTransaction (2) Disconnect

There is a requirement of some resources that all activity for a single resource occurs on a single thread. The capability to execute Listen, Get and Put with the same connection thread is possible.

Obviously, if a connection thread is being used during the execution of a map, it cannot be listening for the arrival of new events. Therefore, multiple listener threads must be started to enable concurrent execution and responsive handling of events. This adds a new constraint for the Resource Manager—the number of maps that can run concurrently to process a specific watch depends on the number of listener threads that have been created.

The **config.yaml** configuration file has a parameter (MaxWatchThreads), which is the maximum number of threads that can be created for connections for a given resource. This is used to determine the number of threads that will be started to listen for a given event if the adapter property MPIP_ADAPTER_LISTEN_USESAMECONN is TRUE.

COM API overview

The COM API enables applications to invoke maps remotely.

This API provides interfaces for running and controlling maps. It provides additional functionality, masks the need for visible structures, and removes the need to construct command lines.

The COM API is based on Map, Card, Adapter, and Stream objects. Card, Adapter, and Stream objects are used for source and target overrides.

Adapter objects interface with the adapter API. Stream objects eliminate the need for large, contiguous, memory buffers for passing data directly to a map object.

- [COM API interfaces overview](#)
The COM API uses interfaces for map, card, adapter and stream objects.
- [COM API object hierarchy](#)
The COM API objects exist in a defined hierarchy.

COM API interfaces overview

The COM API uses interfaces for map, card, adapter and stream objects.

- [IMFactory interface overview](#)
The IMFactory interface initializes or uninitialized the TX Programming Interface and creates a map object.
- [IMMap interface overview](#)
The IMMap method creates a map object instance from the byte array representing a compiled map file. This method creates hierarchies of map, card and adapter objects.
- [IMCard interface overview](#)
The IMCard interface uses a card object from the TX Programming Interface that represents an input or output card of a map in program memory.
- [IMAdapter interface overview](#)
The IMAdapter interface uses an adapter object from the TX Programming Interface that represents a resource adapter.
- [IMStream interface overview](#)
The IMStream interface uses a Stream object from the TX Programming Interface that represents an input or output card of a map in program memory.
- [IMStreamPage interface overview](#)
The IMStreamPage interface uses a StreamPage object from the TX Programming Interface that represents an input or output card of a map in program memory.

IMFactory interface overview

The IMFactory interface initializes or uninitialized the TX Programming Interface and creates a map object.

- [IMFactory virtual table order](#)
The IMFactory virtual table (vtable) lists all the methods for this interface.

IMFactory virtual table order

The IMFactory virtual table (vtable) lists all the methods for this interface.

- [IMFactory::InitMercAPI method](#)
The **IMFactory::InitMercAPI** method initializes the IBM Transformation Extender environment.
- [IMFactory::ExitMercAPI method](#)
The **IMFactory::ExitMercAPI** method cleans up the IBM Transformation Extender environment.
- [IMFactory::MapLoadMemory method](#)
The **IMFactory::MapLoadMemory** method is used with the IMFactory interface to load map memory.
- [IMFactory::MapReloadMemory method](#)
The **IMFactory::MapReloadMemory** method is used with the IMFactory interface to reload map memory.
- [IMFactory::MapLoadFile method](#)
The **IMFactory::MapLoadFile** method is used with the IMFactory interface to load a map file.
- [IMFactory::MapReloadFile method](#)
The **IMFactory::MapReloadFile** method is used with the IMFactory interface to reload a map file.

IMFactory::InitMercAPI method

The **IMFactory::InitMercAPI** method initializes the IBM Transformation Extender environment.

It should be called prior to any other call and only one time for each process. The bstrConfigFile can be passed in as `NULL` as long as name resolution is not required.

Quick info

```
HRESULT InitMercAPI(BSTR bstrConfigFile)
```

Parameters

bstrConfigFile
[in] - name of the configuration file used to determine the resource files used

Return values

The following value is returned when this method is called:

`S_OK`
Success status

IMFactory::ExitMercAPI method

The **IMFactory::ExitMercAPI** method cleans up the IBM Transformation Extender environment.

It should be called after any other call and only one time for each process.

Quick info

```
HRESULT ExitMercAPI()
```

Return values

The following value is returned when this method is called:

S_OK

Success status

IMFactory::MapLoadMemory method

The **IMFactory::MapLoadMemory** method is used with the IMFactory interface to load map memory.

Quick info

```
HRESULT MapLoadMemory([in] BSTR bstrIdentifier, [in] BSTR bstrWorkingDirectory,  
[in] VARIANT byteArray, [in] long sSize)
```

Parameters

bstrIdentifier

[in] - text string used to identify the compiled map

bstrWorkingDirectory

[in] - directory that should be used as the working directory for the map

byteArray

[in] - bytes that comprise a compiled map

sSize

[in] - number of bytes pointed to pcCompiledMap

Return values

The following value is returned when this method is called:

S_OK

Success status

IMFactory::MapReloadMemory method

The **IMFactory::MapReloadMemory** method is used with the IMFactory interface to reload map memory.

Quick info

```
HRESULT MapReloadMemory([in] BSTR bstrIdentifier,  
[in] BSTR bstrWorkingDirectory, [in] VARIANT byteArray, [in] long sSize)
```

Parameters

bstrIdentifier

[in] - text string used to identify the compiled map

bstrWorkingDirectory

[in] - directory that should be used as the working directory for the map

byteArray

[in] - bytes that comprise a compiled map

sSize

[in] - number of bytes pointed to pcCompiledMap

Return values

The following value is returned when this method is called:

S_OK

Success status

IMFactory::MapLoadFile method

The **IMFactory::MapLoadFile** method is used with the IMFactory interface to load a map file.

Quick info

```
HRESULT MapLoadFile(BSTR lpszMapFileName, IIMMap** pMMMap);
```

Parameters

lpszMapFileName
[in] - name of the map file (MMC)
pMMMap
[out, retval] - handle to map interface

Return values

The following value is returned when this method is called:

S_OK
Success status

IMFactory::MapReloadFile method

The **IMFactory::MapReloadFile** method is used with the IMFactory interface to reload a map file.

Quick info

```
HRESULT MapReloadFile(BSTR lpszMapFileName, IIMMap** pMMMap);
```

Parameters

lpszMapFileName
[in] - name of the map file (MMC)
pMMMap
[out, retval] - handle to map interface

Return values

The following value is returned when this method is called:

S_OK
Success status

IMMap interface overview

The IMMap method creates a map object instance from the byte array representing a compiled map file. This method creates hierarchies of map, card and adapter objects.

Client applications call the methods of the IMMap to do the following map functions:

- load
- control
- run

The map object is created when a map is loaded from a compiled map (.mmc) file or from memory. The TX Programming Interface caches maps. Therefore, if multiple map objects are created for the same map, the map file is read only one time.

A map object is reentrant. After a map object has been created, the map can be run an infinite number of times without any need for the map to be reloaded.

Multiple threads cannot reference the same map object concurrently. If multiple threads need to execute the same map concurrently, a separate map object should be obtained for each thread.

The identifier is used to identify the specific map. This might be the name of the map or any other identifier that the application will use. Subsequent calls to this method with the same identifier will not introduce additional processing of the compiled map. Therefore, after the initial call to this method, the memory consumed by **compiledMap** might be freed.

- [IMMap virtual table order](#)
The IMMap virtual table (vtable) lists all the methods for this interface.
- [IMMap properties](#)
The IMMap method creates a map object instance from the byte array representing a compiled map file. This method creates hierarchies of map, card and adapter objects. The map object supports the following types of properties:

IMMap virtual table order

The IMMap virtual table (vtable) lists all the methods for this interface.

- **[IMMap::Run method](#)**
The **IMMap::Run** method is used with the IMMap interface to run a map.
- **[IMMap::Refresh method](#)**
The **IMMap::Refresh** method is used with the IMMap interface.
- **[IMMap::Abort method](#)**
The **IMMap::Abort** method is used with the IMMap interface.
- **[IMMap::Pause method](#)**
The **IMMap::Pause** method is used with the IMMap interface.
- **[IMMap::Continue method](#)**
The **IMMap::Continue** method is used with the IMMap interface.
- **[IMMap::GetInputCard method](#)**
The **IMMap::GetInputCard** method is used with the IMMap interface.
- **[IMMap::GetOutputCard method](#)**
The **IMMap::GetOutputCard** method is used with the IMMap interface.
- **[IMMap::MapSetupNotifications method](#)**
The **IMMap::MapSetupNotifications** method is used with the IMMap interface.
- **[IMMap::GetInputCardCount method](#)**
The **IMMap::GetInputCardCount** method is used with the IMMap interface.
- **[IMMap::GetOutputCardCount method](#)**
The **IMMap::GetOutputCardCount** method is used with the IMMap interface.
- **[IMMap::GetPropertyCount method](#)**
The **IMMap::GetPropertyCount** method is used with the IMMap interface.
- **[IMMap::MapUnload method](#)**
The **IMMap::MapUnload** method is used with the IMMap interface.
- **[IMMap::GetIntegerProperty method](#)**
The **IMMap::GetIntegerProperty** method is used with the IMMap interface.
- **[IMMap::SetIntegerProperty method](#)**
The **IMMap::SetIntegerProperty** method is used with the IMMap interface.
- **[IMMap::GetTextProperty method](#)**
The **IMMap::GetTextProperty** method is used with the IMMap interface.
- **[IMMap::SetTextProperty method](#)**
The **IMMap::SetTextProperty** method is used with the IMMap interface.
- **[IMMap::GetBinaryProperty method](#)**
The **IMMap::GetBinaryProperty** method is used with the IMMap interface.
- **[IMMap::SetBinaryProperty method](#)**
The **IMMap::SetBinaryProperty** method is used with the IMMap interface.
- **[IMMap::GetPropertiesXML method](#)**
The **IMMap::GetPropertiesXML** method is used with the IMMap interface.
- **[IMMap::SetPropertiesXML method](#)**
The **IMMap::SetPropertiesXML** method is used with the IMMap interface.
- **[IMMap::GetAllPropertiesXML method](#)**
The **IMMap::GetAllPropertiesXML** method is used with the IMMap interface.

IMMap::Run method

The **IMMap::Run** method is used with the IMMap interface to run a map.

Quick info

HRESULT Run ()

Return values

The following value is returned when this method is called:

S_OK
Success status

IMMap::Refresh method

The **IMMap::Refresh** method is used with the IMMap interface.

Quick info

HRESULT Refresh ()

Return values

The following value is returned when this method is called:

S_OK
Success status

IMMap::Abort method

The **IMMap::Abort** method is used with the IMMap interface.

Quick info

```
HRESULT Abort()
```

Return values

The following value is returned when this method is called:

S_OK
Success status

IMMap::Pause method

The **IMMap::Pause** method is used with the IMMap interface.

Quick info

```
HRESULT Pause()
```

Return values

The following value is returned when this method is called:

S_OK
Success status

IMMap::Continue method

The **IMMap::Continue** method is used with the IMMap interface.

Quick info

```
HRESULT Continue()
```

Return values

The following value is returned when this method is called:

S_OK
Success status

IMMap::GetInputCard method

The **IMMap::GetInputCard** method is used with the IMMap interface.

Quick info

```
HRESULT GetInputCard([in] long sCardNum, [out, retval] IMCard **pIMCard)
```

Parameters

sCardNum
[in] - number of the input card (starting at 1)

pIMCard
[out, retval] - handle to the card interface

Return values

The following value is returned when this method is called:

S_OK
Success status

IMMap::GetOutputCard method

The **IMMap::GetOutputCard** method is used with the IMMap interface.

Quick info

```
HRESULT GetOutputCard([in] long sCardNum, [out, retval] IMCARD **pIMCard)
```

Parameters

sCardNum
[in] - number of the output card (starting at 1)

pIMCard
[out, retval] - handle to the card interface

Return values

The following value is returned when this method is called:

S_OK
Success status

IMMap::MapSetupNotifications method

The **IMMap::MapSetupNotifications** method is used with the IMMap interface.

Quick info

```
HRESULT MapSetupNotifications (long sNotifyWhen)
```

Parameters

sNotifyWhen
[in] - bitmask indicating the events of which the registrator wished to be notified

Return values

The following value is returned when this method is called:

S_OK
Success status

IMMap::GetInputCardCount method

The **IMMap::GetInputCardCount** method is used with the IMMap interface.

Quick info

```
HRESULT GetInputCardCount ([out, retval] long* InputCardCount)
```

Parameters

InputCardCount
[out, retval] - address of the integer variable to receive the card count

Return values

The following value is returned when this method is called:

S_OK
Success status

IMMap::GetOutputCardCount method

The **IMMap::GetOutputCardCount** method is used with the IMMap interface.

Quick info

```
HRESULT GetOutputCardCount(long* pOutputCardCount)
```

Parameters

pOutputCardCount
[out, retval] - address of the integer variable to receive the card count

Return values

The following value is returned when this method is called:

S_OK
Success status

IMMap::GetPropertyCount method

The **IMMap::GetPropertyCount** method is used with the IMMap interface.

Quick info

```
HRESULT GetPropertyCount(long lProperty, long *pValue)
```

Parameters

lProperty
[in] - property ID
pValue
[out, retval] - count of the property values

Return values

The following value is returned when this method is called:

S_OK
Success status

IMMap::MapUnload method

The **IMMap::MapUnload** method is used with the IMMap interface.

Quick info

```
HRESULT MapUnload()
```

Return values

The following value is returned when this method is called:

S_OK
Success status

IMMap::GetIntegerProperty method

The **IMMap::GetIntegerProperty** method is used with the IMMap interface.

Quick info

```
HRESULT GetIntegerProperty([in] long sProperty,  
                           [in] long sIndex, [out, retval] long *pVal)
```

Parameters

sProperty
[in] - property ID

sIndex
[in] - index of the property

pVal
[out, retval] - integer value of the property

Return values

The following value is returned when this method is called:

S_OK
Success status

IMMap::SetIntegerProperty method

The **IMMap::SetIntegerProperty** method is used with the IMMap interface.

Quick info

```
HRESULT SetIntegerProperty([in] long sProperty, [in] long sIndex, [in] long newVal)
```

Parameters

sProperty
[in] - property ID

sIndex
[in] - index of the property

newVal
[in] - integer value of the property

Return values

The following value is returned when this method is called:

S_OK
Success status

IMMap::GetTextProperty method

The **IMMap::GetTextProperty** method is used with the IMMap interface.

Quick info

```
HRESULT GetTextProperty(LONG sProperty, LONG sIndex, BSTR* pVal)
```

Parameters

sProperty
[in] - property ID

sIndex
[in] - index of the property

pVal
[out, retval] - text property

Return values

The following value is returned when this method is called:

S_OK
Success status

IMMap::SetTextProperty method

The **IMMap::SetTextProperty** method is used with the IMMap interface.

Quick info

```
HRESULT SetTextProperty(long sProperty, long sIndex, BSTR newVal)
```

Parameters

sProperty
[in] - property ID

sIndex
[in] - index of the property

pVal
[out, retval] - text value of the property

Return values

The following value is returned when this method is called:

S_OK
Success status

IMMap::GetBinaryProperty method

The **IMMap::GetBinaryProperty** method is used with the IMMap interface.

Quick info

```
HRESULT GetBinaryProperty([in] long sProperty,  
                         [in] long sIndex, [out] VARIANT *pVal)
```

Parameters

sProperty
[in] - property ID

sIndex
[in] - index of the property

pVal
[out] - binary value of the property

Return values

The following value is returned when this method is called:

S_OK
Success status

IMMap::SetBinaryProperty method

The **IMMap::SetBinaryProperty** method is used with the IMMap interface.

Quick info

```
HRESULT SetIntegerProperty([in] long sProperty, [in] long sIndex, [in] long newVal)
```

Parameters

sProperty
[in] - property ID

sIndex
[in] - index of the property

newVal
[out, retval] - binary value of the property

Return values

The following value is returned when this method is called:

S_OK
Success status

IMMap::GetPropertiesXML method

The **IMMap::GetPropertiesXML** method is used with the IMMap interface.

Quick info

```
HRESULT GetPropertiesXML(long* idArray, long cmax, BSTR* xmlProperties)
```

Parameters

idArray
[in] - an integer array which lists the property IDs that are to be returned. For indexed type properties all index values are returned
cmax
[in] - array length
xmlProperties
[out, retval] - populated XML property document. Each property that was requested is returned in the XML document.

Return values

The following value is returned when this method is called:

S_OK
Success status

IMMap::SetPropertiesXML method

The **IMMap::SetPropertiesXML** method is used with the IMMap interface.

Quick info

```
HRESULT SetPropertiesXML(BSTR xmlProperties)
```

Parameters

xmlProperties
[in] - populated XML property document

Return values

The following value is returned when this method is called:

S_OK
Success status

IMMap:: GetAllPropertiesXML method

The **IMMap:: GetAllPropertiesXML** method is used with the IMMap interface.

Quick info

```
HRESULT GetAllPropertiesXML(BOOL bRecurse, BSTR* xmlProperties)
```

Parameters

bRecurse
[in] - If false then only the properties of the object will be returned. If true then properties of related objects will also be returned. Specifically:

- If the object is an IMCard, then the properties of the IMCard and the related IMAdapter object will be returned.
- If the object is an IMMap, then the properties of the IMMap, all of its associated IMCard objects and the related IMAdapter objects will be returned.

xmlProperties
[out, retval] - populated XML property document. Each property that was requested is returned in the XML document.

Return values

The following value is returned when this method is called:

S_OK
Success status

IMMap properties

The IMMap method creates a map object instance from the byte array representing a compiled map file. This method creates hierarchies of map, card and adapter objects. The map object supports the following types of properties:

- [General](#)
- [Audit](#)
- [Trace](#)
- [Workspace](#)
- [Sliding century](#)
- [Custom validation](#)
- [Map retry](#)

IMCard interface overview

The IMCard interface uses a card object from the TX Programming Interface that represents an input or output card of a map in program memory.

- [IMCard virtual table order](#)
The IMCard virtual table (vtable) lists all the methods for the IMCard interface.
- [IMCard properties](#)
The IMCard object supports the following types of properties:

IMCard virtual table order

The IMCard virtual table (vtable) lists all the methods for the IMCard interface.

- [IMCard::GetAdapter](#)
The **IMCard::GetAdapter** method is used with the IMCard interface.
- [IMCard::GetAdapterType](#)
The **IMCard::GetAdapterType** method is used with the IMCard interface.
- [IMCard::OverrideAdapter](#)
The **IMCard::OverrideAdapter** method is used with the IMCard interface.
- [IMCard::GetMap](#)
The **IMCard::GetMap** method is used with the IMCard interface.
- [IMCard::GetIntegerProperty](#)
The **IMCard::GetIntegerProperty** method is used with the IMCard interface.
- [IMCard::SetIntegerProperty](#)
The **IMCard::SetIntegerProperty** method is used with the IMCard interface.
- [IMCard::GetTextProperty](#)
The **IMCard::GetTextProperty** method is used with the IMCard interface.
- [IMCard::SetTextProperty](#)
The **IMCard::SetTextProperty** method is used with the IMCard interface.
- [IMCard::GetBinaryProperty](#)
The **IMCard::GetBinaryProperty** method is used with the IMCard interface.
- [IMCard::SetBinaryProperty](#)
The **IMCard::SetBinaryProperty** method is used with the IMCard interface.
- [IMCard::GetPropertiesXML](#)
The **IMCard::GetPropertiesXML** method is used with the IMCard interface.
- [IMCard::SetPropertiesXML](#)
The **IMCard::SetPropertiesXML** method is used with the IMCard interface.
- [IMCard::GetAllPropertiesXML](#)
The **IMCard::GetAllPropertiesXML** method is used with the IMCard interface.

IMCard::GetAdapter

The **IMCard::GetAdapter** method is used with the IMCard interface.

Quick info

```
HRESULT GetAdapter(IMAdapter **pIMAdapter)
```

Parameters

pIMAdapter

[ret, out] - handle to the adapter interface

Return values

The following value is returned when this method is called:

S_OK
Success status

IMCard::GetAdapterType

The **IMCard::GetAdapterType** method is used with the IMCard interface.

Quick info

```
HRESULT GetAdapterType(long *pVal)
```

Parameters

pVal
[out, retval] - type of the adapter

Return values

The following value is returned when this method is called:

S_OK
Success status

IMCard::OverrideAdapter

The **IMCard::OverrideAdapter** method is used with the IMCard interface.

Quick info

```
HRESULT OverrideAdapter(BSTR bstrAlias, long sType)
```

Parameters

bstrAlias
[in] - alias name of the adapter
sType
[in] - numeric type of the adapter

Return values

The following value is returned when this method is called:

S_OK
Success status

IMCard::GetMap

The **IMCard::GetMap** method is used with the IMCard interface.

Quick info

```
HRESULT GetMap(IMMap **pMMap)
```

Parameters

pMMap
[out, retval] - handle to the map interface

Return values

The following value is returned when this method is called:

S_OK
Success status

IMCard::GetIntegerProperty

The **IMCard::GetIntegerProperty** method is used with the IMCard interface.

Quick info

```
HRESULT GetIntegerProperty([in] long sProperty,  
                           [in] long sIndex, [out, retval] long *pVal)
```

Parameters

sProperty
[in] - property ID

sIndex
[in] - index of the property

pVal
[out, retval] - integer value of the property

Return values

The following value is returned when this method is called:

S_OK
Success status

IMCard::SetIntegerProperty

The **IMCard::SetIntegerProperty** method is used with the IMCard interface.

Quick info

```
HRESULT SetIntegerProperty([in] long sProperty, [in] long sIndex, [in] long newVal)
```

Parameters

sProperty
[in] - property ID

sIndex
[in] - index of the property

newVal
[in] - integer value of the property

Return values

The following value is returned when this method is called:

S_OK
Success status

IMCard::GetTextProperty

The **IMCard::GetTextProperty** method is used with the IMCard interface.

Quick info

```
HRESULT GetTextProperty(LONG sProperty, LONG sIndex, BSTR* pVal)
```

Parameters

sProperty
[in] - property ID

sIndex
[in] - index of the property
pVal
[out, retval] - text property

Return values

The following value is returned when this method is called:

S_OK
Success status

IMCard::SetTextProperty

The **IMCard::SetTextProperty** method is used with the IMCard interface.

Quick info

```
HRESULT SetTextProperty(long sProperty, long sIndex, BSTR newVal)
```

Parameters

sProperty
[in] - property ID
sIndex
[in] - index of the property
pVal
[out, retval] - text value of the property

Return values

The following value is returned when this method is called:

S_OK
Success status

IMCard::GetBinaryProperty

The **IMCard::GetBinaryProperty** method is used with the IMCard interface.

Quick info

```
HRESULT GetBinaryProperty([in] long sProperty,  
                         [in] long sIndex, [out] VARIANT *pVal)
```

Parameters

sProperty
[in] - property ID
sIndex
[in] - index of the property
pVal
[out] - binary value of the property

Return values

The following value is returned when this method is called:

S_OK
Success status

IMCard::SetBinaryProperty

The **IMCard::SetBinaryProperty** method is used with the IMCard interface.

Quick info

```
HRESULT SetIntegerProperty([in] long sProperty, [in] long sIndex, [in] long newVal)
```

Parameters

sProperty
[in] - property ID

sIndex
[in] - index of the property

newVal
[out, retval] - binary value of the property

Return values

The following value is returned when this method is called:

S_OK
Success status

IMCard::GetPropertiesXML

The **IMCard::GetPropertiesXML** method is used with the IMCard interface.

Quick info

```
HRESULT GetPropertiesXML(long* idArray, long cmax, BSTR* xmlProperties)
```

Parameters

idArray
[in] - an integer array which lists the property IDs that are to be returned. For indexed type properties all index values are returned
cmax
[in] - array length
xmlProperties
[out, retval] - populated XML property document. Each property that was requested is returned in the XML document.

Return values

The following value is returned when this method is called:

S_OK
Success status

IMCard::SetPropertiesXML

The **IMCard::SetPropertiesXML** method is used with the IMCard interface.

Quick info

```
HRESULT SetPropertiesXML(BSTR xmlProperties)
```

Parameters

xmlProperties
[in] - populated XML property document

Return values

The following value is returned when this method is called:

S_OK
Success status

IMCard::GetAllPropertiesXML

The **IMCard::GetAllPropertiesXML** method is used with the IMCard interface.

Quick info

```
HRESULT GetAllPropertiesXML(BOOL bRecurse, BSTR* xmlProperties)
```

Parameters

bRecurse

[in] - If false then only the properties of the object will be returned. If true then properties of related objects will also be returned. Specifically:

- If the object is an IMCard, then the properties of the IMCard and the related IMAdapter object will be returned.
- If the object is an IMMap, then the properties of the IMMap, all of its associated IMCard objects and the related IMAdapter objects will be returned.

xmlProperties

[out, retval] - populated XML property document. Each property that was requested is returned in the XML document.

Return values

The following value is returned when this method is called:

S_OK

Success status

IMCard properties

The IMCard object supports the following types of properties:

- [General](#)
- [Backup](#)

IMAdapter interface overview

The IMAdapter interface uses an adapter object from the TX Programming Interface that represents a resource adapter.

Defined methods allow the adapter properties to be modified.

- [IMAdapter virtual table order](#)
The IMAdapter virtual table (vtable) lists all the methods for this interface.
- [IMAdapter properties](#)
The IMAdapter object supports the following types of properties:

IMAdapter virtual table order

The IMAdapter virtual table (vtable) lists all the methods for this interface.

- [IMAdapter::GetCard](#)
The IMAdapter::GetCard method is used with the IMAdapter interface.
- [IMAdapter::GetStream](#)
The IMAdapter::GetStream method returns the stream object associated with the adapter object.
- [IMAdapter::GetInputStream](#)
The IMAdapter::GetInputStream method is used with the IMAdapter interface.
- [IMAdapter::GetOutputStream](#)
The IMAdapter::GetOutputStream method is used with the IMAdapter interface.
- [IMAdapter::GetIntegerProperty](#)
The IMAdapter::GetIntegerProperty method is used with the IMAdapter interface.
- [IMAdapter::SetIntegerProperty](#)
The IMAdapter::SetIntegerProperty method is used with the IMAdapter interface.
- [IMAdapter::GetTextProperty](#)
The IMAdapter::GetTextProperty method is used with the IMAdapter interface.
- [IMAdapter::SetTextProperty](#)
The IMAdapter::SetTextProperty method is used with the IMAdapter interface.
- [IMAdapter::GetBinaryProperty](#)
The IMAdapter::GetBinaryProperty method is used with the IMAdapter interface.
- [IMAdapter::SetBinaryProperty](#)
The IMAdapter::SetBinaryProperty method is used with the IMAdapter interface.
- [IMAdapter::GetPropertiesXML](#)
The IMAdapter::GetPropertiesXML method is used with the IMAdapter interface.
- [IMAdapter::SetPropertiesXML](#)
The IMAdapter::SetPropertiesXML method is used with the IMAdapter interface.
- [IMAdapter::GetAllPropertiesXML](#)
The IMAdapter::GetAllPropertiesXML method is used with the IMAdapter interface.

IMAdapter::GetCard

The **IMAdapter::GetCard** method is used with the IMAdapter interface.

Quick info

```
HRESULT GetCard(IMCard **pMCARD)
```

Parameters

pMCARD
[out, retval] - handle to card interface

Return values

The following value is returned when this method is called:

S_OK
Success status

IMAdapter::GetStream

The **IMAdapter::GetStream** method returns the stream object associated with the adapter object.

Quick info

```
HRESULT GetStream(long sProperty, long sIndex, IMStream **pIMStream)
```

Parameters

sProperty
[in] - property ID
sIndex
[in] - index of the property
pIMStream
[out,retval] - handle to stream object

Return values

The following value is returned when this method is called:

S_OK
Success status

IMAdapter::GetInputStream

The **IMAdapter::GetInputStream** method is used with the IMAdapter interface.

Quick info

```
HRESULT GetInputStream(IMStream **pIMStream)
```

Parameters

pIMStream
[out,retval] - handle to input stream object

Return values

The following value is returned when this method is called:

S_OK
Success status

IMAdapter::GetOutputStream

The **IMAdapter::GetOutputStream** method is used with the IMAdapter interface.

Quick info

```
HRESULT GetOutputStream(IMStream **pIMStream)
```

Parameters

pIMStream
[out, retval] - handle to output stream object

Return values

The following value is returned when this method is called:

S_OK
Success status

IMAdapter::GetIntegerProperty

The **IMAdapter::GetIntegerProperty** method is used with the IMAdapter interface.

Quick info

```
HRESULT GetIntegerProperty([in] long sProperty,  
                           [in] long sIndex, [out, retval] long *pVal)
```

Parameters

sProperty
[in] - property ID

sIndex
[in] - index of the property
pVal
[out, retval] - integer value of the property

Return values

The following value is returned when this method is called:

S_OK
Success status

IMAdapter::SetIntegerProperty

The **IMAdapter::SetIntegerProperty** method is used with the IMAdapter interface.

Quick info

```
HRESULT SetIntegerProperty([in] long sProperty, [in] long sIndex, [in] long newVal)
```

Parameters

sProperty
[in] - property ID

sIndex
[in] - index of the property
newVal
[in] - integer value of the property

Return values

The following value is returned when this method is called:

S_OK
Success status

IMAdapter::GetTextProperty

The **IMAdapter::GetTextProperty** method is used with the IMAdapter interface.

Quick info

```
HRESULT GetTextProperty(LONG sProperty, LONG sIndex, BSTR* pVal)
```

Parameters

sProperty
[in] - property ID

sIndex
[in] - index of the property

pVal
[out, retval] - text property

Return values

The following value is returned when this method is called:

S_OK
Success status

IMAdapter::SetTextProperty

The **IMAdapter::SetTextProperty** method is used with the IMAdapter interface.

Quick info

```
HRESULT SetTextProperty(long sProperty, long sIndex, BSTR newVal)
```

Parameters

sProperty
[in] - property ID

sIndex
[in] - index of the property

pVal
[out, retval] - text value of the property

Return values

The following value is returned when this method is called:

S_OK
Success status

IMAdapter::GetBinaryProperty

The **IMAdapter::GetBinaryProperty** method is used with the IMAdapter interface.

Quick info

```
HRESULT GetBinaryProperty([in] long sProperty,  
                           [in] long sIndex, [out] VARIANT *pVal)
```

Parameters

sProperty
[in] - property ID

sIndex
[in] - index of the property

pVal
[out] - binary value of the property

Return values

The following value is returned when this method is called:

S_OK
Success status

IMAdapter::SetBinaryProperty

The **IMAdapter::SetBinaryProperty** method is used with the IMAdapter interface.

Quick info

```
HRESULT SetIntegerProperty([in] long sProperty, [in] long sIndex, [in] long newVal)
```

Parameters

sProperty
[in] - property ID

sIndex
[in] - index of the property
newVal
[out, retval] - binary value of the property

Return values

The following value is returned when this method is called:

S_OK
Success status

IMAdapter::GetPropertiesXML

The **IMAdapter::GetPropertiesXML** method is used with the IMAdapter interface.

Quick info

```
HRESULT GetPropertiesXML(long* idArray, long cmax, BSTR* xmlProperties)
```

Parameters

idArray
[in] - an integer array which lists the property IDs that are to be returned. For indexed type properties all index values are returned
cmax
[in] - array length
xmlProperties
[out, retval] - populated XML property document. Each property that was requested is returned in the XML document.

Return values

The following value is returned when this method is called:

S_OK
Success status

IMAdapter::SetPropertiesXML

The **IMAdapter::SetPropertiesXML** method is used with the IMAdapter interface.

Quick info

```
HRESULT SetPropertiesXML(BSTR xmlProperties)
```

Parameters

xmlProperties

[in] - populated XML property document

Return values

The following value is returned when this method is called:

S_OK
Success status

IMAdapter::GetAllPropertiesXML

The **IMAdapter::GetAllPropertiesXML** method is used with the IMAdapter interface.

Quick info

```
HRESULT GetAllPropertiesXML(BOOL bRecurse, BSTR* xmlProperties)
```

Parameters

bRecurse

[in] - If false then only the properties of the object will be returned. If true then properties of related objects will also be returned.

xmlProperties

[out, retval] - properties are specified in the form of an XML document. The XML Schema for this is mercpi_properties.xsd. Both the request document, which specifies which properties to return, and the returned document comply to this XML Schema.

Return values

The following value is returned when this method is called:

S_OK
Success status

IMAdapter properties

The IMAdapter object supports the following types of properties:

- [General](#)
- [On Success](#)
- [On Failure](#)
- [Retry](#)
- [Scope](#)
- [Warnings](#)
- [Aggregation](#)

IMStream interface overview

The IMStream interface uses a Stream object from the TX Programming Interface that represents an input or output card of a map in program memory.

- [IMStream virtual table order](#)
The IMStream virtual table (vtable) lists all the methods for this interface.

IMStream virtual table order

The IMStream virtual table (vtable) lists all the methods for this interface.

- [IMStream::Seek](#)
The **IMStream::Seek** method moves the stream cursor to the specified offset.
- [IMStream::Tell](#)
The **IMStream::Tell** method retrieves the current cursor absolute position (relative to the start).
- [IMStream::IsEnd](#)
The **IMStream::IsEnd** method checks if the cursor is at the stream end.
- [IMStream::GetSize](#)
The **IMStream::GetSize** method returns the current size of the stream.
- [IMStream::SetSize](#)
The **IMStream::SetSize** method sets the new stream size.
- [IMStream::Write](#)
The **IMStream::Write** method writes a portion of an array of bytes.

- [**IMStream::Modify**](#)
The **IMStream::Modify** method modifies stream settings.
- [**IMStream::Read**](#)
The **IMStream::Read** method reads bytes into a portion of a byte array.
- [**IMStream::Flush**](#)
The **IMStream::Flush** method forces the current data in memory to be written to file.
- [**IMStream::ReadPage**](#)
The **IMStream::ReadPage** method returns the **StreamPage** object associated with the specified stream.
- [**IMStream::NewStreamPage**](#)
The **IMStream::NewStreamPage** method creates a new stream page object.
- [**IMStream::DeletePage**](#)
The **IMStream::DeletePage** method deletes the stream page object associated with the stream.

IMStream::Seek

The **IMStream::Seek** method moves the stream cursor to the specified offset.

Quick info

```
HRESULT Seek(long lPos, long lOrigin)
```

Parameters

- lPos*
[in] - the offset from the origin
lOrigin
[in] - origin to which *iPos* is relative

Return values

The following value is returned when this method is called:

- S_OK
Success status

IMStream::Tell

The **IMStream::Tell** method retrieves the current cursor absolute position (relative to the start).

Quick info

```
HRESULT Tell([out, retval] long *pVal)
```

Parameters

- pVal*
[out, retval] - cursor position

Return values

The following value is returned when this method is called:

- S_OK
Success status

IMStream::IsEnd

The **IMStream::IsEnd** method checks if the cursor is at the stream end.

Quick info

```
HRESULT IsEnd(BOOL *pVal)
```

Parameters

- pVal*
[out, retval] - Is *true* if the cursor is at stream end. Is *false* if the cursor is not at stream end.

Return values

The following value is returned when this method is called:

S_OK

Success status

IMStream::GetSize

The **IMStream::GetSize** method returns the current size of the stream.

Quick info

```
HRESULT GetSize([out, retval] long *pVal)
```

Parameters

pVal

[out, retval] - stream size (in bytes)

Return values

The following value is returned when this method is called:

S_OK

Success status

IMStream::SetSize

The **IMStream::SetSize** method sets the new stream size.

If the new size is less than the old size, the stream is truncated to the specified size. If the new size is greater than the old one the stream is appended with the appropriate number of zeros to occupy the new size in bytes.

Quick info

```
HRESULT SetSize(long lVal)
```

Parameters

iVal

[in] - new size

Return values

The following value is returned when this method is called:

S_OK

Success status

IMStream::Write

The **IMStream::Write** method writes a portion of an array of bytes.

Quick info

```
HRESULT Write(VARIANT vData, long lSize)
```

Parameters

vData

[in] - array of bytes

lSize

[in] - number of bytes to write

Return values

The following value is returned when this method is called:

S_OK
Success status

IMStream::Modify

The **IMStream::Modify** method modifies stream settings.

The stream data is written to memory until it reaches a threshold and then it is written to a file. The created file can be temporary or explicit. A temporary file can be created in the system temp directory or in a specified one. If both the fileName and dir are empty strings or null, a temporary file is created in the system's temp dir. If only the fileName is empty (null), a temporary file is created in the specified directory. If the fileName is specified, dir is ignored.

Quick info

```
HRESULT Modify(long lMaxMem, BSTR bstrDirectory, BSTR bstrFileName,  
VARIANT_BOOL bKeepFile)
```

Parameters

lMaxMem
[in] - threshold size for stream. After the stream exceeds this size, it will begin to be flushed to file.
bstrDirectory
[in] - directory in which to create the temporary file
bstrFileName
[in] - explicit file name path
bKeepFile
[in] - If it is `true`, the file is preserved after the stream is destroyed. If it is `false`, the file is deleted.

Return values

The following value is returned when this method is called:

S_OK
Success status

IMStream::Read

The **IMStream::Read** method reads bytes into a portion of a byte array.

The maximum number of bytes that the method reads is specified by the *lsize* parameter.

Quick info

```
HRESULT Read(long lSize, VARIANT *pvData)
```

Parameters

lSize
[in] - number of bytes to read
pvData
[out] - buffer to receive bytes. The method reads as many bytes as will fit in the array. The actual number of bytes read is returned.

Return values

The following value is returned when this method is called:

S_OK
Success status

IMStream::Flush

The **IMStream::Flush** method forces the current data in memory to be written to file.

Quick info

```
HRESULT Flush()
```

Return values

The following value is returned when this method is called:

S_OK
Success status

IMStream::ReadPage

The **IMStream::ReadPage** method returns the **StreamPage** object associated with the specified stream.

Quick info

```
HRESULT ReadPage(IMStreamPage **pIStreamPage)
```

Parameters

pIStreamPage
[out, retval] - handle to the stream page

Return values

The following value is returned when this method is called:

S_OK
Success status

IMStream::NewStreamPage

The **IMStream::NewStreamPage** method creates a new stream page object.

Quick info

```
HRESULT NewStreamPage(VARIANT vData, long lSize, VARIANT_BOOL bCopy,  
IMStreamPage** pIStreamPage)
```

Parameters

vData
[in] - data to the stream in bytes
lSize
[in] - size of the new stream
bCopy
[in] - copies the data to the new stream page

Return values

The following value is returned when this method is called:

S_OK
Success status

IMStream::DeletePage

The **IMStream::DeletePage** method deletes the stream page object associated with the stream.

Quick info

```
HRESULT DeletePage(IMStreamPage* pIStreamPage)
```

Parameters

pIStreamPage
[in] - StreamPage interface handle

Return values

The following value is returned when this method is called:

S_OK
Success status

IMStreamPage interface overview

The IMStreamPage interface uses a StreamPage object from the TX Programming Interface that represents an input or output card of a map in program memory.

- [IMStreamPage virtual table order](#)
The IMStreamPage virtual table (vtable) lists all the methods for this interface.

IMStreamPage virtual table order

The IMStreamPage virtual table (vtable) lists all the methods for this interface.

- [IMStreamPage::GetSize](#)
The IMStreamPage::GetSize method returns the size of the stream.
- [IMStreamPage::GetData](#)
The IMStreamPage::GetData method returns the data in the stream page.
- [IMStreamPage::DeletePage](#)
The IMStreamPage::DeletePage method deletes the stream page object associated with the stream.

IMStreamPage::GetSize

The IMStreamPage::GetSize method returns the size of the stream.

Quick info

```
HRESULT GetSize(long *pVal)
```

Parameters

pVal
[out, retval] - stream size

Return values

The following value is returned when this method is called:

S_OK
Success status

IMStreamPage::GetData

The IMStreamPage::GetData method returns the data in the stream page.

Quick info

```
HRESULT GetData(VARIANT_BOOL bIntrep, VARIANT* pvData);
```

Parameters

bIntrep
[in]
pvData
[out] - return data in bytes

Return values

The following value is returned when this method is called:

S_OK
Success status

IMStreamPage::DeletePage

The **IMStreamPage::DeletePage** method deletes the stream page object associated with the stream.

Quick info

HRESULT DeletePage ()

Return values

The following value is returned when this method is called:

S_OK
Success status

COM API object hierarchy

The COM API objects exist in a defined hierarchy.

A map object can contain one or more card objects.

The object relationship is described in the following list:

- Each card object always has an adapter object associated with it.
- A map object has a number of card objects. The number is dependent on the number of input and output cards in a map.

Java API overview

The Java API interfaces enable you to create, edit, and run maps and schemas from a Java™ program.

IBM Transformation Extender provides a design-time Java API and a runtime Java API. The design-time Java API is different than the runtime Java API in that you can use the design-time Java API to create maps and type trees on Windows operating systems only.

For more information about the design-time Java API client interface, see the Javadocs by opening the overview-tree.html file located in *install_dir\docs\dtxds*.

For more information about the runtime Java API client interface, see the Javadocs by opening the overview-summary.html file located in *install_dir\docs\dtxpi*.

Java Tools overview

The Java™ Tools provide objects and JNI utilities that you can use to get information about your map instances.

For more information about the Java Tools client interfaces, see the Javadocs by opening the package-summary.html file located in *install_dir\docs\dtxpi\java\com\ibm\websphere\dtx\dtxpi\tools*.

Python API

Use the Python API to create, edit, and run maps and schemas from a Python program. For details, see the documentation installed in *install_dir\python\api\hip\docs\html\index.html*. Examples of the Python API are installed in *install_dir\python\api\examples*. Documentation for the Python API examples is available in *install_dir\python\api\docs\html\examples.html*.

IBM Transformation Extender REST API

IBM® Transformation Extender provides a REST API to run maps on a web server. An application can use virtually any programming language or technology to call the API, enabling transformation to be offered as a service from any web server or cloud platform.

- [Introduction](#)

- [Swagger documentation](#)

The APIs exposed by the ITX REST API are documented using Swagger.

- [Running a map directly](#)

- [REST API output](#)

When the map run completes, REST API output is text, only if a single card (and no other items) is returned. If two or more cards or any of the other optional items are added to a single (or multiple) card call, the output is a JSON data structure, with the actual card output being encoded with BASE64. Response is similar to the following example:

- [Cataloging a map](#)

- [Tutorial](#)

- [Important: Delete the artifacts from asynchronous map runs](#)
 - [Map caching](#)
 - [Map execution logs](#)
-

Introduction

An application can run an IBM® Transformation Extender map on a web server either synchronously or asynchronously. Use *synchronous* processing to run a map that must complete before it returns a response to the calling application. When you run a map *asynchronously*, the API returns URLs that an application can use to check the status of a long-running map and to retrieve output data and audit and trace logs when the map completes.

Running a map asynchronously requires a Redis server in addition to the web server. See the product [release notes](#) for details.

The servlet that implements the ITX REST APIs can run as a separate process from the web server process, in *fenced* run mode. Fenced mode ensures that an ITX failure does not affect the web server. Fenced mode is required to run a map asynchronously. In *unfenced* run mode, the servlet runs within the web server. Run maps in unfenced mode only for demos or during testing; do not use unfenced mode in production.

The HTTP request that invokes a map can pass data to one or more of the map's input cards. The HTTP response can return one or more map outputs. The API call can specify an adapter override that causes the map to receive input from an adapter or write output data to an adapter.

In addition, ITX provides a REST API that you can use to catalog maps and generate Swagger documentation for them. An application can run a cataloged map without specifying the URL of the map location or the cards to override, simplifying the web service call. This approach enables maps to be exposed as REST API endpoints without the need for the caller of the API to be aware of any ITX specifics.

The product [release notes](#) describe the prerequisites and required configuration to use the REST API and the tutorial.

Swagger documentation

The APIs exposed by the ITX REST API are documented using Swagger.

To access the Swagger documentation, enter the following URL in a browser. If the tx-rest.war file is installed on a server other than the local host, adjust the host name.

`http://localhost:8080/tx-rest/api-docs?url=/tx-rest/swagger.json`

You can invoke the APIs directly from the Swagger documentation by clicking Try it out under each API definition.

Running a map directly

Use curl to run a map from the command line.

- The HTTP PUT request runs the map synchronously.
- The HTTP POST request runs the map asynchronously.

Both requests use the same command parameters.

- [Synchronous direct calls](#)
 - [Asynchronous direct calls and map status queries](#)
-

Synchronous direct calls

The example below synchronously runs a map on a web server that's installed on the local host. The map has one input card and one output card. The command overrides the input card with the text string `test data` and requests the output data and audit log to be returned in the response:

```
> curl -X PUT -d "test data" "http://localhost:8080/tx-rest/v1/itx/maps/direct/OneInOneOut?input=1&output=1&return=audit"
```

<code>PUT</code>	Runs a map synchronously.
<code>-d "test data"</code>	Specifies the data to override a single card. To override multiple input cards, send the data for each card as multi-part form data, with the card number as the part name. For example: <code>"1=Card1 data" "2=Card2 data"</code>
<code>http://localhost:8080/tx-rest/v1/itx/maps/direct/</code>	The hostname and port of the web server where the tx-rest.war file is deployed, and the base URL for invoking maps directly.
<code>OneInOneOut</code>	The path to the compiled map file, relative to the <code>map.dirs=</code> path specified in the in the <code>install_dir/tx-rest.properties</code> file. The map file type (.mmc) is not required.

<code>input=1&output=1</code>	<p>The map input and output cards to override with data. The format of the parameters depends on whether a single card is overridden or multiple cards are overridden.</p> <p>Transfer data to or from a single card or a single audit or trace file as binary data. To override a single card, specify the card number as <code>input=cardnum</code> or <code>output=cardnum</code>.</p> <p>Transfer data to or from multiple cards or transfer multiple returns (such as a single output card with a trace file) as multi-part form data.</p> <ul style="list-style-type: none"> To override multiple input cards, send the data for each card as multi-part form data, with the card number as the part name. For example: <pre>"1=Card1 data" "2=Card2 data"</pre> <ul style="list-style-type: none"> To overwrite multiple output cards, specify: <pre>output=cardnum,cardnum</pre>
<code>return=[audit] [trace] [status] [auditonerror]</code> <code>[traceonerror] [statusonerror]</code>	<p>Specifies what to return from the map. Separate multiple values with commas, for example:</p> <pre>return=audit,trace</pre> <p>If you specify <code>auditonerror</code>, <code>traceonerror</code>, or <code>statusonerror</code>, the map returns audit, trace, or status only if the map does not complete successfully.</p>

Asynchronous direct calls and map status queries

The following command runs a map asynchronously and requests all outputs, the audit log, and the trace log to be returned from the map. To run a map asynchronously, you must set `rest.mode` in the `install_dir/config.yaml` configuration file.

```
> curl -X POST -F "1=Card1 data" -F "2=Card2 data" http://localhost:8080/tx-rest/v1/itx/maps/direct/TwoInTwoOut?
output=1,2&return=audit,trace
```

<code>POST</code>	Runs the map asynchronously.
<code>-F "1=Card1 data" -F "2=Card2 data"</code>	Overrides input card 1 and input card 2 with multi-part form data.
<code>http://localhost:8080/tx-rest/v1/itx/maps/direct</code>	The hostname and port of the web server where the tx-rest.war file is deployed, and the base URL for invoking maps directly.
<code>TwoInTwoOut</code>	The path to the compiled map file, relative to the <code>rest.persistence.maps</code> path specified in the in the <code>install_dir/config.yaml</code> file. The map file type (.mmc) is not required.
<code>output=1,2</code>	Overrides output cards 1 and 2 and returns a link to the data for both output cards in the response.
<code>return=audit,trace</code>	Returns a link to the audit log and trace log in the response when the map completes.

Running a map asynchronously immediately returns a response similar to the following. The response contains the URL that an application can use to query the status of a queued or long-running map.

```
{"id": "8b1f7e5e-5d16-480b-a972-03d66949cf7b", "href": "http://localhost:8080/tx-rest/v1/itx/maps/direct/8b1f7e5e-5d16-480b-a972-03d66949cf7b/status"}
```

Use a **GET** request and the URL from the response to query the map status:

```
> curl -X GET http://localhost:8080/tx-rest/v1/itx/maps/direct/8b1f7e5e-5d16-480b-a972-03d66949cf7b/status
```

The response from the **GET** request indicates whether the map execution is in progress or queued, the start time, and the number of milliseconds that the map has been queued or running. Call **GET** repeatedly until the map completes.

When the map completes, the status response is similar to the following example:

```
{
  "outputs": [
    {
      "href": "http://localhost:8080/tx-rest/v1/itx/maps/direct/8b1f7e5e-5d16-480b-a972-03d66949cf7b/outputs/1",
      "mime_type": "application/octet-stream",
      "card_number": 1
    },
    {
      "href": "http://localhost:8080/tx-rest/v1/itx/maps/direct/8b1f7e5e-5d16-480b-a972-03d66949cf7b/outputs/2",
      "mime_type": "application/octet-stream",
      "card_number": 2
    }
  ],
  "status": 0,
  "audit_href": "http://localhost:8080/tx-rest/v1/itx/maps/direct/8b1f7e5e-5d16-480b-a972-03d66949cf7b/audit",
  "trace_href": "http://localhost:8080/tx-rest/v1/itx/maps/direct/8b1f7e5e-5d16-480b-a972-03d66949cf7b/trace",
  "status_message": "Map completed successfully",
  "elapsed_time": 3,
  "start_timestamp": "2018-03-26T19:28:21.085+0000"
}
```

Use **GET** requests and the returned URLs to retrieve the output data, audit log, and trace log.

REST API output

When the map run completes, REST API output is text, only if a single card (and no other items) is returned. If two or more cards or any of the other optional items are added to a single (or multiple) card call, the output is a JSON data structure, with the actual card output being encoded with BASE64. Response is similar to the following example:

```
{
  "type": {
    "type": "multipart",
    "subtype": "related",
    "parameters": {},
    "wildcardType": false,
    "wildcardSubtype": false
  },
  "allAttachments": [
    {
      "headers": {
        "Content-ID": [
          "1"
        ],
        "Content-Type": [
          "application/octet-stream"
        ]
      },
      "object": null,
      "contentDisposition": null,
      "contentId": "1",
      "contentType": {
        "type": "application",
        "subtype": "octet-stream",
        "parameters": {},
        "wildcardType": false,
        "wildcardSubtype": false
      },
      "dataHandler": null
    },
    {
      "headers": {
        "Content-ID": [
          "2"
        ],
        "Content-Type": [
          "application/octet-stream"
        ]
      },
      "object": null,
      "contentDisposition": null,
      "contentId": "2",
      "contentType": {
        "type": "application",
        "subtype": "octet-stream",
        "parameters": {},
        "wildcardType": false,
        "wildcardSubtype": false
      },
      "dataHandler": null
    }
  ],
  "childAttachments": [
    {
      "headers": {
        "Content-ID": [
          "2"
        ],
        "Content-Type": [
          "application/octet-stream"
        ]
      },
      "object": null,
      "contentDisposition": null,
      "contentId": "2",
      "contentType": {
        "type": "application",
        "subtype": "octet-stream",
        "parameters": {},
        "wildcardType": false,
        "wildcardSubtype": false
      },
      "dataHandler": null
    }
  ],
  "rootAttachment": {
    "headers": {
      "Content-ID": [
        "1"
      ],
      "Content-Type": [
        "application/octet-stream"
      ]
    },
    "object": null,
    "contentDisposition": null,
    "contentId": "1",
    "contentType": {
      "type": "application",
      "subtype": "octet-stream",
      "parameters": {}
    }
  }
}
```

```
        "wildcardType": false,
        "wildcardSubtype": false
    },
    "dataHandler": null
}
```

Cataloging a map

Cataloging a map simplifies calls to the map and generates Swagger documentation that describes the APIs. The catalog provides a way to expose maps as services and to hide the implementation details of the services.

A map is defined in the catalog as a JSON document, similar to the `lin1out` map example that's included in the tutorial:

```
{
  "name": "lin1out",
  "summary": "Single input & single output",
  "description": "Uppercases the output from a single input",
  "tags": [ "Test" ],
  "inputs": [
    {
      "card_number": 1,
      "mime_type": "text/plain",
      "description": "Plain text"
    }
  ],
  "outputs": [
    {
      "card_number": 1,
      "mime_type": "text/plain",
      "description": "Uppercased input"
    }
  ],
  "url_path": "test/lin1out",
  "map_path": "OneInOneOut"
}
```

The Swagger documentation for the ITX REST API defines the structure of the JSON document and explains the purpose of each element.

You register a map in the catalog by posting the JSON document that defines it to the catalog directory specified on `rest.persistence.catalog` in the `install_dir/config.yaml` file. For example:

```
> curl -X POST "http://localhost:8080/tx-rest/v1/itx/catalog" -H "Content-Type: application/json" -d
@..\descriptors\lin1out.json
```

Use Swagger to display map details and to test-run maps. To see the Swagger documentation for the cataloged maps, enter the following URL in a browser:

```
http://localhost:8080/tx-rest/api-docs?url=/tx-rest/v1/itx/docs
```

If the tx-rest.war file is installed on a server other than the local host, adjust the host name.

You can run a cataloged map without specifying which cards to override or the name and path of the calling map. For example, the following command synchronously runs the cataloged version of the OneInOneOut map. Compare it to the direct call that runs the same map in [Synchronous direct calls](#):

```
> curl -X PUT -d "test data" "http://localhost:8080/tx-rest/v1/itx/maps/catalog/test/lin1out"
```

Tutorial

Transformation Extender installs a REST API tutorial in the `install_dir\examples\restapi` directory. The tutorial provides example maps, scripts, and README instructions that demonstrate how to catalog maps, access them from the Swagger interface, and call them using curl commands.

Before you use the tutorial, see the [release notes](#) for the REST API configuration requirements.

Important: Delete the artifacts from asynchronous map runs

The status, audit log, and trace log from an asynchronous map run are stored on the Redis server until you explicitly delete them. You must explicitly delete these artifacts when you run a map asynchronously, whether you run the map through a direct call, through the catalog, or from the Swagger interface.

To delete the artifacts of a map run, use a **DELETE** request and the URL of the map-run resource returned from the initial **POST** request. (Omit `/status` from the URL.) For example:

```
> curl -X DELETE "http://localhost:8080/tx-rest/v1/itx/maps/direct/8b1f7e5e-5d16-480b-a972-03d66949cf7b"
```

To verify that the map artifacts are deleted, use a **GET** request to request the map's status. For example:

```
> curl -X GET http://localhost:8080/tx-rest/v1/itx/maps/direct/8b1f7e5e-5d16-480b-a972-03d66949cf7b/status
```

After the artifacts of the map run are deleted, the **GET** request returns a response similar to the following:

```
{"code":404,"message":"No map run is in progress with id 8b1f7e5e-5d16-480b-a972-03d66949cf7b."}
```

Map caching

The ITX REST API caches maps in memory to avoid reloading them each time the map is executed.

To prevent maps that are used infrequently from consuming server resources, the REST API unloads a map that has not been accessed for a specified amount of time. The `rest.map.unloadTime` configuration parameter in the `install_dir/config.yaml` file specifies the amount of time (in minutes) that can elapse before an unaccessed map is unloaded.

You do not need to restart the web server when you change a map. The REST API detects when a map is updated, unloads it from memory, and reloads the map. Subsequent map runs use the updated map.

Map execution logs

Configure and enable map execution logging in the `install_dir/config.yaml` file.

Remote Method Invocation (RMI) API overview

Remote Method Invocation (RMI) is used when writing object-oriented programming so objects on different machines are able to communicate in a distributed network environment.

RMI is the Java™ version of a remote procedure call (RPC), however, it also includes the capability to pass one or more objects along with the request. The RMI API provides functionality for loading and running maps, overriding adapters in cards, setting and retrieving object properties, as well as feeding data to maps and obtaining data from maps. The remote feature in the RMI API makes maps available to be run outside of the client process space.

For more information about the RMI API client interfaces, see the Javadocs by opening the `package-summary.html` file located in `install_dir/docs/dtxpl/com/ibm/websphere/dtx/dtxpl/rmi`.

- [RMI server overview](#)
The RMI API is composed of two parts: client and server.
- [RMI server configurations](#)
You must configure the RMI server to run your maps.
- [RMI API configuration overview](#)
A fatal exception within the transformation logic that runs the map could cause the calling process to terminate. To mitigate against this, there are three configuration options available for the RMI API.

RMI server overview

The RMI API is composed of two parts: client and server.

The client interface component consists of Java™ classes used to load maps and override map settings. The server component, or RMI server, runs the maps.

- [RMI server implementation](#)
The RMI server runs maps in its own process and operates in one of three modes: single process, multi-process, and in-process.
- [RMI server error handling](#)
All error conditions generated by the IBM Transformation Extender Programming Interface are wrapped and re-thrown as RMI API exceptions.
- [RMI server process modes](#)
The RMI server can run in either a single or multi-process mode.

RMI server implementation

The RMI server runs maps in its own process and operates in one of three modes: single process, multi-process, and in-process.

The mode is specified when the server is started.

Related tasks

- [Starting the RMI server](#)

Related reference

- [RMI server mode property](#)

RMI server error handling

All error conditions generated by the IBM Transformation Extender Programming Interface are wrapped and re-thrown as RMI API exceptions.

These include various map, adapter, and internal failures. Errors generated by the RMI server are also thrown as RMI API exceptions as well as all unexpected error conditions. Low-level RMI and network-related exceptions are passed unchanged to the client. The MRmiApiException class has a list of RMI server-specific error conditions.

RMI server process modes

The RMI server can run in either a single or multi-process mode.

The different process modes control the way in which the RMI server runs the maps.

- [RMI server single process mode](#)
The RMI server can run in a single process mode.
 - [RMI server multi-process mode](#)
The RMI server can run in a multi-process mode.
 - [RMI server in-process mode](#)
The RMI server can run in an in-process mode. The in-process implementation is contained within a custom application process and is used when maps must be run within the client process.
-

RMI server single process mode

The RMI server can run in a single process mode.

Use the single process implementation to run maps in a single server process.

RMI server multi-process mode

The RMI server can run in a multi-process mode.

Use the multi-process implementation to run maps in multiple server processes. Each map runs in its own child process, referred to as the Map Process, of the main server process, referred to as the Control Process.

RMI server in-process mode

The RMI server can run in an in-process mode. The in-process implementation is contained within a custom application process and is used when maps must be run within the client process.

The in-process implementation is contained within a custom application process and is used when maps must be run within the client process. Though the in-process mode appears to an RMI API user as a client-server process, it is actually a single process. To run the RMI server in an in-process mode, the server must be installed on the same machine as the RMI API client component.

RMI server configurations

You must configure the RMI server to run your maps.

The configuration activities include setting the RMI server properties, starting the RMI server, and installing the RMI API client.

- [RMI server properties overview](#)
The RMI server has a set of properties that are defined in the rmiserver.properties file.
 - [Starting the RMI server](#)
You must start the RMI server from the command line before running the RMI API.
 - [RMI API client installation overview](#)
There are two configuration scenarios regarding the RMI API client component: installing the client component on the same machine running the RMI server, or on a separate machine.
-

RMI server properties overview

The RMI server has a set of properties that are defined in the rmiserver.properties file.

The rmiserver.properties file is read when the RMI server is started. However, a different property file can be specified at startup. The RMI server properties can be read from or persisted to the rmiserver.properties file using the **loadServerProperties()** and **storeServerProperties()** methods in the RMI API IMManager class. To read from or store these properties to a different file, use the **load()** and **store()** methods in java.util.Properties, and the **setServerProperties()** and **getServerProperties()** methods in the RMI API IMManager class.

The following table lists server properties and the mode they support.

- [RMI server properties](#)

The following table lists server properties and the RMI server process modes that they support.

RMI server properties

The following table lists server properties and the RMI server process modes that they support.

Property	Mode
RMI server mode property	single/multi
RMI server port property	single/multi
RMI server log mode full property	single/multi
RMI server pool.mode.managed property	multi-process
RMI server pool.max.process.count property	multi-process
RMI server pool.max.keep.idle.count property	multi-process
RMI server pool.max.idle.time property	multi-process
RMI server pool.acquire.process.timeout property	multi-process
RMI server pool.map.auto.unload.timeout property	multi-process

- [RMI server mode property](#)

The RMI server mode is specified through the server.mode.multi.process property defined in the rmiserver.properties file.

- [RMI server port property](#)

An instance of the RMI server is identified by specifying the TCP/IP server port using the server.port property defined in the rmiserver.properties file.

- [RMI server log.mode.full property](#)

The log.mode.full RMI server property specifies whether log information includes both server activity and errors, or error information only.

- [RMI server pool.mode.managed property](#)

The pool.mode.managed RMI server property turns pool management on or off.

- [RMI server pool.max.process.count property](#)

The pool.max.process.count specifies the maximum number of processes that the pool can create at any time.

- [RMI server pool.max.keep.idle.count property](#)

The pool.max.keep.idle.count specifies the maximum number of idle processes that are kept in the pool.

- [RMI server pool.max.idle.time property](#)

The pool.max.idle.time specifies the maximum amount of time, in seconds, that the pool will idle before disposing of the idle processes.

- [RMI server pool.acquire.process.timeout property](#)

The pool.acquire.process.timeout specifies the maximum amount of time the pool can block a request in order to acquire a process before it returns a failure.

- [RMI server pool.map.auto.unload.timeout property](#)

The pool.map.auto.unload.timeout specifies the maximum amount of time a map can stay loaded.

RMI server mode property

The RMI server mode is specified through the server.mode.multi.process property defined in the rmiserver.properties file.

The following table lists the valid values for the RMI server mode and their description.

Values	Description
--------	-------------

true	Makes the RMI server available to run in multiprocess mode.
------	---

false	Makes the RMI server available to run in single process mode.
-------	---

When multiprocess mode is enabled (server.mode.multi.process=true), map execution takes place in a separate JVM process. You set the startup options for this JVM process with the multi.process.jvm.options RMI option.

Related reference

- [RMI server implementation](#)

RMI server port property

An instance of the RMI server is identified by specifying the TCP/IP server port using the server.port property defined in the rmiserver.properties file.

The RMI server uses four consecutive port numbers starting with the specified port. For example, if you specify 2500 as the port number, the RMI server will actually use ports 2500-2503.

The following table lists the valid values for the RMI server port and their description.

Values	Description
--------	-------------

integer	Specifies TCP/IP server port.
---------	-------------------------------

RMI server log.mode.full property

The log.mode.full RMI server property specifies whether log information includes both server activity and errors, or error information only.

The following table lists the valid values for the log.mode.full RMI server property and their description.

Values	Description
true	Makes available both error and activity logging.
false	Makes available error logging only.

RMI server pool.mode.managed property

The pool.mode.managed RMI server property turns pool management on or off.

When it is turned off, there is no limit to the number of processes, and time idle processes are kept.

If this property is set to false, settings for all of the properties that control the process pool behavior are ignored.

The following table lists the valid values for the pool.mode.managed RMI server property and their description.

Values	Description
true	Turns pool management on.
false	Turns pool management off.

Related reference

- [Process pool management](#)

RMI server pool.max.process.count property

The pool.max.process.count specifies the maximum number of processes that the pool can create at any time.

When the limit is reached, new requests are queued and the client is blocked until either a process becomes available, or the request times out. If the request times out, the pool will return a failure.

The following table lists the valid values for the pool.max.process.count RMI server property and their description.

Values	Description
≥0	Maximum number of processes that the pool can create at any time. Zero or a negative value disables this feature.

Related reference

- [RMI server pool.acquire.process.timeout property](#)

RMI server pool.max.keep.idle.count property

The pool.max.keep.idle.count specifies the maximum number of idle processes that are kept in the pool.

All released processes above this number are destroyed.

The following table lists the valid values for the pool.max.keep.idle.count RMI server property and their description.

Values	Description
≥0	Maximum number of idle processes that are kept in the pool. A negative value disables this feature.

RMI server pool.max.idle.time property

The pool.max.idle.time specifies the maximum amount of time, in seconds, that the pool will idle before disposing of the idle processes.

The pool is idling if there are no active processes.

The following table lists the valid values for the pool.max.idle.time RMI server property and their description.

Values	Description
≥0	Maximum amount of time, in seconds, that the pool will idle before disposing of the idle processes. A negative value disables this feature.

RMI server pool.acquire.process.timeout property

The pool.acquire.process.timeout specifies the maximum amount of time the pool can block a request in order to acquire a process before it returns a failure.

This is used when a new request arrives, but the pool has already reached the maximum process count.

The following table lists the valid values for the pool.acquire.process.timeout RMI server property and their description.

Values	Description
≥0	Maximum amount of time the pool can block a request in order to acquire a process before it returns a failure. A negative value disables this feature.

Related reference

- [RMI server pool.max.process.count property](#)

RMI server pool.map.auto.unload.timeout property

The pool.map.auto.unload.timeout specifies the maximum amount of time a map can stay loaded.

When the time expires, the map is automatically unloaded and the associated process is released.

The following table lists the valid values for the pool.map.auto.unload.timeout RMI server property and their description.

Values	Description
≥0	Maximum amount of time a map can stay loaded. A negative value disables this feature.

Starting the RMI server

You must start the RMI server from the command line before running the RMI API.

The RMI server is started from the command line on both the Windows and UNIX operating systems.

To start the RMI server

1. Edit the *install_dir\startRMIServer.bat* file by setting the following variables:
 - JAVAHOME: Java 2 SDK installation directory (UNIX only)
 - MERCHOME: *tx_install_dir* (Windows only)
 - CONFIG_FILE: RMI server property filename and pathFor UNIX operating systems, modify the *install_dir/bin/startRMIServer.sh* file.
2. Save the file.
3. For UNIX operating systems only, enter the following command from a command prompt:
install_dir/setup
4. From a command prompt, enter the following command to start the RMI server:
 - Windows: *install_dir\startRMIServer.bat*
 - UNIX: *install_dir/startRMIServer.sh*

Related reference

- [RMI server implementation](#)

RMI API client installation overview

There are two configuration scenarios regarding the RMI API client component: installing the client component on the same machine running the RMI server, or on a separate machine.

The RMI API client component and the RMI server are included with the DK installation.

- [Installing the DK on the same machine](#)

If the DK is installed on the same machine running the RMI server, no additional installation steps are required because the client component is already installed.

- [Installing the DK on a separate machine](#)

If the DK is installed on a separate machine from the machine running the RMI server, you must provide additional configuration for the RMI API client component.

Installing the DK on the same machine

If the DK is installed on the same machine running the RMI server, no additional installation steps are required because the client component is already installed.

If the DK is installed on the same machine running the RMI server, no additional installation steps are required because the client component is already installed.

Installing the DK on a separate machine

If the DK is installed on a separate machine from the machine running the RMI server, you must provide additional configuration for the RMI API client component.

If the DK is installed on a separate machine from the one running the RMI server, the following additional steps are required to configure the RMI API client component.

To install the RMI API client component on a separate machine

Copy the following file to the machine hosting the RMI API client:

- Windows: `install_dir\dtxpi.jar`
- UNIX: `install_dir/libs/dtxpi.jar`

RMI API configuration overview

A fatal exception within the transformation logic that runs the map could cause the calling process to terminate. To mitigate against this, there are three configuration options available for the RMI API.

The configuration options are:

- [RMI API in-process configuration](#)
- [RMI API single process configuration](#)
- [RMI API multi-process configuration](#)

- [RMI API in-process configuration](#)

To prevent a fatal exception within the transformation logic that runs the map and causes the calling process to terminate, you can use the in-process configuration option.

- [RMI API single process configuration](#)

To prevent a fatal exception within the transformation logic that runs the map and causes the calling process to terminate, you can use the single process configuration option.

- [RMI API multi-process configuration](#)

To prevent a fatal exception within the transformation logic that runs the map and causes the calling process to terminate, you can use the multi-process configuration option.

RMI API in-process configuration

To prevent a fatal exception within the transformation logic that runs the map and causes the calling process to terminate, you can use the in-process configuration option.

The maps are run within the process of the calling program (in the same JVM). In this mode, there are no RMI calls - a call to the RMI API calls directly to the transformation logic that runs the map. This option is best-suited to a situation where you need optimal performance and the loss of the calling program is not catastrophic.

This option is not recommended for clients running on an application server, since a fatal exception could cause the server process to terminate.

Related reference

- [Comparison of the RMI API configuration options](#)

RMI API single process configuration

To prevent a fatal exception within the transformation logic that runs the map and causes the calling process to terminate, you can use the single process configuration option.

The maps are run within the RMI server process, with each map running in a separate thread. This protects the calling program from fatal map exceptions because the maps are running remotely from the calling program. However, if one map fails, the RMI server stops and all concurrent maps are terminated.

This option is well suited to programs running on an application server.

Related reference

- [Comparison of the RMI API configuration options](#)
-

RMI API multi-process configuration

To prevent a fatal exception within the transformation logic that runs the map and causes the calling process to terminate, you can use the multi-process configuration option.

The maps are run in processes created from the RMI Server process.

This option offers the best protection, since a fatal exception in one map will not adversely affect other maps.

The downside of this approach is that adapter connections are not shared across map processes. This means that a connection that is started in one process for a particular map can be reused only by maps that run in that same process. This might be an issue if many different maps are run that connect to a wide array of resources, such as different databases.

Related reference

- [Comparison of the RMI API configuration options](#)
-

Web Services

Web Services includes the SOAP Adapter and WSDL (Web Services Definition Language) Importer. You can use these tools to provide or consume Web services.

SOAP (WSDL) Adapter

SOAP defines the XML-based message format that applications use to communicate and inter-operate with each other over the Web.

The heterogeneous environment of the Web demands that applications support a common data encoding protocol and message format.

SOAP is a standard for encoding messages in XML that invoke functions in other applications running on any hardware platform regardless of different operating systems or programming languages. It is analogous to Remote Procedure Calls (RPC) used in many technologies such as Distributed Component Object Model (DCOM), but eliminates some of the complexities of using these interfaces.

The SOAP (WSDL) Adapter supports the SOAP 1.1 protocol as defined by <http://www.w3.org/TR/SOAP/>.

Use the SOAP (WSDL) Adapter to help to create SOAP request messages and to interpret response messages. The SOAP (WSDL) Adapter does not handle the delivery of messages and should be used with a transport adapter, such as HTTP, to deliver the request and receive the response message.

When you send a SOAP-based HTTP request, the default encoding of the content of the HTTP request is ISO-8859-1. The SOAP adapter requires a UTF-8 encoded response from the HTTP server, but the request can be in any Western or UTF-8 encoded format. See the description of the **-TYPE** command in the HTTP adapter documentation for details about how to specify the encoding.

The adapter has the following capabilities:

- Adds SOAP envelopes if they are not provided
- Parses the response message to extract scalar data elements
- Removes envelope and simplifies the form of SOAP responses to aid mapping of the response data
- Understands fault codes and interprets them as map failures

The SOAP (WSDL) Adapter does not provide a listener. The transport adapter provides this functionality.

- [System requirements](#)
 - [Command alias](#)
 - [SOAP Adapter commands](#)
 - [Syntax summary](#)
 - [Using the adapter](#)
 - [Return codes and error messages](#)
-

System requirements

The minimum system requirements and operating system requirements for the SOAP (WSDL) Adapter are detailed in the release notes.

Command alias

Adapter commands can be specified by using a command string on the command line or by creating a command file that contains adapter commands. The execution command syntax is:

```
-IA[alias] card_num  
-OA[alias] card_num
```

where **-IA** is the Input Source Override execution command and **-OA** is the Output Target Override execution command, alias is the adapter alias, and **card_num** is the number of the input or output card. The following table shows the adapter alias and its execution command.

Adapter	Alias	As Input	As Output
SOAP	SOAP	-IASOAPcard_num	-OASOAPcard_num

SOAP Adapter commands

This documentation describes the functions and use of the SOAP (WSDL) Adapter commands and their options.

- [List of commands](#)

List of commands

The following table lists valid commands for the SOAP (WSDL) Adapter, the command syntax, and whether the command is supported (✓) for use with data sources, targets, or any other combination.

Commands can be specified on the adapter's command line or through adapter properties. Note that the commands are not case sensitive but that each value is case sensitive.

Command	Syntax	Source	Target
Decode (-DECODE)	-DECODE	✓	
Encode (-ENCODE)	-ENCODE		✓
Header (-HDR)	-HDR	✓	
Raw (-RAW)	-RAW	✓	
Return (-RETURN)	-RETURN element	✓	
SOAPAction (-SA)	-SA soap_action		✓
Trace (-T)	-T[E][NX][+] [filename]	✓	✓
Transport (-TRANSPORT)	-TRANSPORT 'adapter_name (adapter_command_line)'	✓	✓

- [Decode \(-DECODE\)](#)
- [Encode \(-ENCODE\)](#)
- [Header \(-HDR\)](#)
- [Raw \(-RAW\)](#)
- [Return \(-RETURN\)](#)
- [SOAPAction \(-SA\)](#)
- [Trace \(-T\)](#)
- [Transport \(-TRANSPORT\)](#)

Decode (-DECODE)

Use the Decode adapter command (-DECODE) to instruct the adapter to decode the SOAP message, which involves stripping the envelope and SOAP envelope header, and handling SOAP faults.

-DECODE

For example, to decode a message received over HTTP:

```
GET ("SOAP", "-DECODE -TRANSPORT 'HTTP (-URL http://myurl/)'")
```

The **-DECODE** option is optional since it is implied by the invocation of the SOAP (WSDL) Adapter in an input card or GET function.

For more information on encoding and decoding data, see *Encoding and Decoding Data* in the Resource Adapters documentation.

Encode (-ENCODE)

Use the Encode adapter command (-ENCODE) to instruct the adapter to encode the SOAP message, which involves adding a SOAP envelope if none is provided.

-ENCODE

For example, to encode a SOAP message before transporting over HTTP:

```
PUT ("SOAP", "-ENCODE -TRANSPORT 'HTTP (-URL http://myurl/)'  
SoapRequest")
```

The **-ENCODE** option is optional since it is implied by the invocation of the SOAP (WSDL) Adapter in an output card or PUT function.

For more information on encoding and decoding data, see *Encoding and Decoding Data* in the Resource Adapters documentation.

Header (-HDR)

Use the Header adapter command (-HDR) to specify that the SOAP envelope header be returned. This is an optional command. The default is to not return the SOAP header.

-HDR

See [Source Options Behavior](#) for more information on the behavior of the SOAP (WSDL) Adapter and its combinations of available options.

Raw (-RAW)

Use the Raw adapter command (-RAW) to specify that the response message be returned unmodified, except the SOAP envelope header is removed.

-RAW

If the -RAW command is specified, the only processing of the response data is:

The SOAP envelope header is stripped out. To keep the header specify -HDR.

Fault codes are interpreted

For example, using the -RAW command results in the data being returned as follows (intact, but without SOAP envelope header):

```
<?xml version="1.0" encoding="UTF-8" standalone="no" ?>
- <SOAP-ENV:Envelope SOAP-
  ENV:encodingStyle=http://schemas.xmlsoap.org/soap/encoding/
  xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
- <SOAP-ENV:Body>
  - <SOAPSSDK1:AddResponse
    xmlns:SOAPSSDK1="http://tempuri.org/message/">
    <Result>290</Result>
  </SOAPSSDK1:AddResponse>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

If the -RAW command is not specified, then the following is returned:

- The body of the request without the enclosing **<Body>...</Body>** tags
- Any namespace prefixes are removed
- Any attributes are removed

For example, the message shown above is returned as follows:

```
- <AddResponse>
  <Result>290</Result>
</AddResponse>
```

See [Source Options Behavior](#) for more information on the behavior of the SOAP (WSDL) Adapter and its combinations of available options.

Return (-RETURN)

Use the Return adapter command (-RETURN) to return a single scalar value from the response message. In many cases, a map only has interest in a single value. This option simplifies mapping in these types of cases. This command can only be used if the value of the element is a scalar value.

This command is incompatible with the -HDR command.

This is an optional command. The default is a null reply.

-RETURN element

Option

Description

element

Specify the path and the element to be returned using XPath syntax.

XPath is a language for addressing parts of an XML document. For more information on the XPath language, go to www.w3.org/TR/xpath.

For example, to specify /cat/dog as the XML element to be returned:

```
-RETURN /cat/dog
```

See [Source Options Behavior](#) for more information on the behavior of the SOAP (WSDL) Adapter and its combinations of available options.

- [Return option and XPath](#)

Return option and XPath

The element to return is specified using an XPath subject to the following limitations:

- The element to return must be a scalar type. That is, it cannot be an array or contain another element.
- Only fully qualified paths are accepted, for example:
/Message/StockQuote/Price
- Attributes of elements can be specified using the standard XPath '@' notation, for example:
/Message/StockQuote/@currency

While the SOAP standard does not forbid the use of attributes, it recommends not using them.

No other XPath syntax is supported

The limitations in the previous examples are taken into consideration in the following sample message:

```
<Message>
  <StockQuote>
    <Symbol>XYZ</Symbol>
    <Price currency="dollar">20</Price>
  </StockQuote>
</Message>
```

The names of the elements in the path correspond to the name in the returned data. Therefore, if the `-RAW` command is used, the names correspond to the element names in the raw response message. In this case, the elements are often qualified by namespace prefixes.

Consider the following example return message:

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:SOAP-ENC="http://schemas.xmlsoap.org/soap/encoding/">
<SOAP-ENV:Body SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
<NS1:BabelFishResponse xmlns:NS1="urn:BorlandBabelIntf-IBorlandBabel">
<return xsi:type="xsd:string">Freeend</return>
</NS1:BabelFishResponse>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

If we want to return only the `<return>` element, the following command(s) should be used:

```
-RAW -RETURN /SOAP-ENV:Envelope/SOAP-ENV:Body/
NS1:BabelFishResponse/return
```

Or, if the `-RAW` command is not specified:

```
-RETURN /BabelFishResponse/return
```

SOAPAction (-SA)

When the transport is HTTP, many Web services require the SOAPAction field to be added to the HTTP header of a SOAP request. Use the SOAPAction adapter command (`-SA`) to indicate to the server the intent of the request.

Use one of the following values:

- `NULL` (`-SA` with no argument), means that there is no indication of the intent
- `Empty string` (`-SA ""`), means that the intent of the request is provided by the HTTP request URI or Uniform Resource Identifier (no value is specified between the single quotes)
- `URI` (`-SA 'http://www.me.com/'`), where the intent is provided by the URI

The presence and content of the SOAPAction header field can be used by servers to appropriately filter SOAP request messages.

This is an optional command. The default is no SOAPAction.

```
-SA soap_action
```

Option	Description
<code>soap_action</code>	Specify the intent of the request for the SOAP message.

When using the HTTP adapter as the transport, the HTTP adapter command `-HDR1+` must be used to instruct the HTTP adapter to add the specified SOAPAction to the HTTP header.

For example, to specify urn:BorlandBabelIntf-IBorlandBabel#BabelFish as the SOAPAction:

```
GET ("SOAP", "-SA 'urn:BorlandBabelIntf-  
IBorlandBabel#BabelFish' -TRANSPORT 'http(-HDR+ -HDIR+ -URL  
http://ww6.borland.com/webservices/BorlandBabel/  
BorlandBabel.exe/soap/IBorlandBabel -T)'", RequestData)
```

If the WSDL Importer is used to create the schemas for the SOAP request messages, the value that should be assigned to the SOAP action command can be found in the description of the corresponding operation.

See the Schema Importers documentation for more information on the WSDL Importer.

Trace (-T)

Use the Trace adapter command (-TRACE or -T) to produce a diagnostics file that contains detailed information about SOAP (WSDL) Adapter SOAP (WSDL) Adapter activity.

The default filename is **m4soap.mtr** and the file is created in the map directory unless otherwise specified.

```
-T [E] [NX] [+ ] [filename]
```

Option	Description
E	Produce a trace file containing only the adapter errors that occurred during map execution.
NX	Excludes information about XML data from the trace file.
+	Append trace information to the existing trace file.
filename	Creates a trace file with the specified name in the specified directory.

You can override the adapter command line trace options dynamically using the Management Console.

Transport (-TRANSPORT)

Use the Transport adapter command (-TRANSPORT) to specify which transport adapter to use and the command line for the transport adapter.

The syntax of the command must include the single quotes and parentheses as shown below.

```
-TRANSPORT 'adapter_name (adapter_command_line)'
```

Option	Description
adapter_name	Specify which transport adapter to use.
adapter_command_line	Specify the command line for the transport adapter. You must enclose this option within parentheses () .

For example, to specify the HTTP adapter as the transport adapter, and to send the SOAP request to the http://localhost/WebService/test URL, enter the following command:

```
-TRANSPORT 'http (-URL http://localhost/WebService/test -T)'
```

Syntax summary

This documentation discusses the SOAP syntax summary and how it is used.

- [Data sources](#)
- [Data targets](#)
- [Source options behavior](#)

Data sources

The following is the command syntax for the SOAP (WSDL) Adapter commands used for data sources:

```
-TRANSPORT 'adapter_name (adapter_command_line)'  
[-DECODE]  
[-RAW]  
[-HDR]  
[-RETURN element]  
[-T [E] [NX] [+ ] [filename]]
```

- [Example](#)

Example

For the GET Source setting in an input card, select SOAP. In the **GET > Source > Command** field, enter:

```
-T -RAW -TRANSPORT 'http (-URL http://www.myws.com/DoWS) '
```

This command specifies that the SOAP message:

- Received data from the HTTP URL `http://www.myws.com/DoWS` using the HTTP adapter (`-TRANSPORT`)
- Returned data in full (`-RAW`)
- Adapter trace is enabled (`-T`) and a diagnostics file is produced

Data targets

The following is the command syntax for the SOAP adapter commands used for data targets:

```
-TRANSPORT 'adapter_name (adapter_command_line)'
[-ENCODE]
[-SA soap_action]
[-T[E] [NX] [+| [filename]]]
```

- [Example](#)

Example

For the PUT Target setting in an output card, select SOAP. In the **PUT > Target > Command** field, enter:

```
-SA " -T -TRANSPORT 'http (-URL http://www.myws.com/DoWS) '
```

This command specifies that the SOAP message:

- Sent data over HTTP to the URL `http://www.myws.com/DoWS` using the HTTP adapter (`-TRANSPORT`)
- SOAPAction is provided by the HTTP request URL (`SA "`)
- Adapter trace is enabled (`-T`) and a diagnostics file is produced

Source options behavior

The following table defines the behavior of the SOAP (WSDL) Adapter based upon combinations of available options.

Commands	Behavior
No formatting commands	Only the contents of the body of the message are returned (this is the default). No attributes are returned and namespace prefixes are removed.
-RAW alone	If a header is in the message's envelope, it is removed, but the envelope and body are returned intact.
-HDR alone	Equivalent to -RAW and -HDR , since an envelope is required to ensure the well-formedness of the XML document. For more information on XML document standards go to www.w3.org/TR/1998/REC-xml-19980210 .
-RAW and -HDR	The SOAP request is returned in its entirety.
-RETURN XPath	Only the single scalar value is returned. See Return Option and XPath for more information.

Using the adapter

Use the SOAP (WSDL) Adapter to either invoke a Web service by sending a request message and processing the response, or in the implementation of a Web service.

Note that two adapters are used in communicating between the map and the Web service:

- The SOAP (WSDL) Adapter
- A transport adapter, such as an HTTP adapter

There are two primary usage scenarios for the SOAP (WSDL) Adapter:

- As a consumer of Web services

- As a provider of Web services
 - [Consumer scenario](#)
 - [Provider scenario](#)
 - [Example files](#)
-

Consumer scenario

A common scenario is the flow of a request from a map, through SOAP and HTTP adapters to the Web Service, and the flow of the response back to the map from the Web Service.

The adapter is designed to be called in a GET map function call. The third parameter of the GET is the request message. For example:

```
Response = GET("SOAP", "-HDR -TRANSPORT 'HTTP(-URL http://  
www.stuff.com -T)', PACKAGE(Request))
```

Adapter Action Sequence

The following describes the sequence of actions that occur in the consumer scenario.

- The request message is first passed to the SOAP (WSDL) Adapter.
 - The request message might or might not have a SOAP envelope. If the envelope is not provided, the adapter adds it.
 - The request message might or might not have a SOAP envelope header. The adapter does nothing with the header. If it is not provided in the request, then no header is sent.
 - The adapter specified in the -TRANSPORT command is invoked to issue the request and receive a reply. Typically, this is the HTTP adapter that will then send an HTTP request and return the HTTP response.
 - The SOAP (WSDL) Adapter is invoked a second time to process the reply.
 - The adapter looks in the body to see if there is a Fault tag. If there is, the SOAP (WSDL) Adapter returns an error message that is comprised of the faultstring and the faultcode.
 - The SOAP envelope is removed unless the -RAW command is specified.
 - If the adapter is called with the -HDR command, and there is a SOAP envelope header, the header data will be returned.
 - If the adapter is called with the -RETURN command, then all XML tags are removed and only the specified datum from the response message is returned. If the response message returns more than one datum, then the whole XML response message is returned with a warning code. This command is mutually exclusive with the -HDR command.
-

Provider scenario

The following descriptions explain using the SOAP (WSDL) Adapter with the HTTP adapter.

- [Response](#)
-

Response

The HTTP Adapter is invoked to send the response message.

Example files

The `install_dir\examples\web_services` directory contains sample files to be used with Web Services components, including the WSDL Importer, SOAP and HTTP adapters, to implement Web services strategies.

See `readme.txt` located in the example directory for more information about the examples.

Return codes and error messages

Return codes and messages are returned when the particular activity completes. Return codes and messages might also be recorded as specified in the audit logs, trace files and execution summary files.

- [Messages](#)
-

Messages

The following is a listing of all the codes and messages that can be returned as a result of using the SOAP (WSDL) Adapter for sources or targets.

Adapter return codes with positive numbers are warning codes that indicate a successful operation. Adapter return codes with negative numbers are error codes that indicate a failed operation.

Return Code

Message	
-2001	General internal adapter error
-2002	Invalid command line option
-2003	Missing argument in command line option
-2004	Missing command line option
-2005	Invalid argument in command line option
-2006	FC A fault code has been generated.
-2007	Xpath data not found
-2008	Xpath duplicate data found
-2009	Stream error
-2010	XML invalid

WSDL Importer

The Web Services Description Language (WSDL) Importer is a utility for generating type trees from message definitions contained within WSDL documents.

Tuning Importer behavior

The WSDL Importer has a Java™ native interface and uses the JVM options specified in the dtx.ini file to create a Java virtual machine (JVM). For information about configuring JVM options to tune the importer's behavior, see ["Configuring JVM options from the dtx.ini file"](#).

Import options

The WSDL import process can create a classic type tree or a native XML schema (.xsd) file from the WSDL document. You select the type of output in the WSDL Importer wizard.

- When the WSDL document structure requires support for derived types (such as through the xsi:type attribute), create a native XML schema.
- When the WSDL document structure does not require support for derived types, create a classic XML type tree. Classic type trees are smaller than native XML schemas.

Classic type tree processing

The importer reads a WSDL document and assembles lists of port types, operations, services, messages, and bindings. When the parsing is complete, the importer creates a map for each web service operation and adds it to the resulting XML map source file. During this process, the XML Schema Importer is used to create the types defined in the input WSDL document.

After the WSDL Importer imports the WSDL document, it creates a WSDL type tree (.mtt) file and an XML (*_map.xml) file. Use File > Import > Map from XML in the Design Studio to import the XML file to create a map. The import process creates a map (*WSDLname.map.mms*) that uses the new WSDL type tree in the output cards.

Native XML schema processing

When you select native XML schema validation in the wizard, the WSDL Importer imports the WSDL document and creates two XML schemas:

WSDLname.xsd
Contains the XSD and WSDL-based schema types.
WSDLname_soap.xsd
Contains the SOAP envelope, header, and body types.

Use the *WSDLname_soap.xsd* as the card's native type tree.

- [Support for SOAP arrays](#)
- [WSDL document elements](#)
- [WSDL document types](#)
- [Running the WSDL Importer](#)
Use the WSDL Importer to automatically generate type trees or XML schemas.
- [Special symbols](#)
- [Character restrictions](#)
- [WSDL example](#)

z/OS Configuration overview

The z/OS® Configuration documentation provides additional configuration information for the IBM® Transformation Extender for z/OS features that you have already installed. This information will help you customize IBM Transformation Extender for z/OS for your site.

The z/OS Configuration information applies to the following IBM Transformation Extender for z/OS features:

- IBM Transformation Extender - Common Components
- IBM Transformation Extender with Command Server
- IBM Transformation Extender with Launcher
- IBM Transformation Extender for Application Programming

The z/OS Configuration documentation also contains information that applies to specific products, as well as across products. They provide additional configuration instructions to assist you in using the installed features for:

- deploying the installed features to your runtime environment
 - setting environmental variables
 - running examples
 - other post-installation information
- [Related publications for z/OS](#)
This topic lists Transformation Extender for z/OS functions and related documentation.
- [SMP/E installation](#)
IBM Transformation Extender for z/OS features are installed through System Modification Program Extended (SMP/E).

Related reference

- [Related publications for z/OS](#)

Related publications for z/OS

This topic lists Transformation Extender for z/OS® functions and related documentation.

IBM Transformation Extender for IBM Business Process Manager Advanced Function	Documentation
To customize IBM Transformation Extender for z/OS for your site.	z/OS Configuration
To use the IBM Transformation Extender with Command Server	Command Server
To use the IBM Transformation Extender for Application Programming in your IMS execution environment	IMS/DC Execution Option
To use the IBM Transformation Extender for Application Programming in your CICS® execution environment	Command Server (z/OS CICS section)
To use the IBM Transformation Extender for Application Programming in your z/OS Batch execution environment	Platform API
To use the IBM Transformation Extender for Application Programming in your USS execution environment	TX Programming Interface C API Java™ API RMI API
To use the IBM Transformation Extender with Launcher	Launcher
To use the IBM Transformation Extender with Launcher in your USS execution environment	Launcher
To use the z/OS native administration interface for the IBM Transformation Extender with Launcher	Launcher
To use the IBM Transformation Extender for Integration Servers	<ul style="list-style-type: none">• IBM® Transformation Extender for IBM Integration Bus• IBM Transformation Extender for Sterling B2B Integrator
To bind the application plan package using the sample DTXBIND JCL	Installed sample readme files in SDTXSAMP PDS: <ul style="list-style-type: none">• DTXCDRME• DTXDMRME
To use the multiple versions of CICS Execution Option in a single CICS region	Command Server (z/OS CICS section)
To configure and use the map and mdq preloader exit	IMS/DC Execution Option
To create the resource name file (.mrn) on Windows	Resource Registry
To use the CICS Adapter	CICS Adapter
To use the DB/2 (z/OS ODBC) Adapter	DB/2 (z/OS ODBC) Adapter
To use the DB/2 (Windows/UNIX) Adapter	DB/2 (Windows/UNIX) Adapter

Related concepts

- [z/OS Configuration overview](#)

SMP/E installation

IBM® Transformation Extender for z/OS® features are installed through System Modification Program Extended (SMP/E).

For instructions about installing the IBM Transformation Extender for z/OS features, see the *Program Directory for IBM Transformation Extender for Use with z/OS* (order number: GI13-5666-01).

IBM Transformation Extender - Common Components

One of the features of the IBM® Transformation Extender for z/OS® product is the IBM Transformation Extender - Common Components.

The files that are installed with the IBM Transformation Extender - Common Components base feature are programs and JCL that are used with the other IBM Transformation Extender for z/OS optional features that you have installed.

- [Configuring the common components](#)

These are the required configuration steps for the IBM Transformation Extender - Common Components.

Configuring the common components

These are the required configuration steps for the IBM® Transformation Extender - Common Components.

- [Accessing DB2 databases in CICS and IMS environments](#)

To access DB2 databases in CICS and IMS environments, there are configuration steps that you must follow.

- [Accessing DB2 databases in Batch and USS environments](#)

To access DB2 databases in Batch and UNIX System Services (USS) environments, there are configuration steps that you must follow.

- [Allocating an zFS on USS environments](#)

To allocate an zFS on UNIX System Services (USS) environments, there are configuration steps that you must follow.

- [Defining programs to CICS region for IBM Transformation Extender CICS Adapter server-side components](#)

To define programs to CICS region for CICS® Adapter server-side components, there are configuration steps that you must follow.

Accessing DB2 databases in CICS and IMS environments

To access DB2 databases in CICS and IMS environments, there are configuration steps that you must follow.

When you are running the IBM® Transformation Extender in your CICS and IMS environments and need to access and manipulate data in your DB2 database, you must use the DB/2 (z/OS®) Call Attachment Facility (CAF) Adapter.

To use the DB/2 (z/OS) CAF Adapter, follow the steps described in [Binding the Data Base Request Module \(DBRM\)](#).

Related reference

- [Binding the Data Base Request Module \(DBRM\)](#)

Accessing DB2 databases in Batch and USS environments

To access DB2 databases in Batch and UNIX System Services (USS) environments, there are configuration steps that you must follow.

When you are running the IBM® Transformation Extender with Command Server feature in your Batch environment or the IBM Transformation Extender for Application Programming feature in your UNIX System Services (USS) runtime environment and need to access and manipulate data in your DB2 databases, you can use either the DB/2 (z/OS®) Call Attachment Facility (CAF) Adapter or the DB2 (z/OS ODBC) Adapter.

To use the DB/2 (z/OS) CAF Adapter, follow the steps described in [Binding the Data Base Request Module \(DBRM\)](#).

The prerequisite for using the DB2 (z/OS ODBC) Adapter is that the ODBC support is set up in the DB2 subsystem. For more information about installing and configuring ODBC, see the *DB2 UDB for z/OS ODBC Guide and Reference*. There are no additional configuration steps required to use the DB2 (z/OS ODBC) Adapter.

Allocating an zFS on USS environments

To allocate an zFS on UNIX System Services (USS) environments, there are configuration steps that you must follow.

Define a unique mount point on your UNIX System Services (USS) environment for the deployed runtime environment for the IBM® Transformation Extender USS features that you have installed. Run the DTXDHFS JCL that is installed in the DTX.SDTXSAMP PDS. The DTXDHFS JCL will allocate a Hierarchical File System (HFS) data set on your USS environment, mount it, and set the owner and file permissions.

HFS is deprecated and not supported on the newest z/OS systems. It is supported with ITX and can be used if required.

To allocate an HFS on UNIX System Services (USS) environments, there are configuration steps that you must follow.

Define a unique mount point on your UNIX System Services (USS) environment for the deployed runtime environment for the IBM Transformation Extender USS features that you have installed. Run the DTXDHFS JCL that is installed in the DTX.SDTXSAMP PDS. The DTXDHFS JCL will allocate a Hierarchical File System (HFS) data set on your USS environment, mount it, and set the owner and file permissions.

Defining programs to CICS region for IBM Transformation Extender CICS Adapter server-side components

To define programs to CICS region for CICS® Adapter server-side components, there are configuration steps that you must follow.

The programs can be defined to CICS by adding statements to the CICS CSD definitions. To update the CSD, use the definitions in the DTXCRDEF member of the DTX.SDTXSAMP PDS included in the installation. The DTXCRDEF file contains example RDO commands.

Add the DTX.SDTXLOAD load library to the DFHRPL concatenation for the CICS region that will be running the CICS Adapter server-side components.

IBM Transformation Extender with Command Server

One of the features of the IBM® Transformation Extender for z/OS® product is the IBM Transformation Extender with Command Server.

- [Configuring IBM Transformation Extender with Command Server](#)

These are the required configuration steps for the IBM Transformation Extender with Command Server.

Configuring IBM Transformation Extender with Command Server

These are the required configuration steps for the IBM® Transformation Extender with Command Server.

- [Binding the DB2 plan](#)
- [Examples](#)

There are several examples that are described in the readme files installed in the DTX.SDTXSAMP PDS and are used to demonstrate the various feature components.

Binding the DB2 plan

For the configuration steps for the Data Base Request Module (DBRM) that is included with the installation files, see [Binding the Data Base Request Module \(DBRM\)](#).

Examples

There are several examples that are described in the readme files installed in the DTX.SDTXSAMP PDS and are used to demonstrate the various feature components.

For your Batch environment:

DTXBMRME
 Burst Map example
DTXDMRME
 DB2 Map example
 run only if you will be using the DB2 Adapter
DTXMQRME
 WebSphere MQ example
 run only if you will be using the IBM MQ Adapter
DTXMSRME
 Resource Registry example
 run only if you want to try the Resource Registry alias feature

IBM Transformation Extender with Launcher

One of the features of the IBM® Transformation Extender for z/OS® product is the IBM Transformation Extender with Launcher.

- [Configuring IBM Transformation Extender with Launcher](#)

These are the required configuration steps for the IBM Transformation Extender with Launcher.

Configuring IBM Transformation Extender with Launcher

These are the required configuration steps for the IBM® Transformation Extender with Launcher.

- [Configuring UNIX System Services \(USS\)](#)

Configuring UNIX System Services (USS)

For additional configuration steps that are common to other IBM® Transformation Extender features installed on UNIX System Services (USS), see [Additional configuration on UNIX System Services \(USS\)](#) that is in [Information common to all features](#).

IBM Transformation Extender for Application Programming

One of the features of the IBM® Transformation Extender for z/OS® product is the IBM Transformation Extender for Application Programming.

The IBM Transformation Extender for Application Programming feature supports the following runtime environments:

- CICS®
- IMS/TM
- z/OS Software Development Kit
- UNIX System Services Software Development Kit
- [CICS environment](#)
The IBM Transformation Extender for Application Programming includes the CICS Execution Option through which you can run maps in your CICS environments.
- [IMS environment](#)
The IBM Transformation Extender for Application Programming includes the IMS/DC Execution Option through which you can run maps in your IMS environments.
- [UNIX System Services \(USS\) Software Development Kit](#)
The IBM Transformation Extender for Application Programming includes the UNIX System Services (USS) SDK through which you can run maps in your USS environments.

CICS environment

The IBM® Transformation Extender for Application Programming includes the CICS® Execution Option through which you can run maps in your CICS environments.

Prior to the 8.1 release, this functionality was provided in the CICS Command Server product.

- [CICS configuration requirements](#)
You must modify your CICS configuration to use IBM Transformation Extender for Application Programming.
- [Examples for the CICS environment](#)
There are several examples that are described in the readme files installed in the DTX.SDTXSAMP PDS and are used to demonstrate the various feature components.

CICS configuration requirements

You must modify your CICS configuration to use IBM® Transformation Extender for Application Programming.

The following modifications are described in this documentation:

- DFHRPL changes differ from prior releases.
- XPLINK requires an XPLINK Language Environment (LE) Library.
- [Updating CICS CSD definitions and DFHRPL](#)
The IBM Transformation Extender for Application Programming for the CICS runtime environment installs the DTXCTBL file in the DTX.SDTXSAMP PDS, which contains the RDO definitions.
- [XPLINK and non-XPLINK](#)
The CICS Execution Option takes advantage of the C/C++ Extra Performance Linkage (XPLINK) compiler optimization introduced in CICS Transaction Server for z/OS V 3.1.
- [Loading Maps](#)
To use the IBM Transformation Extender for Application Programming feature on CICS runtime environments, and to successfully complete the IVP, you must load the IBM Transformation Extender for Application Programming executable maps into a VSAM keyed-sequential data set (KSDS).
- [Language Environment \(LE\) requirement](#)
The transformation server is ported to CICS using the IBM "C/C++" compiler and requires the z/OS Language Environment runtime libraries.
- [\(Optional\) Setting up the DB2 \(z/OS\) Adapter to use the CICS CAF](#)
To use the DB2 (z/OS) Adapter, you will need to set it up to use the CICS DB/2 Attachment Facility.
- [Multiple versions of the IBM Transformation Extender for Application Programming for the CICS runtime environment in a single CICS region](#)
You can run multiple versions of the IBM Transformation Extender for Application Programming for the CICS runtime environment in a single CICS region.
- [Additional updates to CICS CSD definitions](#)
If you plan to run multiple, concurrently executing IBM Transformation Extender for Application Programming for the CICS runtime environment tasks that share

the map keyed-sequential data sets (KSDS), set the CSDSTRNO parameter for the map KSDS equal to the maximum number of concurrently executing tasks you want to allow in the z/OS CICS region.

- **Sizing temporary storage**
Temporary Storage Queue (TSQ) is used to back up IBM Transformation Extender for Application Programming work spaces required to run a map.
- **(Optional) Defining and loading the Resource Name file**
Optionally define and load the VSAM entry-sequenced data set (ESDS) that stores the DTXCRMNR resource name file.

Updating CICS CSD definitions and DFHRPL

The IBM® Transformation Extender for Application Programming for the CICS runtime environment installs the DTXCTBL file in the DTX.SDTXSAMP PDS, which contains the RDO definitions.

Use the supplied RDO definitions to update the CICS CSD definitions, defining transactions, programs and data sets for the region or regions where you will be using the IBM Transformation Extender for Application Programming for the CICS runtime environment.

XLINK and non-XLINK

The CICS® Execution Option takes advantage of the C/C++ Extra Performance Linkage (XLINK) compiler optimization introduced in CICS Transaction Server for z/OS V 3.1.

IBM® Transformation Extender for z/OS® includes two load libraries, DTX.SDTXLOAD and DTX.SDTXLOD2. The DTX.SDTXLOAD load library contains files that support the use of XLINK. If you want to use XLINK, you must add the DTX.SDTXLOAD load library to the DFHRPL concatenation.

- **DTXCTBL CSD definitions**
The IBM Transformation Extender for z/OS installation includes a DTXCTBL file containing the program definitions.

DTXCTBL CSD definitions

The IBM® Transformation Extender for z/OS® installation includes a DTXCTBL file containing the program definitions.

The DTXCTBL file is divided into the following program definition categories:

- program definitions for various components such as the mapper, resource manager, adapters and ICU
- transaction resource definitions
- file resource definitions

The DTXCTBL CSD is used for standard linkage. The DTXCTBL file contains the program definitions for the same product components, but the program names and the specific definitions are different.

The program definitions for the CICS drivers, DTXnnnCI (XLINK version) and DTXnnnKI (non-XLINK version) are used for the current version of IBM Transformation Extender for z/OS. The *nnn* variable represents the version number. An example is DTX820CI. The aliases for MERCCICS and DSTXCICS are provided for backwards compatibility when migrating from the Mercator or DSTX/CICS Command Server products in older releases.

- **DTXCTBL CSD definitions for XLINK**
The default settings in the DTXCTBL file are for the XLINK versions of the program definitions for IBM Transformation Extender for z/OS.
- **DTXCTBL CSD definitions for non-XLINK**
You will need to change the default settings in the DTXCTBL file for non-XLINK versions of the program definitions for IBM Transformation Extender for z/OS.

DTXCTBL CSD definitions for XLINK

The default settings in the DTXCTBL file are for the XLINK versions of the program definitions for IBM® Transformation Extender for z/OS®.

To use the XLINK versions of the program definitions, use the file the way it is delivered, with the default settings.

To use the non-XLINK versions of the program definitions, comment and uncomment the sections as described in [DTXCTBL CSD definitions for non-XLINK](#).

DTXCTBL CSD definitions for non-XLINK

You will need to change the default settings in the DTXCTBL file for non-XLINK versions of the program definitions for IBM® Transformation Extender for z/OS®.

Comment and uncomment the sections in the DTXCTBL file that is delivered with the IBM Transformation Extender for z/OS as described here:

- program definitions
 - comment the driver definitions in the XLINK section

```
* DEFINE PROGRAM(MERCCICS)
* GROUP(DTXCICS)
* DESCRIPTION(IBM Websphere Trans Ext/CICS DRIVER)
* LANGUAGE(ASSEMBLER)
* CONCURRENCY(THREADSAFE)
```

```

* DATA (ANY)
* DEFINE PROGRAM(DSTXCICS)
* GROUP (DTXCICS)
* DESCRIPTION(IBM Websphere Trans Ext/CICS DRIVER)
* LANGUAGE (ASSEMBLER)
* CONCURRENCY (THREADSAFE)
* DATA (ANY)

```

- uncomment the driver definitions in the non-XPLINK section

```

DEFINE PROGRAM(MERCCICS)
GROUP (DTXCICS)
DESCRIPTION(IBM Websphere Trans Ext/CICS DRIVER)
LANGUAGE (ASSEMBLER)
DATA (ANY)

DEFINE PROGRAM(DSTXCICS)
GROUP (DTXCICS)
DESCRIPTION(IBM Websphere Trans Ext/CICS DRIVER)
LANGUAGE (ASSEMBLER)
DATA (ANY)

```

- transaction resource definitions

- comment the XPLINK version

```

* XP Link definition of DTXT transaction
* DEFINE TRANSACTION(DTXT)
* GROUP (DTXCICS)
* DESCRIPTION(IBM Websphere Trans Ext/CICS SAMPLE
TEST PROGRAM)
* PROGRAM(DTXCTST)
* TASKDATALOC (ANY)

```

- uncomment the non-XPLINK version

```

* non-XP Link definition of DTXT transaction
DEFINE TRANSACTION(DTXT)
GROUP (DTXCICS)
DESCRIPTION(IBM Websphere Trans Ext/CICS SAMPLE
TEST PROGRAM)
PROGRAM(DTXKTST)
TASKDATALOC (ANY)

```

To use the non-XPLINK versions of the program definitions, comment and uncomment the sections as described above.

To use the XPLINK versions of the program definitions, use the file the way it is delivered, with the default settings as described in [DTXCTBL CSD definitions for XPLINK](#).

Loading Maps

To use the IBM® Transformation Extender for Application Programming feature on CICS runtime environments, and to successfully complete the IVP, you must load the IBM Transformation Extender for Application Programming executable maps into a VSAM keyed-sequential data set (KSDS).

To load the maps, do the following steps:

1. Define the VSAM cluster used to store IBM Transformation Extender for Application Programming executable maps. You can use the sample DTXCIJCL JCL to define the VSAM KSDS.
2. Extract, format, and load IBM Transformation Extender for Application Programming maps into the VSAM map KSDS. You can use the sample DTXCMJCL JCL that will extract, format, and load the maps into the VSAM KSDS that was defined by the DTXCIJCL JCL.

Language Environment (LE) requirement

The transformation server is ported to CICS using the IBM "C/C++" compiler and requires the z/OS Language Environment runtime libraries.

The z/OS IBM® Transformation Extender for Application Programming for the CICS runtime environment is compatible with all supported releases of Language Environment (LE).

This will require that LE support be installed in CICS.

(Optional) Setting up the DB2 (z/OS) Adapter to use the CICS CAF

To use the DB2 (z/OS) Adapter, you will need to set it up to use the CICS® DB/2 Attachment Facility.

You will be updating the CSD. For information about updating the CSD, see the *IBM CICS System Definition Guide*.

To set up the adapter to use the CICS DB/2 Attachment Facility:

1. Bind the Data Base Request Module (DBRM).
See [Binding the Data Base Request Module \(DBRM\)](#).
2. Create a DB2ENTRY in the CICS CSD for the DBUTILE plan.

3. Create DB2TRAN entries in the CICS CSD for each CICS transaction that will call the IBM Transformation Extender for Application Programming for the CICS runtime environment and invoke a DB2 map.
4. Create a DB2TRAN entry in the CICS command line for the DTXI (used for backward compatibility to a previous release) and DTXT transactions.
5. If your site uses the RCT macro, DSNCRCT, to define the entries for the CICS DB/2 Attachment Facility, you will need to code and assemble the DSNCRCT TYPE=ENTRY macro instruction to add the entry for the DBUTILE plan.

Multiple versions of the IBM Transformation Extender for Application Programming for the CICS runtime environment in a single CICS region

You can run multiple versions of the IBM® Transformation Extender for Application Programming for the CICS runtime environment in a single CICS region.

If you are installing this feature for the first time, continue with [Additional updates to CICS CSD definitions](#).

- **Description**
This feature enables people who use the IBM Transformation Extender for Application Programming for the CICS runtime environment or the Mercator Integration Broker or Ascential DataStage TX to run multiple versions of these products concurrently in a single CICS region.
- **Installing multiple versions**

Description

This feature enables people who use the IBM® Transformation Extender for Application Programming for the CICS runtime environment or the Mercator Integration Broker or Ascential DataStage TX to run multiple versions of these products concurrently in a single CICS region.

The benefit of this feature is the ability to do a staged migration to IBM Transformation Extender for Application Programming while being able to run your existing product release in the same CICS region.

This feature is only available starting with Ascential DataStage TX version 7.5. As of Ascential DataStage TX 7.5, only one prior release of Ascential DataStage TX can be used at the same time as this version.

For more information, see the topic about *Multiple Command Server versions* in the *z/OS CICS* section of the Command Server documentation in the IBM Transformation Extender information center that is published on the WebSphere Transformation Extender Library web page (<http://www.ibm.com/software/integration/wdatastagetx/library/index.html>). For other locations where you can find the Command Server documentation and additional documentation related to z/OS, see [Related publications for z/OS](#).

If you are not installing multiple versions of these products in the same CICS region, continue with [Additional updates to CICS CSD definitions](#).

Installing multiple versions

To install multiple versions, do the following steps:

1. Complete the feature installation steps as described in this documentation.
 - a. Install IBM® Transformation Extender for Application Programming for the CICS runtime environment in a region that has an existing version of the Mercator Integration Broker or Ascential DataStage TX.
Both versions will be separate instances of the product for the CICS execution environment that can run concurrently.
2. The CICS RDO definitions supplied with this release contain resource definitions that provide backwards compatibility for existing users of the product for the CICS runtime environment. These definitions will create conflicts and render the prior release unusable. Before installing the RDO definitions for this release, either comment out or delete the following items:
 - a. program entry: MERCCICS or DSTXCICS
 - b. transaction definitions:

MERC or DSTX
for the start CICS transaction
MERT, DSTT or DSTX
for the start the test sample transaction

If you do not comment out or delete the above items, these RDO definitions will create conflicts between an older release and the new release and as a result, you will not be able to use the prior release.
There are multiple program entry and transaction definitions options depending on which release of the product for the CICS runtime environment you currently have installed.
3. Do not delete the CICS RDO group, TSIMERC or ASCLDSTX.
4. Place the data set definition of the IBM Transformation Extender for Application Programming load library for the CICS runtime environment for this release after the load library of the existing release in the DFHRPL concatenation.

Additional updates to CICS CSD definitions

If you plan to run multiple, concurrently executing IBM® Transformation Extender for Application Programming for the CICS runtime environment tasks that share the map keyed-sequential data sets (KSDS), set the CSDSTRNO parameter for the map KSDS equal to the maximum number of concurrently executing tasks you want to allow in the z/OS CICS region.

If you are adding the IBM Transformation Extender for Application Programming to an existing IBM Transformation Extender for Application Programming installation (meaning, that this is not a new installation), you must update the CSD. Unless you are planning on running multiple versions of IBM Transformation Extender for Application Programming in a single CICS region, remove the ASCLDSTX group from the CSD and add the new system definitions.

The use of CICS Temporary Storage Queues might increase if you upgrade from Mercator Integration Broker or Ascential Datastage TX. The VSAM-based work space manager has been replaced with a TSQ-based work space manager.

The benefits of using a TSQ-based work space manager is that it simplifies the installation process and improves the performance of your map execution. The storage requirements for TSQ might be impacted. You must ensure that there is enough available TSQ.

For more information about how to size temporary storage, see [Sizing temporary storage](#).

Sizing temporary storage

Temporary Storage Queue (TSQ) is used to back up IBM® Transformation Extender for Application Programming work spaces required to run a map.

Work space refers to a physical store that is used to shadow input and output data. Shadow data is usually created when the map uses an adapter or Transient Data Queues, or the map is expecting the data to contain record separators. Work spaces are also used by the core transformation services to store information about the data that is being mapped.

Because you can develop a map with no inputs and outputs, work spaces might not be used in some situations.

The minimum number of possible work spaces for a map execution is 1. The maximum possible number is $1 + (2 * \text{number of inputs}) + (2 * \text{number of outputs})$.

TSQ will not be used by a map if the map's source and targets are passed as memory buffers and the workspace in memory option (-WM) is used.

Each work space is keyed by a unique TSQ qname. The format of the qname is TXnnnnNdddddd, where nnnn is the CICS task id and ddddddd is a sequence number.

In theory, the maximum size of a single work space is 1,073,545,221 bytes.

To calculate the TSQ space requirements for a particular map, you need to determine the maximum data size of each input and each output.

The formula for a specific input is:

```
TSQ size = (max data size * 5)
```

The criteria for the multiplier of 5 is 1 for the shadow file and 4 for the work space used by the core transformation service. The amount of space required by the core transformation service might actually be less. It depends on a number of factors such as the number of objects being validated. Four (4) is typical for a complex type tree with lots of types defined.

- [Sizing temporary storage example](#)

This is an example of how the temporary storage for a map is sized.

Sizing temporary storage example

This is an example of how the temporary storage for a map is sized.

Assume that your map has two inputs and one output. Input 1 is a WebSphere MQ message. Input 2 is a lookup from a DB/2 table. The target destination for the output is to DB/2.

The MQ message is 100,000 bytes. The DB/2 lookup is 1,200 bytes. The target destination for the output is to DB/2 and is 25,000 bytes.

Work space estimates:

Input

```
500,000 = 100,000 + (100,000 * 4)
```

Input 2

```
6,000 = 1,200 + (1,200 * 4)
```

Output 1

```
25,000 = 25,000
```

Map total size is: 531,000.

In this example, the actual amount of TSQ used will be greater based on the fact that the size of the TSQ records used by the work space manager is 32,763. Also, the map's page size setting will determine the minimum amount of space used by the core.

Assuming a 64K page size setting, the estimate would be:

Input 1

```
622,497 = ((100,000 / 32,763) + 32,763) +
((400,000 / 65,535) + 65,535) / 32,763 + 32,763
```

Input 2

```
131,052 = 32,763 + ((65,535 / 32,763) + 32,763)
```

Output 1

32,763

New map total size is 786,312.

The actual required amount will depend on how many 64K pages are necessary to validate the input. An additional factor for which you need to account when sizing temporary storage is the maximum number of concurrent transactions that will be running the map.

Assuming there can be 3 concurrent tasks, the total temporary storage would be:

$$2,358,936 = 3 * 786,312$$

(Optional) Defining and loading the Resource Name file

Optionally define and load the VSAM entry-sequenced data set (ESDS) that stores the DTXCRMNRN resource name file.

The step should be done only if you want to use the IBM® Transformation Extender for Application Programming resource name file, and successfully complete the IVP for the Resource Registry example described in the DTXCRRME readme file.

To define and load the resource name file (.mrn), run the DTXRRJCL example JCL that is in the DTX.SDTXSAMP PDS.

To use the resource name file on CICS, create the resource name file. For information about creating the resource name file (.mrn) on Windows, see the IBM Transformation Extender Resource Registry documentation. The DTXCRMNRN example resource name file is provided in the IBM Transformation Extender for Application Programming installation in the DTX.SDTXSAMP PDS.

Examples for the CICS environment

There are several examples that are described in the readme files installed in the DTX.SDTXSAMP PDS and are used to demonstrate the various feature components.

For the required configuration steps, see [Loading Maps](#).

DTXVRMME

Burst Map example

DTXCDRMME

DB2 Map example

run only if you will be using the DB2 Adapter and the DB2 plan

DTXCRMME

WebSphere MQ example

run only if you will be using the IBM MQ Adapter

DTXCRREME

Resource Registry example

run only if you will be using the Resource Registry

for configuration requirements, see [\(Optional\) Defining and loading the Resource Name file](#).

IMS environment

The IBM® Transformation Extender for Application Programming includes the IMS/DC Execution Option through which you can run maps in your IMS environments.

- [Configuration steps for the IMS environment](#)

There might be configuration steps required for you to do before you can use the IBM Transformation Extender for Application Programming feature, depending on your business needs and if specific system administration and set up prerequisites have not yet been done in your IMS environment.

Configuration steps for the IMS environment

There might be configuration steps required for you to do before you can use the IBM® Transformation Extender for Application Programming feature, depending on your business needs and if specific system administration and set up prerequisites have not yet been done in your IMS environment.

For more information, see the IMS/DC Execution Option documentation. For other locations where you can find the IMS/DC Execution Option documentation and additional documentation related to z/OS, see [Related publications for z/OS](#).

- [\(Optional\) Binding the DB2 plan](#)

(Optional) Binding the DB2 plan

For configuration steps for the Data Base Request Module (DBRM) that is included with the installation files, see [Binding the Data Base Request Module \(DBRM\)](#).

UNIX System Services (USS) Software Development Kit

The IBM® Transformation Extender for Application Programming includes the UNIX System Services (USS) SDK through which you can run maps in your USS environments.

For additional configuration steps that are common to other IBM Transformation Extender for Application Programming features installed on UNIX System Services (USS), see [Additional configuration on UNIX System Services \(USS\)](#), that is in [Information common to all features](#).

For information about using the API on USS, see the Platform API documentation in the IBM Transformation Extender information center that is published on the WebSphere Transformation Extender Library web page (<http://www.ibm.com/software/integration/wdatastagetx/library/index.html>). For other locations where you can find the Platform API documentation and additional documentation related to z/OS, see [Related publications for z/OS](#).

Information common to all features

There is configuration information that is common across the different IBM® Transformation Extender features.

- [Binding the Data Base Request Module \(DBRM\)](#)

These are the steps for binding the Data Base Request Module (DBRM) that are common across the different IBM Transformation Extender features.

- [Additional configuration on UNIX System Services \(USS\)](#)

To use IBM Transformation Extender features in your USS environments, there are additional configuration activities that you might need to do.

- [Preserving the encoding of XML files](#)

To preserve the encoding of XML files that were created on an ASCII platform, transfer the files (including files such as the Resource Registry .mrc and .mrn files) in BINARY mode (for example, by using FTP or IND\$FILE). Note that ASCII files that are transferred in BINARY mode are still ASCII-encoded, and therefore are not readable on the mainframe platform.

- [Support for Getting Started Sub-capacity Pricing \(GSSP\)](#)

All IBM Transformation Extender runtime environments on z/OS systems support Getting Started Sub-capacity Pricing (GSSP).

- [Support for z/OS XML System Services processing](#)

The IBM Steling Transformation Extender for z/OS Batch, API, IMS, and CICS can use z/OS XML System Services to parse and, optionally, validate XML documents.

Binding the Data Base Request Module (DBRM)

These are the steps for binding the Data Base Request Module (DBRM) that are common across the different IBM® Transformation Extender features.

- [Upgrading from an earlier version](#)

If you are using the DB2 (z/OS) Adapter with your current release of the IBM Transformation Extender and with a prior release or prior releases of the IBM Transformation Extender concurrently, it is recommended that you bind the application plans as a package.

- [Binding the DBUTILE application plan package](#)

You must bind the DBRM to use the DB/2 (z/OS®) Call Attachment Facility (CAF) Adapter.

Related reference

- [Accessing DB2 databases in CICS and IMS environments](#)

Upgrading from an earlier version

If you are using the DB2 (z/OS) Adapter with your current release of the IBM® Transformation Extender and with a prior release or prior releases of the IBM Transformation Extender concurrently, it is recommended that you bind the application plans as a package.

This will allow multiple releases of the IBM Transformation Extender to be bound to the same application plan.

Binding the DBUTILE application plan package

You must bind the DBRM to use the DB/2 (z/OS®) Call Attachment Facility (CAF) Adapter.

There is only one Data Base Request Module (DBRM), named DBUTILE. It is contained in the DTXDBUTE member of the DTX.SDTXSAMP PDS that is included with the installation files. It can be bound as the DBUTILE plan by running the DTXBIND JCL that is included in the feature installation.

The DBUTILE plan name is used when the DB2 Adapter connects to the DB2 subsystem where its DB request module was bound.

For more instructions on binding the application plan by using the sample DTXBIND JCL, see the readme files DTXCDRME and DTXDMRME that are installed with DTX.SDTXSAMP PDS.

Additional configuration on UNIX System Services (USS)

To use IBM® Transformation Extender features in your USS environments, there are additional configuration activities that you might need to do.

- [Modifying the DTXINST JCL](#)

The DTXINST JCL job is used to create the IBM Transformation Extender UNIX System Services (USS) runtime environment.

- [Setting USS environment variables](#)

There are several UNIX System Services (USS) environment variables that you need to set.

- [Using IBM Transformation Extender for Application Programming on USS](#)

IBM Transformation Extender for Application Programming contains Java™ interface components that run the UNIX System Services (USS) SDK on z/OS® operating systems in a UNIX environment. Because the native character-encoding of USS differs from the native character-encoding of JVM, you must set type tree properties to accommodate the difference in encoding.

- [\(Optional\) Defining DB2 system load library](#)

When you use either the DB2 (z/OS) Adapter or DB2 (z/OS ODBC) Adapter in your UNIX System Services (USS) runtime environment, you need to define the DB2 system load library to the system linklist.

- [\(Optional\) Defining IBM MQ system load libraries](#)

When you use the IBM MQ Adapter in your UNIX System Services (USS) runtime environment, you need to define the IBM MQ system load libraries to the system linklist.

- [\(Optional\) Accessing Oracle 9i database](#)

The Oracle 9i Client Adapter is available for you to use in maps on your UNIX System Services (USS) runtime environment.

- [\(Optional\) Using Language Environment parameters](#)

The IBM Transformation Extender UNIX System Services (USS) runtime environment is a 31-bit application. The Language Environment (LE) parameters at many sites are optimized for COBOL programs and 24-bit applications.

- [\(Optional\) Defining port numbers when using the E-mail Adapter](#)

The E-mail Adapter is available for use in maps run on your UNIX System Services (USS) runtime environment.

Modifying the DTXINST JCL

The DTXINST JCL job is used to create the IBM® Transformation Extender UNIX System Services (USS) runtime environment.

Run this job if you have not yet run it to deploy the installed files on USS and only after you have installed one or more of the following features:

- JDTXnn2 IBM Transformation Extender for Application Programming
- JDTXnn3 IBM Transformation Extender for Integration Servers
- JDTXnn4 IBM Transformation Extender with Launcher

where *nn* is the release number. An example is JDTX904, the IBM Transformation Extender with Launcher V9.0 feature.

Before you run the DTXINST JCL, make the following modifications:

1. Modify the **SET INSTHOME** statement in the JCL to define the location of your IBM Transformation Extender - Common Components base feature installation if it is different than the default location settings. The default location is: /usr/lpp/dtx/VnRnMn/IBM/common.
2. Update the JAVA_HOME_DIR environment variable in the STDENV DD statement of the INSTALL job step. Change the *javahome* variable to the directory location of the IBM Runtime Environment, Java™ Technology Edition for your system. See the product [release notes](#) for the required version of IBM Runtime Environment, Java Technology Edition.

For example, change

```
JAVA_HOME_DIR=javahome
```

to

```
JAVA_HOME_DIR=/usr/lpp/java/IBM/J1.6
```

3. Update the BIT_PREFERENCE keyword in the STDENV DD statement of the INSTALL job step. Change the *BITS* variable to the word length of the installation runtime library and binary file selections. The *BITS* variable is one of the following:

Option	Description
64	64-bit library and binary files

4. Modify the **SET CRTHOME** statement in the JCL to define the location of the directory structure where you want the UNIX System Services (USS) runtime environment for the optional features you installed to be created if it is different than the default location settings. The default location is: /u/dtxuser/dtx.

When you run the DTXINST JCL, it will run the **SET** commands, and then run the following two steps: **MAKEDIR** and **INSTALL**.

- [MAKEDIR step](#)

The **MAKEDIR** step of the DTXINST JCL runs the BPXBATCH MVS™ utility program, which uses a **MKDIR** command to create the IBM Transformation Extender home directory.

- [INSTALL step](#)

The **INSTALL** step of the DTXINST JCL runs the BPXBATCH MVS utility program, which runs the createuser shell script from the bin directory of the IBM Transformation Extender - Common Components base feature.

Related reference

- [MAKEDIR step](#)
- [INSTALL step](#)

MAKEDIR step

The **MAKEDIR** step of the DTXINST JCL runs the BPXBATCH MVS™ utility program, which uses a **MKDIR** command to create the IBM® Transformation Extender home directory.

Related tasks

- [Modifying the DTXINST JCL](#)

INSTALL step

The **INSTALL** step of the DTXINST JCL runs the BPXBATCH MVS™ utility program, which runs the createuser shell script from the bin directory of the IBM® Transformation Extender - Common Components base feature.

The createuser shell script creates the following subdirectories in the IBM Transformation Extender home directory (DTX_HOME_DIR) location, specified in the **CRTHOME** setting, and installs the files from the location specified in the **INSTHOME** setting into these subdirectories:

- bin
- config
- data
- dump
- libs
- logs
- maps
- OSGI Bundle
- src
- systems
- tmp
- wmqi

Note: The wmqi and OSGI Bundle subdirectories are created if you have installed the IBM Transformation Extender for Integration Servers optional feature.

The createuser script creates symbolic links to the bin, libs, and config directories of the installed IBM Transformation Extender features. It also creates the setup shell script and optionally creates the deploy shell script for the IBM Transformation Extender for Integration Servers if you have installed this optional feature.

The **INSTALL** step uses the **INSTHOME** symbolic parameter to identify the installed location of the common components directory and the **CRTHOME** symbolic parameter to identify the location of the user runtime directory.

Related tasks

- [Modifying the DTXINST JCL](#)

Setting USS environment variables

There are several UNIX System Services (USS) environment variables that you need to set.

1. Export the **_CEE_RUNOPTS** environment variable.

If you are using IBM® Runtime Environment, Java™ Technology Edition on a z/OS® operating system and you want to use the IBM Transformation Extender in your UNIX System Services (USS) runtime environment, you must add **XPLINK** to the **_CEE_RUNOPTS** USS environment variable. For example:

```
export _CEE_RUNOPTS=$_CEE_RUNOPTS,"XPLINK(ON)"
```

2. Modify the setup script in the current directory to set the **DISPLAY**, **JAVAHOME** environment variable.

The setup script defines the environment variables required to run the IBM Transformation Extender. The **DISPLAY**, **JAVAHOME** environment variable are defined in the following way:

```
DISPLAY  
      Address of the X Server  
JAVAHOME  
      Installation directory of JRE on the z/OS environments
```

The **DISPLAY** and **JAVAHOME** variables are commented out in the script. Either uncomment these environment variables in the setup script or include them in your profile (.profile). If you include these settings in your profile in your UNIX System Services (USS) home directory, you can avoid having to reset these variables each time you install the IBM Transformation Extender in your USS environment.

3. Set the environment variables for IBM Transformation Extender.

Enter " . ./setup" at the command prompt from the directory where you deployed the runtime environment under UNIX System Services (USS) for the IBM Transformation Extender features that you have installed.

Now you can begin to run the IBM Transformation Extender that you installed on your UNIX System Services (USS) execution environment.

Using IBM Transformation Extender for Application Programming on USS

IBM® Transformation Extender for Application Programming contains Java™ interface components that run the UNIX System Services (USS) SDK on z/OS® operating systems in a UNIX environment. Because the native character-encoding of USS differs from the native character-encoding of JVM, you must set type tree properties to accommodate the difference in encoding.

The USS environment uses EBCDIC encoding, and the Java JVM uses UTF-8 encoding. LATIN-1 data is the most appropriate data type to exchange with the JVM.

When you write a map to call a Java-based adapter on USS, set the following properties:

- In the type tree, set the Data language property to LATIN-1 for all character text data items that are to be passed to the adapter.
- The Java-based adapter typically returns a LATIN-1 encoded byte stream. In most cases, use the following type tree properties to set the format of output results that are to be passed from the adapter to the map:
 - Set the Item Subclass property to Text.

- o Set the Interpret as property to Binary.

These settings are especially important if the results are to be used as input into another map or adapter.

For additional information about the Java programming code page considerations, see the reference information for Java on z/OS on the IBM web site.

(Optional) Defining DB/2 system load library

When you use either the DB2 (z/OS) Adapter or DB2 (z/OS ODBC) Adapter in your UNIX System Services (USS) runtime environment, you need to define the DB2 system load library to the system linklist.

To define the DB2 system load library to the system linklist, do one of the following steps:

1. Define the hlq.SDSNLOAD DB2 system load library to the system linklist or
2. Export the STEPLIB variable with the location of the DB2 load library by using the following command:

```
export  
STEPLIB=hlq. SDSNLOAD
```

where hlq is the high-level qualifier for the DB2 installation.

Each time you run the IBM® Transformation Extender, you can do one of the following steps:

- use this export statement at the shell prompt or
- add the export statement to the /etc/profile or the .profile file of the specific user of the feature

(Optional) Defining IBM MQ system load libraries

When you use the IBM® MQ Adapter in your UNIX System Services (USS) runtime environment, you need to define the IBM MQ system load libraries to the system linklist.

To define the IBM MQ system load libraries to the system linklist, do one of the following steps:

1. Define the IBM MQ system load libraries to the system linklist or
2. Export the STEPLIB variable with the location of the IBM MQ load library by using the following command:

```
export STEPLIB=hlq..SCSQANLE:hlq..SCSQAUTH
```

where hlq is the high-level qualifier for the IBM MQ installation.

Each time you run the IBM Transformation Extender, you can do one of the following steps:

- use this export statement at the shell prompt or
- add the export statement to the /etc/profile or the .profile file of the specific user of the feature

(Optional) Accessing Oracle 9i database

The Oracle 9i Client Adapter is available for you to use in maps on your UNIX System Services (USS) runtime environment.

This adapter allows access to data stored in Oracle 9i databases on remote systems.

To be able to access data using the Oracle 9i Client Adapter:

1. Install and configure Oracle 9i.
2. Be sure that Oracle 9i is included in the PATH= environment variable and that other environment variables required by the Oracle 9i database are set appropriately. Environment variables required by the Oracle 9i database that are involved with translation from ASCII to EBCDIC require particularly close attention.

Support is not currently provided for direct connection to an Oracle database on the same z/OS system on which the IBM® Transformation Extender is executing.

(Optional) Using Language Environment parameters

The IBM® Transformation Extender UNIX System Services (USS) runtime environment is a 31-bit application. The Language Environment (LE) parameters at many sites are optimized for COBOL programs and 24-bit applications.

To prevent the application from running out of memory because it lacked storage below the 16 megabyte (MB) line, and then terminating abnormally with an abend code such as 80A, the LE parameters listed later in this section are recommended.

Note: These LE parameters are documented in the *z/OS: Language Environment Programming Reference* publication.

Put the following LE parameters in an export statement to be added to the .profile file of the specific user of the feature.

Memory Management

```
STACK(128K,64K,ANY,FREE)  
HEAP(256K,64K,ANY,FREE)  
HEAPPOLS(ON)
```

Error Management

```
TER(UADUMP) or TER(UAIMM)
```

Examples:

(The following LE parameters must be placed all on one line, without any space between them.)

```
_CEE_RUNOPTS="TER(UADUMP),STACK(128K,64K,ANY,FREE),HEAPPOOLS(ON),  
HEAP(256K,64K,ANY,FREE)"
```

(Optional) Defining port numbers when using the E-mail Adapter

The E-mail Adapter is available for use in maps run on your UNIX System Services (USS) runtime environment.

To ensure correct operation of the E-mail Adapter, you need to define the port numbers for SMTP and pop3 in the TCPIP.ETC.SERVICES or /etc/services files.

1. Define the well-known port number for SMTP as 25.
2. Define the well-known port number for pop3 as 110.

For a complete list and discussion of well-known port numbers, see the Internet Assigned Numbers Authority (IANA) website at <http://www.iana.org>.

Preserving the encoding of XML files

To preserve the encoding of XML files that were created on an ASCII platform, transfer the files (including files such as the Resource Registry .mrc and .mrn files) in BINARY mode (for example, by using FTP or IND\$FILE). Note that ASCII files that are transferred in BINARY mode are still ASCII-encoded, and therefore are not readable on the mainframe platform.

An XML instance document might contain an encoding attribute that identifies the code page of its content. If you transfer an XML file to your mainframe in ASCII mode or convert it to EBCDIC, you must manually change the encoding setting in the resulting XML instance document to the correct code page in order to ensure that the XML parser correctly processes the file. In most cases, change UTF-8 encoding to IBM-1047 encoding.

Support for Getting Started Sub-capacity Pricing (GSSP)

All IBM® Transformation Extender runtime environments on z/OS systems support Getting Started Sub-capacity Pricing (GSSP).

After IBM Transformation Extender registers with the hosting subsystem, GSSP system management facility (SMF) accounting records are collected continuously on behalf of IBM Transformation Extender for the hosting subsystem's address space.

Support for z/OS XML System Services processing

The IBM Sterling Transformation Extender for z/OS Batch, API, IMS, and CICS can use z/OS XML System Services to parse and, optionally, validate XML documents.

However, due to new compiler restrictions, IBM Sterling Transformation Extender for z/OS on USS can no longer use z/OS XML System Services to parse and validate XML documents.

- [Invoking z/OS System Services parsing](#)

IBM Transformation Extender for z/OS® uses z/OS XML System Services parsing when you supply an input schema in OSR format.

- [z/OS XML System Services parsing limitations](#)

Invoking z/OS System Services parsing

IBM® Transformation Extender for z/OS® uses z/OS XML System Services parsing when you supply an input schema in OSR format.

To use z/OS XML System Services to parse an XML document, supply the input schema as a binary Optimized Schema Representation (OSR) file. Use the z/OS **xsdorg** command to generate the OSR file. Specify the path to the OSR file in the Schema.>Type.>Metadata property of the input card. See [z/OS XML System Services User's Guide and Reference](#) for a description of the **xsdorg** command and the list of data encodings that z/OS XML System Services supports.

- To parse an OSR file without validation, set the input card DocumentVerification.>Xerces property to Well formed.
- To parse an OSR file with validation, set the input card DocumentVerification.>Xerces property to Schema Validation.

When the Schema.>Type.>Metadata property points to an XSD file instead of an OSR file, IBM Transformation Extender uses its internal native-schema parsing to parse the file.

z/OS XML System Services parsing limitations

The following limitations apply to z/OS XML System Services processing:

- Because the z/OS XML System Services parser stops parsing when it encounters an error, native-schema-enabled maps that use z/OS XML System Services parsing do not support the REJECT function.
- The z/OS XML System Services parser and the **xsdorg** command flag null characters at the end of a file as invalid data. To use z/OS XML System Services processing, specify variable-length record format for the input data set. Fixed-length records pad the data set with nulls to the specified block size.
- Native-schema-enabled maps that use the z/OS XML System Services parser do not support CDATA.

In addition, these limitations that are common to native-schema-enabled maps also apply to native-schema-enabled maps that are parsed by z/OS XML System Services:

- IBM® Transformation Extender maps do not handle XML schema sequences that contain elements with the same name. When an XML schema contains a sequence of elements that do not have unique names, a map creates the output correctly only for the first element. The map creates the output with empty values for all the subsequent elements that have the same name.
- A map that runs successfully in versions of WebSphere Transformation Extender earlier than version 8.2.0.4 might fail to run in version 8.2.0.4 and later versions. The process of mapping items from the schema changed in version 8.2.0.4, and you might have to revise the map.

IBM Transformation Extender for Integration Servers overview

IBM® Transformation Extender for Integration Servers packages extenders to IBM products.

It provides installation and integration support when combining IBM Transformation Extender with additional IBM products. It includes IBM Transformation Extender components that are ready to deploy in message flows, mediation flows, and business processes. IBM Transformation Extender complements the native capabilities of these products, and can process large documents and messages with more complex formats that are not based on XML.

The following IBM Transformation Extender products are packaged as IBM Transformation Extender for Integration Servers:

- IBM Transformation Extender for IBM Integration Bus
- IBM Transformation Extender for Sterling B2B Integrator
- IBM Transformation Extender for IBM Business Process Manager Advanced

System requirements

IBM® Transformation Extender for Integration Servers requirements are detailed in the [system requirements](#).

IBM Transformation Extender for Sterling B2B Integrator

IBM Transformation Extender for Sterling B2B Integrator combines the business process management features of IBM® Sterling B2B Integrator with the universal transformation capability of IBM Transformation Extender. This integration of business-to-business (B2B) gateway and universal transformation solution gives you new options for solving B2B and application-to-application (A2A) integration problems.

- **[Product uses for Sterling B2B Integrator customers](#)**
IBM Transformation Extender includes a transformation engine, a graphical map editor, and industry packs that provide pre-defined support for multiple document standards, and a range of technology adapters.
- **[Product uses for IBM Transformation Extender customers](#)**
If you already use IBM Transformation Extender, you can invoke your existing IBM Transformation Extender maps in predefined business processes for business-to-business (B2B) processing, or create new processes that use Sterling B2B Integrator functionality to address your specific needs.
- **[Advantages of IBM Transformation Extender for Sterling B2B Integrator](#)**
The integration of IBM Transformation Extender with Sterling B2B Integrator provides map reuse, support for industry standards, and support for complex maps.
- **[Examples](#)**
You can learn about IBM Transformation Extender for Sterling B2B Integrator by running the examples that are provided with the product. The examples illustrate how to use IBM Transformation Extender for Sterling B2B Integrator to accomplish specific tasks. The examples and readme files that describe them are located in the `tx_install_dir/examples` directory.
- **[Standards, globalization, and data-type support](#)**
Sterling B2B Integrator services are developed using industry-accepted specifications for data formats, communication protocols, workflow modeling, and security in order to maximize interoperability between systems and trading partners.
- **[Transforming data](#)**
When a sender and receiver have different message format specifications, you can design your business process to transform the data in a document before delivering it. You can design a business process to reformat a document, enrich a message, or perform numeric and string calculations based on the value of certain fields.
- **[WTX Map service overview](#)**
The WTX Map service is a component of IBM Transformation Extender for Sterling B2B Integrator. The WTX Map service provides the ability to run a IBM Transformation Extender map within a Sterling B2B Integrator business process. The WTX Map service combines the data transformation power of IBM Transformation Extender with the business integration capabilities of Sterling B2B Integrator.
- **[Services that support IBM Transformation Extender maps](#)**
In addition to the WTX Map service, certain Sterling B2B Integrator services can run IBM Transformation Extender maps.
- **[Business process concepts](#)**
A *business process* is a goal-driven, ordered flow of activities that accomplishes a business objective. A *service* is a set of instructions that Sterling B2B Integrator uses to perform an activity in a business process. *Process data* is data that a process stores in an XML document when the process runs. For example, process data that a service extracts from a document can be used to determine which service runs next in the process.
- **[Sterling B2B Integrator user interface permissions](#)**
You must explicitly set specific permissions to use many of the interfaces in the Sterling B2B Integrator dashboard.
- **[Developing a business process that uses the WTX Map service](#)**
You can develop a business process that uses a IBM Transformation Extender map to transform data.
- **[Deploying IBM Transformation Extender maps](#)**
The location where you store a IBM Transformation Extender map determines which services can use the map.
- **[Testing IBM Transformation Extender maps](#)**
You can use IBM Transformation Extender Design Studio to manually test the IBM Transformation Extender map on the Sterling B2B Integrator server.
- **[Running a business process](#)**
A business process can run on a predefined schedule, in response to an activity, or manually.
- **[Configuring the Translation service to use IBM Transformation Extender maps](#)**
When the Sterling B2B Integrator Translation service uses a IBM Transformation Extender map, set the `exhaust_input` setting to YES to enable the service to process all of the input records.

- **Problem determination**

You can enable trace and audit logs to diagnose WTX Map service errors. You can configure the WTX Map service to extract data from a document or a process for tracking purposes.

- **Performance tuning**

The `tx_install_dirconfig.yaml` file contains the `runtime/stream/MaxMemLimit` property and the `runtime/stream/MaxMemDirectory` property. When data passed to or from an adapter exceeds the size (in MB) that is configured on the `StreamMaxMemLimit` property, the data is paged to a temporary file to limit the memory consumption of the process. The `StreamMaxMemDirectory` property specifies the directory where the temporary files are located.

Product uses for Sterling B2B Integrator customers

IBM Transformation Extender includes a transformation engine, a graphical map editor, and industry packs that provide pre-defined support for multiple document standards, and a range of technology adapters.

You can embed IBM Transformation Extender within an application or run it in parallel to an application. With different engine configurations, you can use the transformation capabilities of IBM Transformation Extender through:

- Direct integration with specific IBM® products
- The API
- The command line
- Event-based triggers

IBM Transformation Extender has these unique features:

- Transformation of multiple inputs and multiple outputs in a single translation process
- High throughput execution
- Code-free design and deployment
- Robust design and testing functions through the Eclipse IDE
- Ontological data model for all data types
- Support for industry standards such as HL7, SEPA, NCPDP, and ACORD

IBM Transformation Extender addresses the business needs that are common to both application-to-application (A2A) and business-to-business (B2B) integration. Integration with Sterling B2B Integrator adds these capabilities:

- Trading partner management
- Enveloping and de-enveloping or bulking and debulking
- Document management and archive
- Automated acknowledgment generation
- Reporting
- Managed file transfer with Sterling File Gateway

IBM Transformation Extender maps can run in existing or new business processes. Within a business process, Sterling B2B Integrator logging, alerts, and error messages are active. IBM Transformation Extender has additional logging and management facilities that can provide product-specific detail. If you have experience with either product, you can use the troubleshooting tools that you know.

Business needs determine whether to use the adapters that are available with IBM Transformation Extender or the adapters that are available with Sterling B2B Integrator.

Product uses for IBM Transformation Extender customers

If you already use IBM Transformation Extender, you can invoke your existing IBM Transformation Extender maps in predefined business processes for business-to-business (B2B) processing, or create new processes that use Sterling B2B Integrator functionality to address your specific needs.

As a B2B gateway, Sterling B2B Integrator provides:

- Integrated communications
- Trading partner management
- Message enveloping and de-enveloping
- Automated acknowledgment generation
- A set of pre-defined EDI and B2B processes

Table 1. Sterling B2B Integrator Capabilities

Sterling B2B Integrator Capabilities	Details
Security	<ul style="list-style-type: none">• Identity management, including authorization and authentication• Perimeter security at DMZ traversal• Role-based data access and system operation• Secured mailbox repository

Sterling B2B Integrator Capabilities	Details
Communications protocols	<ul style="list-style-type: none"> Web services (SOAP) S/FTP/S client and server HTTP and HTTPS SMTP AS1, AS2, AS3 RosettaNet WebDAV Zengin TCP/IP EBICS (France) IBM® Sterling Connect:Direct®
Business Process Management	<ul style="list-style-type: none"> Graphical process modeling tool Business process execution engine
Community Management	<ul style="list-style-type: none"> Manage and grow trading partner communities Centralized visibility into trading partner communities Digital certificates deployment
Application Extensibility and Customization	<ul style="list-style-type: none"> Web services: Support for SOAP, WDSL Eclipse-based support for creating custom adapters and services
Connectivity and Adapters	A comprehensive library of more than 300 available adapters, including: <ul style="list-style-type: none"> Oracle Manugistics PeopleSoft Siebel Vantive JD Edwards I2 IBM® Sterling Connect:Direct® IBM® Sterling Connect:Enterprise® IBM® Sterling Gentran®:Server®, GXS
Mobility	Monitoring and management of Sterling B2B Integrator processes and status from a mobile digital device, including system status, database growth, and average business-process wait time.

Related reference

- [Standards, globalization, and data-type support](#)
- [Services that support IBM Transformation Extender maps](#)

Advantages of IBM Transformation Extender for Sterling B2B Integrator

The integration of IBM Transformation Extender with Sterling B2B Integrator provides map reuse, support for industry standards, and support for complex maps.

- Reuse existing maps
If you already use IBM Transformation Extender for A2A or B2B transformation, you can combine existing business processes (or the sample EDI translation business processes) in Sterling B2B Integrator with your existing IBM Transformation Extender maps.
- Industry standards support
IBM Transformation Extender supports financial services and healthcare industry standards in a Sterling B2B Integrator business process. Maps that you create with either IBM Transformation Extender or Sterling B2B Integrator can coexist within Sterling B2B Integrator.
- Implementation of complex many-to-many maps
Business needs can require the transformation of one source document into multiple target documents, or many-source-to-many-target transformation. With IBM Transformation Extender, a single process can address these transformation requirements without the need for multiple translation processes.

Related concepts

- [WTX Map service overview](#)
- [Transforming data](#)

Related reference

- [Services that support IBM Transformation Extender maps](#)

Examples

You can learn about IBM Transformation Extender for Sterling B2B Integrator by running the examples that are provided with the product. The examples illustrate how to use IBM Transformation Extender for Sterling B2B Integrator to accomplish specific tasks. The examples and readme files that describe them are located in the `tx_install_dir/examples` directory.

Standards, globalization, and data-type support

Sterling B2B Integrator services are developed using industry-accepted specifications for data formats, communication protocols, workflow modeling, and security in order to maximize interoperability between systems and trading partners.

Sterling B2B Integrator supports standards for:

- Internet transports
- Cryptographic services
- Document formats
- Document-enveloping formats
- Business-process sequencing
- Web services

Sterling B2B Integrator is built on a Java™ code base. Because Java supports Unicode, the universal character-encoding scheme, your business processes can interact with data that is written in nearly any language. Virtually any file-based, message-based, or stream-based data type is also supported.

Related concepts

- [Product uses for IBM Transformation Extender customers](#)

Transforming data

When a sender and receiver have different message format specifications, you can design your business process to transform the data in a document before delivering it. You can design a business process to reformat a document, enrich a message, or perform numeric and string calculations based on the value of certain fields.

Sterling B2B Integrator provides built-in services for you to use in your business processes. IBM Transformation Extender for Sterling B2B Integrator extends these data transformation and validation capabilities by enabling you to use IBM Transformation Extender maps and type trees within the Sterling B2B Integrator environment. You can create IBM Transformation Extender type trees that model complex data and use component rules to validate the syntax and semantics of the data in your messages.

For information about the services, see the related topics in the [Sterling B2B Integrator documentation](#).

Related reference

- [Advantages of IBM Transformation Extender for Sterling B2B Integrator](#)

WTX Map service overview

The WTX Map service is a component of IBM Transformation Extender for Sterling B2B Integrator. The WTX Map service provides the ability to run a IBM Transformation Extender map within a Sterling B2B Integrator business process. The WTX Map service combines the data transformation power of IBM Transformation Extender with the business integration capabilities of Sterling B2B Integrator.

Within Sterling B2B Integrator, you can use the Graphical Process Modeler (GPM) to configure a business process. When IBM Transformation Extender for Sterling B2B Integrator is installed on a Sterling B2B Integrator server, the WTX Map service icon is available in the GPM palette for you to include in the business process. Alternatively, you can directly edit the Business Process Modeling Language (BPML) meta-language of a business process to add the WTX Map service.

In a business process, a series of steps process a document. A service performs each step. The WTX Map service receives input from Sterling B2B Integrator, runs the IBM Transformation Extender map that is configured in the map service settings, and creates one or more Sterling B2B Integrator documents as output. The map can have multiple input cards and multiple output cards. Each input card and output card can be associated with a Sterling B2B Integrator document.

When an input card does not receive a document, the card pulls data from the adapter that is specified on the card. If an output card is associated with a document, the map service creates or overwrites the document. If an output card is not associated with a document, the map services sends the data to the adapter that is specified on the output card. Maps can also read and create documents dynamically. You can create a dynamic document by using the IBM Transformation Extender PUT rule.

The WTX Map service can load a compiled IBM Transformation Extender map from the Sterling B2B Integrator map repository or from a local file system on the Sterling B2B Integrator server. The WTX Map service also can dynamically load a map at runtime when the map is specified as a workflow variable.

Related tasks

- [Developing a business process that uses the WTX Map service](#)

Related reference

- [WTX Map service](#)
- [Specifying the map for the WTX Map service](#)

Services that support IBM Transformation Extender maps

In addition to the WTX Map service, certain Sterling B2B Integrator services can run IBM Transformation Extender maps.

These Sterling B2B Integrator services can run IBM Transformation Extender maps that are checked into the Sterling B2B Integrator map repository:

- EDIFACT Enveloping
- EDIFACT De-enveloping
- Translation
- X12 Envelope
- X12 De-envelope

The Sterling B2B Integrator X12 Envelope, X12 De-envelope, and EDIFACT De-enveloping services can use the compliance checkers that are built into the IBM Transformation Extender X12, HIPAA, and EDIFACT Packs as an alternative to the native compliance-check functionality in Sterling B2B Integrator.

See the IBM Transformation Extender Packs documentation for information about compliance-checking and services.

Related concepts

- [Product uses for IBM Transformation Extender customers](#)

Related reference

- [Advantages of IBM Transformation Extender for Sterling B2B Integrator](#)

Business process concepts

A *business process* is a goal-driven, ordered flow of activities that accomplishes a business objective. A *service* is a set of instructions that Sterling B2B Integrator uses to perform an activity in a business process. *Process data* is data that a process stores in an XML document when the process runs. For example, process data that a service extracts from a document can be used to determine which service runs next in the process.

IBM Transformation Extender for Sterling B2B Integrator installs the WTX Map service into Sterling B2B Integrator. The WTX Map service and certain Sterling B2B Integrator services can run a IBM Transformation Extender map as part of a business process.

See the [Sterling B2B Integrator documentation](#) for details about business processes, services, and process data.

Related reference

- [Services that support IBM Transformation Extender maps](#)

Sterling B2B Integrator user interface permissions

You must explicitly set specific permissions to use many of the interfaces in the Sterling B2B Integrator dashboard.

For details, see the description of the permissions needed to access UI resources in the Security documentation in the [Sterling B2B Integrator documentation](#).

Developing a business process that uses the WTX Map service

You can develop a business process that uses a IBM Transformation Extender map to transform data.

To use a IBM Transformation Extender map in a business process:

1. Create a IBM Transformation Extender map.
2. Compile the map and deploy it to Sterling B2B Integrator.
3. Create the business process by using the Sterling B2B Integrator Graphical Process Modeler (GPM) or by editing the Business Process Modeling Language (BPML).
4. Edit the business process to add the WTX Map service.
5. Edit the WTX Map service to identify the IBM Transformation Extender map that you want to run.

In the GPM service editor, use the Message To Service tab to identify the map to run and use either:

MapServerLocation

Specifies a map on the local file system of the Sterling B2B Integrator server. If you configure MapServerLocation, you must also configure the LocalMap field if you want to retrieve and edit the map settings from the file system.

MapName

Specifies a map in the Sterling B2B Integrator map repository.

If you configure both MapName and MapServerLocation, the WTX Map services uses the MapName value.

6. Deploy the business process to Sterling B2B Integrator.

- [WTX Map service](#)

Sterling B2B Integrator users who are familiar with the format of Sterling B2B Integrator service descriptions can use this overview of the WTX Map service. Details

about configuring, deploying, running, and debugging the WTX Map service are covered by the other topics in this information unit.

- **[Specifying the map for the WTX Map service](#)**

The WTX Map service can run a map that is in the file system on the Sterling B2B Integrator server or in the Sterling B2B Integrator map repository. At runtime, a service that runs before the WTX Map service can dynamically override the map that is configured on the WTX Map service.

- **[Map caching](#)**

Map caching can improve the runtime performance of the WTX Map service. You enable map caching at design time. You can cache any map except for dynamic maps.

- **[Resource Registry](#)**

The resource configuration file defines resources and aliases that are available to all WTX Map services that run on the Sterling B2B Integrator.

- **[Input and output cards](#)**

The IBM Transformation Extender map editor represents map input data and output data as map *cards*. The WTX Map service can host maps that have any number of input and output cards. Optionally, you can associate each card with a Sterling B2B Integrator document.

- **[Overriding map and card properties by using map settings](#)**

You can override the settings of a IBM Transformation Extender map that is configured for the WTX Map service. The map can be stored in either the map repository or in the file system of the Sterling B2B Integrator server.

- **[Accessing the Sterling B2B Integrator data harness from maps](#)**

Any service that runs IBM Transformation Extender maps reads and writes to the underlying database of Sterling B2B Integrator through a data harness. The data harness interacts with the Sterling B2B Integrator engine to perform functions such as getting data and setting correlations.

- **[Sending output data from a map to a Sterling B2B Integrator document](#)**

You configure the WTX Map service to send output data to a Sterling B2B Integrator document by configuring the PUT map rule on the Sterling adapter. The IBM Transformation Extender PUT map rule can create a Sterling B2B Integrator document immediately, or based on the success or failure of the map.

Related concepts

- [WTX Map service overview](#)

Related reference

- [WTX Map service BPML](#)
 - [XML entities](#)
-

WTX Map service

Sterling B2B Integrator users who are familiar with the format of Sterling B2B Integrator service descriptions can use this overview of the WTX Map service. Details about configuring, deploying, running, and debugging the WTX Map service are covered by the other topics in this information unit.

System name

WTX Map service

Graphical Process Modeler (GPM) category

IBM Transformation Extender

Description

Runs a IBM Transformation Extender map in a Sterling B2B Integrator business process to transform data.

Business usage

Performs translation of the input document within a business process.

Usage example

See the examples and readme files located in the `tx_install_dir\examples` directory.

Preconfigured?

No

Requires third-party files?

No

Platform availability

All supported application platforms.

Related services

No

Application requirements

The IBM Transformation Extender map that the WTX Map service runs must be located in the Sterling B2B Integrator or on the file system of the Sterling B2B Integrator server.

The connection to the Sterling B2B Integrator server must be configured by using the IBM Transformation Extender Design Studio. See the Map Designer documentation.

Initiates business processes?

No

Invocation

Runs as part of a business process.

Business process context considerations

The well-known `dynamicMap` work flow context variable points to a document object that stores a compiled map that the WTX Map service can load at runtime.

Returned status values

See the problem determination topics in this information unit.

Restrictions

No

Persistence level

None

Testing considerations

See the examples and readme files located in the `TX_install_directory\examples` directory.

Related concepts

- [WTX Map service overview](#)
-

Specifying the map for the WTX Map service

The WTX Map service can run a map that is in the file system on the Sterling B2B Integrator server or in the Sterling B2B Integrator map repository. At runtime, a service that runs before the WTX Map service can dynamically override the map that is configured on the WTX Map service.

- [Using maps from the map repository](#)
Maps that are checked into the Sterling B2B Integrator map repository can be used by the WTX Map service and any Sterling B2B Integrator service that supports IBM Transformation Extender maps. You can check in a map to the Sterling B2B Integrator map repository by using the Sterling B2B Integrator administrative console or the IBM Transformation Extender Design Studio.
- [Using maps on the server file system](#)
You can configure the WTX Map service in your business process to use a IBM Transformation Extender map on the Sterling B2B Integrator server.
- [Dynamically specifying maps at runtime](#)
At run time, you can override the compiled map that you specified in the WTX Map service with a compiled map that is specified in process data.
- [Dynamically loading RUN maps from the Sterling B2B map repository](#)
You can store a RUN map in the Sterling B2B Integrator map repository. A RUN map can be dynamically loaded by the WTX Map service and by other Sterling B2B Integrator services.

Related concepts

- [WTX Map service overview](#)
-

Using maps from the map repository

Maps that are checked into the Sterling B2B Integrator map repository can be used by the WTX Map service and any Sterling B2B Integrator service that supports IBM Transformation Extender maps. You can check in a map to the Sterling B2B Integrator map repository by using the Sterling B2B Integrator administrative console or the IBM Transformation Extender Design Studio.

Before you can use Design Studio to check a map into the map repository, you must:

- Install IBM Transformation Extender for Integration Servers on the computer where Design Studio is installed.
- Use Design Studio to configure the connection to the Sterling B2B Integrator server. See the IBM Transformation Extender Map Designer documentation for details.

To use Design Studio to check a map into the map repository, right-click a source map and select Deploy Map to Sterling B2B Integrator. Design Studio automatically compiles the map before storing it in the repository.

See the [Sterling B2B Integrator documentation](#) for details about checking in maps by using the administrative console.

To configure the WTX Map service to run a map in the Sterling B2B Integrator repository, right-click the WTX Map service in the Graphical Process Modeler and click Properties. Specify the name of the map in the repository in the MapName field.

Using maps on the server file system

You can configure the WTX Map service in your business process to use a IBM Transformation Extender map on the Sterling B2B Integrator server.

In the Sterling B2B Integrator Graphical Process Modeler, right-click the WTX Map service and click Properties. Use the MapServerLocation field to specify a compiled map on the file system of the Sterling B2B Integrator server.

Dynamically specifying maps at runtime

At run time, you can override the compiled map that you specified in the WTX Map service with a compiled map that is specified in process data.

A business process can derive either the map name or the map itself from process data. For example, the /ProcessData/here document can contain the name of another ProcessData document, /ProcessData/there:

```
/ProcessData/here="ProcessData/there"  
/ProcessData/there="hello"
```

- When /ProcessData/here is used as an XPATH expression, it resolves to hello.
- When /ProcessData/here is not used as an XPATH expression, it resolves to /ProcessData/there.

A business process searches for the map to execute in this order:

1. A map from a document

A compiled map can be stored in process data and passed to the WTX Map service.

Source
Process data

Location
A document called dynamicMap

Purpose
To run the compiled map that is supplied as an array of bytes in the process data.

Result
The WTX Map service runs the supplied map.

Storing large maps in the dynamicMap property can affect performance and memory usage. In general, write a large map to disk and reference it from the MapServerLocation property.

2. A map in the Sterling B2B Integrator repository

Source
WTX Map service
Location
/ProcessData/MapName

Purpose
To run a compiled map that is stored in the Sterling B2B Integrator Map Repository.
Result
The specified map runs, unless it is overridden by MapServerLocation or dynamicMap.

3. A map from a file system on the Sterling B2B Integrator server

At runtime, override the map server location with the map server location that was placed in the process data by a prior service in the business process:

Source
The file system
Location
/ProcessData/MapServerLocation

Purpose
To run a compiled map that is in a location other than the location that is configured on the WTX Map service.
Result
The Sterling B2B Integrator business process overrides the map server location with the location that you specified in the process data, and the WTX Map service runs the map from this location instead.

To specify a map in the map repository, omit the file extension from the MapName. To specify a map in the file system on the server, include the file extension in the map name.

When a map name includes a file extension, IBM Transformation Extender for Sterling B2B Integrator searches for the map on the file system of the Sterling B2B Integrator server. If the map is not in the file system, IBM Transformation Extender for Sterling B2B Integrator strips the file extension from the map name and searches for the map name in the Sterling B2B Integrator map repository.

When a map name omits a file extension, IBM Transformation Extender for Sterling B2B Integrator searches the map repository first, then searches the file system.

For more information about process data, see the [Sterling B2B Integrator documentation](#).

Dynamically loading RUN maps from the Sterling B2B map repository

You can store a RUN map in the Sterling B2B Integrator map repository. A RUN map can be dynamically loaded by the WTX Map service and by other Sterling B2B Integrator services.

When a RUN map is in the map repository, the RUN function call must reference its name in the repository. For example:

```
RUN ("map_name_in_repository", ....)
```

Map caching

Map caching can improve the runtime performance of the WTX Map service. You enable map caching at design time. You can cache any map except for dynamic maps.

Map caching can improve performance because the WTX Map service does not load the compiled map every time the map runs. The WTX Map service loads the compiled map only on the first invocation of the map.

When map caching is enabled, changes that you make to the compiled map are not loaded until the Sterling B2B Integrator server is restarted. When map caching is disabled, any changes that you make to the compiled map are loaded the next time the WTX Map service runs.

Map caching can be enabled or disabled for each instance of the WTX Map service. In the Graphical Process Modeler, set the CacheMap property in the Message to Service tab. Alternatively, configure the BPMN with one of the following assignments:

```
<assign to="CacheMap">YES</assign>
<assign to="CacheMap">NO</assign>
```

You can enable RUN map caching in the [RUN Maps] section of the config.yaml configuration file. Set /runtime/Run Maps/RunMapMaxCacheNum to the maximum number of RUN maps to cache.

Related reference

- [WTX Map service BPML](#)
 - [Running multiple concurrent instances of maps](#)
 - [Performance tuning](#)
-

Resource Registry

The resource configuration file defines resources and aliases that are available to all WTX Map services that run on the Sterling B2B Integrator.

For detailed information about the Resource Registry, see the IBM Transformation Extender documentation.

- [Modifying resource names](#)
Use the IBM Transformation Extender editor to create resource aliases to be used with IBM Transformation Extender resources in Sterling B2B Integrator.
 - [Specifying the Resource Registry location](#)
Configure the Resource Registry location on the ResourceFile= property in the *Sterling_installation_dir\properties\translator.properties_wtx_ext.in* file.
-

Modifying resource names

Use the IBM Transformation Extender editor to create resource aliases to be used with IBM Transformation Extender resources in Sterling B2B Integrator.

You can use resource aliases only on a WTX Map service. You can create aliases for these IBM Transformation Extender resources that are used in WTX Map services:

- Unwired source and target resources that are specified in map cards
- GET and PUT functions in map rules
- RUN map overrides
- Audit file names
- Trace file names
- Back up file names
- Work space locations

For a complete list of the resources, see the IBM Transformation Extender Resource Registry documentation.

Specifying the Resource Registry location

Configure the Resource Registry location on the ResourceFile= property in the *Sterling_installation_dir\properties\translator.properties_wtx_ext.in* file.

Input and output cards

The IBM Transformation Extender map editor represents map input data and output data as map *cards*. The WTX Map service can host maps that have any number of input and output cards. Optionally, you can associate each card with a Sterling B2B Integrator document.

Relative paths on input and output cards resolve to the *Sterling_installation_dir\logs* directory.

- [Wired and unwired cards](#)
Input and output cards are considered to be *wired* when the card receives data from or sends data to a Sterling B2B Integrator document. An *unwired* card receives data from or sends data to the IBM Transformation Extender adapter that is specified on the card.
- [Configuring input and output cards in maps](#)
The WTX Map service can host a map that has any number of input and output cards. Optionally, you can associate an input or output card with a Sterling B2B Integrator document.
- [XML entities](#)
There are three ways to read from or write to an XML document: by using a classic type tree, by using a Xerces type tree, or by using a native schema. Optionally, a Xerces or native type tree can validate against any XML schema.

Related reference

- [XML entities](#)
-

Wired and unwired cards

Input and output cards are considered to be *wired* when the card receives data from or sends data to a Sterling B2B Integrator document. An *unwired* card receives data from or sends data to the IBM Transformation Extender adapter that is specified on the card.

Sterling B2B Integrator services associate only the first input card and the last output card with the primary document.

Configuring input and output cards in maps

The WTX Map service can host a map that has any number of input and output cards. Optionally, you can associate an input or output card with a Sterling B2B Integrator document.

You use the Sterling B2B Integrator Service Editor in the Graphical Process Modeler (GPM) to define input and output cards and to associate a card with a document.

1. Right-click the WTX Map service icon and click Properties to open the Service Editor.
 2. In the Message To Service tab, click Advanced.
 3. Click Add to create a new row in the table.
 4. In the Name column, specify the card that is to be associated with a document.
 - The name `in1` specifies input card 1, `in19` specifies input card 19, and so on.
 - The name `out1` specifies output card 1, `out11` specifies output card 11, and so on.
 5. Optional: In the Value column, specify the Sterling B2B Integrator document that is to be associated with the card.
- For example:
- `/ProcessData/PrimaryDocument` associates the document called PrimaryDocument that is located in ProcessData with the selected card. If the Value is an XPath location that points to another location, select Use XPATH.
 - `/ProcessData/SomeDocument/@SCIObjectID` accesses the document called SomeDocument by using the SCI ObjectID.
 - `/ProcessData/Document ID/text()` accesses the document by using the Document ID. For example, when the Document ID is `67797813f405b8237node1`, the path is `/ProcessData/67797813f405b8237node1/text()`

XML entities

There are three ways to read from or write to an XML document: by using a classic type tree, by using a Xerces type tree, or by using a native schema. Optionally, a Xerces or native type tree can validate against any XML schema.

When you use a native schema (instead of a type tree) to define a wired card, use the root of the schema (XSD) to define the card type. The native schema must be in the Sterling B2B Integrator repository.

If the map is specified by MapServerLocation, the schema can be on the file system. If a map is a dynamic map or a map in the repository, the schema must be in the Sterling B2B Integrator repository.

Related tasks

- [Developing a business process that uses the WTX Map service](#)

Related reference

- [Input and output cards](#)

Overriding map and card properties by using map settings

You can override the settings of a IBM Transformation Extender map that is configured for the WTX Map service. The map can be stored in either the map repository or in the file system of the Sterling B2B Integrator server.

You use the Service Editor in the Sterling B2B Integrator Graphical Process Modeler (GPM) to retrieve and modify the configured map settings. You can override the settings only by using the GPM. You cannot override the adapter settings of an unwired card.

1. In GPM, right-click the WTX Map service and click Properties to open the Service Editor.
 2. In the MapSettings field, click Map Settings to retrieve the map settings and populate the field:
 - If the MapName field is configured in the Service Editor, the map settings are retrieved from the specified map in the map repository.
 - If the MapServerLocation field and the LocalMap field are configured in the Service Editor, the map settings are retrieved from the specified map on the Sterling B2B Integrator server.To override the properties of a map that is stored in the file system of the Sterling B2B Integrator server, the IBM Transformation Extender Design Studio must be installed on the computer where you are running GPM.
 3. In the Map Settings window, edit the map settings and click OK to save the new settings or Reset to clear the settings.
The revised settings are saved as an XML string and used at runtime.
- [**Required property for secure Map Test HTTP Server adapter**](#)
When the Sterling B2B Integrator Map Test HTTP Server adapter is configured to use the secure socket layer (SSL), the client computer must use a secure connection to retrieve map settings from the Sterling B2B Integrator map repository.

Required property for secure Map Test HTTP Server adapter

When the Sterling B2B Integrator Map Test HTTP Server adapter is configured to use the secure socket layer (SSL), the client computer must use a secure connection to retrieve map settings from the Sterling B2B Integrator map repository.

To use a secure connection to retrieve map settings from the Sterling B2B Integrator map repository:

1. On the client computer, go to `C:\Documents and Settings\Administrator\pmodeler\host_IP_address\dashboard_port_number\gbm.properties`.
2. Set the following property:

```
WTXMapSettings_SecureConnection = true
```

When the Map Test HTTP adapter setting and the client configuration do not match, an error message displays. To resolve the problem, correct the WTXMapSettings_SecureConnection property in the gbm.properties file and restart the GPM.

Table 1. Incompatible Adapter and Client Configurations and Error Messages

Map-Test HTTP Adapter Setting	Client Configuration	Error Message
Use SSL	WTXMapSetting s_SecureConnection = false	Unable to send SOAP Request to Host: http://hostName:adapter_port/MapServices. The following exception occurred: Unexpected end of file from server Possible protocol error (http/https). Contact server administrator. Map settings retrieval operation was not successful.
Do not use SSL	WTXMapSetting s_SecureConnection = true	Unable to send SOAP Request to Host: https://hostName:adapter_port/MapServices. The following exception occurred: Unrecognized SSL message, plaintext connection? Map settings retrieval operation was not successful.

Accessing the Sterling B2B Integrator data harness from maps

Any service that runs IBM Transformation Extender maps reads and writes to the underlying database of Sterling B2B Integrator through a data harness. The data harness interacts with the Sterling B2B Integrator engine to perform functions such as getting data and setting correlations.

The WTX Map service can invoke the data harness functions by configuring either:

- The PUT and GET rules in the map
- An input card or output card command line that specifies the Sterling adapter.

By using the Sterling adapter, maps can read Sterling B2B Integrator data on input cards and from GET rules, and write to tables from output cards and PUT rules. To execute the harness function as soon as the PUT rule is processed, use the -NOW argument. The -NOW executes the function even if the map fails.

When the PUT rule does not include the -NOW argument, the function executes when the map completes.

When the map runs successfully, the output from the adapter is associated with a Sterling B2B Integrator document for use by normal stream adapters.

- **Data harness functions**

The WTX Map service supports data harness functions that operate on Sterling B2B Integrator correlations, process data, code lists, delimiters, and the transaction register. You can use data harness functions in map rules and type trees.

Data harness functions

The WTX Map service supports data harness functions that operate on Sterling B2B Integrator correlations, process data, code lists, delimiters, and the transaction register. You can use data harness functions in map rules and type trees.

Command	Description
getBinaryDocument	Retrieves the specified binary document.
getEnvelope	Retrieves the specified field from an envelope.
setCorrelation	Updates the Sterling B2B Integrator correlation table with the specified name/value pair for the document or business process.
getProcessData	Retrieves process data from the XML document object model (DOM) specified by the XPath. A business process can store and retrieve process data in order to access the data during process execution. See the Sterling B2B Integrator documentation for details about process data.
setProcessData	Stores process data from the current field into the XML DOM that is specified by the XPath. A business process can store and retrieve process data in order to access the data during process execution. See the Sterling B2B Integrator documentation for details about process data.
getCodeListItemBySenderCode	Retrieves a trading-partner code-list entry from Sterling B2B Integrator by sender code. A <i>trading partner code list</i> consists of one or many pairs of code values containing a sender code and a receiver code. Sterling B2B Integrator uses code pairs in code lists to identify items in transactions between two or more trading partners. See the Sterling B2B Integrator Trading Partner documentation for details about code lists.
getCodeListItemByReceiverCode	Retrieves a trading-partner code list from Sterling B2B Integrator by receiver code. A <i>trading partner code list</i> consists of one or many pairs of code values containing a sender code and a receiver code. Sterling B2B Integrator uses code pairs in code lists to identify items in transactions between two or more trading partners. See the Sterling B2B Integrator Trading Partner documentation for details about code lists.
getTransactionRegister	Compares the fields that you loaded into the Transaction Register with the input data, returns a code, and clears the Transaction Register.
0	Success; no duplicate data found.
1	Failure; duplicate data found.
8	No updates performed to this transaction register yet. Cannot perform a select before an update. The Transaction Register is empty. You must invoke setTransactionRegister to load the Transaction Register before you can invoke getTransactionRegister .

The following example:

```
PUT("Sterling", "-t+ -c setTransactionRegister -k 1", "Reg 1")
GET("Sterling", "-t+ -c getTransactionRegister")
GET("Sterling", "-t+ -c getTransactionRegister")
```

1. Loads Transaction Register field 1 with the literal value Reg 1.
2. Compares the input document data to the content of the Transaction Register and returns 0 or 1.
3. Attempts to invoke **getTransactionRegister** again without loading the Transaction Register, resulting in return code 8.

The following example loads Transaction Register field 3 with the contents of the specified input data field:

```
PUT("Sterling", "-t+ -c setTransactionRegister -k 3", PONumber Field:In1)
```

setTransactionRegister

Loads the specified Transaction Register field with the specified data. You can load up to six fields of data into memory for the Transaction Register. See the [See the Sterling B2B Integrator documentation](#) for details about the size limits of the fields.

After you run the **setTransactionRegister** command, you can run the **getTransactionRegister** command to check for duplicate input data.

getDelimiter

Gets a delimiter from an X12 or EDIFACT envelope.

This example **PUT** rule uses the Sterling adapter to create an element under process data that is called "getmap." The -NOW option creates the key even if the map fails:

```
=PUT("Sterling", "-t+ -c setProcessData -k getmap -now", "data")
```

This example GET rule uses the Sterling adapter to read the value of a key called "getmap" from the process data:

```
=GET rule " + VALID( GET("Sterling", "-t+ -c getProcessData -k getmap")
```

This example of a Sterling adapter command line on a card creates a correlation that is called "adapter" when the card completes. The correlation includes all of the data that was built on the card:

```
-c setCorrelation -k adapter -t+
```

The Sterling adapter supports the harness functions with the following command line arguments:

Table 1. Sterling adapter command line arguments

Command Long Form	Command Short Form	Description
-Command	-C	Command
-Key	-K	Key
-Now	-N	Executes the harness function as soon as the PUT rule is processed, even if the map fails
-T	-T	Adapter trace (overwrite)
-T+	-T+	Adapter trace (append)
-TE	-TE	Adapter trace error (overwrite)
-TE+	-TE+	Adapter trace error (append)
-TV	-TV	Adapter trace verbatim (overwrite)
-TV+	-TV+	Adapter trace verbatim (append)
-Code	-D	Sender/Receiver code
-Table	-B	Code table name

This table describes the data harness functions and key values that you can use on the Sterling adapter:

Table 2. Sterling adapter commands and key values

Command	Key
getBinaryDocument	The name of the document to retrieve.
setCorrelation	The name of the correlation to update.
getProcessData	The XPATH to select.
setProcessData	The XPATH to update.
getCodeListItemBySenderCode	The sender value for the code-list entry to retrieve. The command requires a sender code, codeListTable name, and column.
getCodeListItemByReceiverCode	The receiver value for the code-list entry to retrieve. The command requires a receiver code, codeListTable name, and column.
getEnvelope	The envelope field name.
getTransactionRegister	There are no keys for this command.
setTransactionRegister	Each entry is a number between 1 and 6 that indicates the field to update. See the Sterling B2B Integrator documentation for details about the field-size limits.
getDelimiter	An array of numbers that indicates the fields to select. The list is defined in Table 4 .

This table lists the sender and receiver keys and values to use with the getCodeListItemBySenderCode and getCodeListItemByReceiverCode functions.

Table 3. Sender and receiver keys and values

Key	Value
COL_SENDER	0
COL_RECEIVER	1
COL_DESC	2
COL_TEXT1	4
COL_TEXT2	5
COL_TEXT3	6
COL_TEXT4	7
COL_TEXT5	8
COL_TEXT6	9
COL_TEXT710	10
COL_TEXT8	11
COL_TEXT9	12

This table lists the keys and values to use with the getDelimiter function.

Table 4. Delimiter keys and values

Key	Value
SEGMENT_TERMINATOR	0
TAG_DELIMITER	1
ELEMENT_DELIMITER	2
SUB_ELEMENT_DELIMITER	3
REPEATING_ELEMENT_DELIMITER	4
RELEASE_CHAR	5
DECIMAL_SEPARATOR	6

This table lists the keys and values to use with the setTransactionRegister function.

Table 5. Transaction Register
keys and values

Key	Value
COL_TEXT1	0
COL_TEXT2	1
COL_TEXT3	2
COL_TEXT4	3
COL_TEXT5	4
COL_TEXT6	5

Sending output data from a map to a Sterling B2B Integrator document

You configure the WTX Map service to send output data to a Sterling B2B Integrator document by configuring the PUT map rule on the Sterling adapter. The IBM Transformation Extender PUT map rule can create a Sterling B2B Integrator document immediately, or based on the success or failure of the map.

- [Sending output data by using a PUT wire](#)
The IBM Transformation Extender map PUT rule sends output data from a WTX Map service to a document, either immediately, or based on the outcome of the map. When the output terminal is wired, the data is propagated to the next service in the business process.
- [WTX Map service BPML](#)
You can customize this WTX Map service Business Process Modeling Language (BPML) example to call your own maps. However, to edit map settings, use the Graphical Process Modeler (GPM).
- [Validating the business process modeling language](#)
When you save a business process model in the Graphical Process Modeler (GPM), by default the GPM automatically verifies that the Business Process Modeling Language (BPML) is valid and well-formed. The validation does not verify the configuration of the services within the business process.

Sending output data by using a PUT wire

The IBM Transformation Extender map PUT rule sends output data from a WTX Map service to a document, either immediately, or based on the outcome of the map. When the output terminal is wired, the data is propagated to the next service in the business process.

- [Configuring the WTX Map service PUT rule](#)
To send output data from the WTX Map service to a document, you create a map that uses the PUT rule and create a business process that includes a WTX Map service.
- [PUT rule syntax](#)
To dynamically create a document from a IBM Transformation Extender map, configure a map rule to use the PUT function and the WIRE keyword. Optionally, you can use the -NOW argument to create the document as soon as the PUT function is processed, rather than when the map completes.

Configuring the WTX Map service PUT rule

To send output data from the WTX Map service to a document, you create a map that uses the PUT rule and create a business process that includes a WTX Map service.

PUT rule syntax

To dynamically create a document from a IBM Transformation Extender map, configure a map rule to use the PUT function and the WIRE keyword. Optionally, you can use the -NOW argument to create the document as soon as the PUT function is processed, rather than when the map completes.

The map rule must use the PUT function and specify WIRE as the adapter name:

```
=PUT("WIRE", "output_document",data)
```

Optionally, you can specify that the document is to be created immediately. The -NOW argument creates the output document as soon as the PUT rule is processed. When you omit the -NOW argument, the WTX Map service creates the document when the map completes, and a RUN map creates the document when the parent map completes. When you specify the -NOW argument, the document is created even if the map or RUN map fails.

```
=PUT("WIRE", "output_document -NOW",data)
```

The WIRE and -NOW keywords are not case-sensitive.

WTX Map service BPML

You can customize this WTX Map service Business Process Modeling Language (BPML) example to call your own maps. However, to edit map settings, use the Graphical Process Modeler (GPM).

This BPML example illustrates how you might configure a WTX Map service. It assumes you are familiar with coding in BPML.

```
<process name="default">
  <operation name="WTXSIMapService">
    <participant name="wtxmap"/>
    <output message="WTXSIMapServiceInputMessage">
      <assign to=". " from="*"></assign>
      <assign to="CacheMap">NO</assign>
      <assign to="LocalMap">C:\test\map1.mmc</assign>
      <assign to="MapServerLocation">D:\Sterling\samples\test\map1.mmc</assign>
      <assign to="Threshold"></assign>
      <assign to="MapSettings"><![CDATA[<!-- map settings go here --&gt;]]&amp;gt;&lt;/assign&gt;
      &lt;assign to="in9"&gt;C:\test\GolfCountryClub.xml&lt;/assign&gt;
      &lt;assign to="out4"&gt;C:\test\output.xml&lt;/assign&gt;
    &lt;/output&gt;
    &lt;input message="inmsg"&gt;
      &lt;assign to=". " from="*"></assign>
    </input>
  </operation>
</process>
```

Where:

operation name

Must be WTXSIMapService.

participant name

The name of the IBM Transformation Extender map that the service is to run.

CacheMap

Specifies whether the map should be cached (YES) or loaded each time the service runs (NO). See the description of map caching.

LocalMap

The path to the compiled map in a local file system on the Sterling B2B Integrator server.

MapServerLocation

The path to the compiled map on the Sterling B2B Integrator server.

MapSettings

The map settings in the form of an XML string. Use the Graphical Process Modeler (GPM) to edit the map settings.

Threshold

The input data size limit that determines whether the WTX map service runs in a separate JVM.

in9

The name of the ninth input card.

out4

The name of the fourth output card.

Related tasks

- [Developing a business process that uses the WTX Map service](#)

Related reference

- [Map caching](#)
- [Based on input-data-size threshold](#)

Validating the business process modeling language

When you save a business process model in the Graphical Process Modeler (GPM), by default the GPM automatically verifies that the Business Process Modeling Language (BPML) is valid and well-formed. The validation does not verify the configuration of the services within the business process.

See the [Sterling B2B Integrator documentation](#) for details about business process model validation.

Deploying IBM Transformation Extender maps

The location where you store a IBM Transformation Extender map determines which services can use the map.

You can store a compiled IBM Transformation Extender map:

- In the Sterling B2B Integrator map repository
A map that is stored in the Sterling B2B Integrator map repository is available to the WTX Map service and to all Sterling B2B Integrator services that are enabled for IBM Transformation Extender.
- In a file system on the computer where Sterling B2B Integrator is installed
The WTX Map service can load and run maps from the file system.

- In a Sterling B2B Integrator document object that is to be loaded dynamically at runtime
Another service in the business process can pass a map in the form of a document to the WTX Map service.
- **Deploying to the map repository**
You can store a compiled IBM Transformation Extender map in Sterling B2B Integrator by using IBM Transformation Extender Design Studio or by using Sterling B2B Integrator. Any Sterling B2B Integrator service that is enabled for IBM Transformation Extender can browse for a IBM Transformation Extender map that is stored in the repository.
- **Deploying to a file system**
You can manually transfer a compiled IBM Transformation Extender map to a file system on the Sterling B2B Integrator server. Ensure that the map is compiled for the correct platform, and then use FTP to transfer the map in binary mode. You must manually check in any required schema files by using the Sterling B2B Integrator dashboard.

Deploying to the map repository

You can store a compiled IBM Transformation Extender map in Sterling B2B Integrator by using IBM Transformation Extender Design Studio or by using Sterling B2B Integrator. Any Sterling B2B Integrator service that is enabled for IBM Transformation Extender can browse for a IBM Transformation Extender map that is stored in the repository.

You must compile a source IBM Transformation Extender map before you can store the map in Sterling B2B Integrator.

Before you can use Design Studio to check a map into the map repository, you must:

- Install IBM Transformation Extender for Integration Servers on the computer where Design Studio is installed.
- Use Design Studio to configure the connection to the Sterling B2B Integrator server. See the IBM Transformation Extender Map Designer documentation for details.

To use Design Studio to store a compiled IBM Transformation Extender map in Sterling B2B Integrator, right-click the map in the Design Studio navigator window and click Deploy Map to Sterling B2B Integrator.

To use the Sterling B2B Integrator dashboard to store a IBM Transformation Extender map in Sterling B2B Integrator:

1. In the Administration Menu, click Deployment > Maps.
2. Select Check in new map from Map Editor.

See the description of checking in maps in the [Sterling B2B Integrator documentation](#) for the procedure to check in maps by using the Sterling B2B Integrator Map Editor.

Deploying to a file system

You can manually transfer a compiled IBM Transformation Extender map to a file system on the Sterling B2B Integrator server. Ensure that the map is compiled for the correct platform, and then use FTP to transfer the map in binary mode. You must manually check in any required schema files by using the Sterling B2B Integrator dashboard.

For details about using the Sterling B2B Integrator dashboard, see the [Sterling B2B Integrator documentation](#).

Testing IBM Transformation Extender maps

You can use IBM Transformation Extender Design Studio to manually test the IBM Transformation Extender map on the Sterling B2B Integrator server.

Before you can use Design Studio to test a map, you must:

- Install IBM Transformation Extender for Integration Servers on the computer where Design Studio is installed.
- Use Design Studio to configure the connection to the Sterling B2B Integrator server. See the IBM Transformation Extender Map Designer documentation for details.

To run a map in Design Studio, right-click the map and click Run Map on Sterling B2B Integrator. See the IBM Transformation Extender Map Designer documentation for details.

Running a business process

A business process can run on a predefined schedule, in response to an activity, or manually.

Each step in a business process performs one of these activities:

- A service runs within the system.
- An adapter calls a third-party application to perform activities outside of the system.
- Sterling B2B Integrator performs the BPML activities that are configured in the business process.

For example, the BPML might start or stop the process flow, assign a specified value in the data, or run specified activities simultaneously.

A map runs when the service that invokes it is triggered by the business process.

See the [Sterling B2B Integrator documentation](#) for details about defining business process schedules, bootstrapping, and manually running a business process.

- **Running multiple concurrent instances of maps**

When you run multiple concurrent instances of maps, enable map caching and set unique work file names and audit file names for maps that perform I/O.

- [Running maps locally or in a separate JVM](#)

You can configure a service to run in a separate JVM based on the size of the input data. You also can create a service configuration to use the WTXRMIAAdapter and configure an adapter container to run the service in a separate JVM. Services that run in separate JVMs are isolated from certain failures, which improves the availability of the service.

Running multiple concurrent instances of maps

When you run multiple concurrent instances of maps, enable map caching and set unique work file names and audit file names for maps that perform I/O.

When multiple instances of a map are running in multiple services or multiple business processes, traces are recorded in the designated file for the first business process thread that runs, but other business processes and other threads fail to trace. The map does not fail or cause an exception.

For optimal performance, configure the map settings to use unique work files and audit files as follows:

- When the Map Settings enable map audit, set MapAudit.>.AuditLocation to File and set the audit FileName to Unique.
- In the Map Settings set WorkSpace to File and set WorkFilePrefix to Unique.
- When the Edit Input Card or Edit Output Card settings enable backup for a card, set Backup.>.BackupLocation.>.FileName to Unique. The map does not fail, and it does not throw an exception.

See the IBM Transformation Extender Map Designer documentation for more information about running multiple instances of a map.

Related reference

- [Map caching](#)
- [Map trace, audit log, and work space directories](#)
- [Performance tuning](#)

Running maps locally or in a separate JVM

You can configure a service to run in a separate JVM based on the size of the input data. You also can create a service configuration to use the WTXRMIAAdapter and configure an adapter container to run the service in a separate JVM. Services that run in separate JVMs are isolated from certain failures, which improves the availability of the service.

- [Based on input-data-size threshold](#)

The Threshold property and the size of the input file determine whether all map instances run locally within the same Java™ Virtual Machine (JVM) or each instance of a map service runs in a separate JVM. When the input data exceeds the input-data-size threshold, each map instance runs in its own JVM and accesses the IBM Transformation Extender Translation Engine through the Sterling RMI adapter.

- [Using service configurations and an adapter container JVM](#)

A *service configuration* is a set of saved configuration properties for a specific type of service, such as the WTX Map service. With Sterling B2B Integrator, you can create and manage multiple service configurations. You can configure a service in a business process to use a particular service configuration. You can create an *adapter container JVM* to run services and adapters in a separate Java virtual machine (JVM).

- [Assigning the WTX Map service to a container node](#)

You can enable the WTX Map service to run in a separate Java Virtual Machine (JVM) when input data exceeds a maximum input-data-size threshold. You create a Sterling B2B Integrator adapter container node and configure the WTX Map service properties to run on the node.

Related reference

- [Performance tuning](#)

Based on input-data-size threshold

The Threshold property and the size of the input file determine whether all map instances run locally within the same Java™ Virtual Machine (JVM) or each instance of a map service runs in a separate JVM. When the input data exceeds the input-data-size threshold, each map instance runs in its own JVM and accesses the IBM Transformation Extender Translation Engine through the Sterling RMI adapter.

You configure the default input-data-size threshold on the Threshold= property in the *Sterling_installation_dir\properties\translator.properties_wtx_ext.in* file. The Threshold value is measured in bytes of data. The default threshold is 0, meaning that all maps run in the same JVM.

Related reference

- [WTX Map service BPML](#)

Using service configurations and an adapter container JVM

A *service configuration* is a set of saved configuration properties for a specific type of service, such as the WTX Map service. With Sterling B2B Integrator, you can create and manage multiple service configurations. You can configure a service in a business process to use a particular service configuration. You can create an *adapter container JVM* to run services and adapters in a separate Java™ virtual machine (JVM).

These steps present an overview of the procedures to create a service configuration and adapter container. See the [Sterling B2B Integrator documentation](#) for details about the procedures.

1. Create and enable one or more service configurations. See the [Sterling B2B Integrator services and adapters documentation](#).
2. Open a business process in the Graphical Process Modeler (GPM) and specify which configuration you want a service to use to run a IBM Transformation Extender map:
 - a. Right-click the service and click Properties.
 - b. Click Advanced.
 - c. Click Add to create a new row in the table.
 - d. In the Name column, enter WTXRMIAAdapter.
 - e. In the Value column, specify the name of the service configuration that you want the service to run.
 - f. Click OK.
3. Set up and start the adapter container JVM. See the [Sterling B2B Integrator services and adapters documentation](#).

Assigning the WTX Map service to a container node

You can enable the WTX Map service to run in a separate Java Virtual Machine (JVM) when input data exceeds a maximum input-data-size threshold. You create a Sterling B2B Integrator adapter container node and configure the WTX Map service properties to run on the node.

1. Configure the adapter container JVM node as described in the Sterling B2B Integrator services and adapters documentation. Run the Sterling B2B Integrator `setupContainer.cmd containerNumber` command, where `containerNumber` is the adapter container node number.
For example, run `setupContainer.cmd 1` to create a container node named `Node1AC1`. See the [Sterling B2B Integrator documentation](#) for additional details.
2. Set the Threshold property in the `Sterling_installation_dir\properties\translator.properties_wtx_ext.in` file.
3. Run the Sterling B2B Integrator `setupFiles.cmd` command.
4. Run the Sterling B2B Integrator `startWindowsService.cmd` command to start the JVM instance.
This command brings up the node and all containers on the node.
5. Edit the WTX Map service and set the environment to the node name, for example, `Node1AC1`:
 - a. Log into the Sterling B2B Integrator dashboard as an administrator.
 - b. Go to Deployment>Services>Configuration and search for the WTX Map service.
 - c. Click Edit. The first page contains the option to select a node. For example, select `Node1AC1` to specify container 1 on node 1.
 - d. Save the changes.

The WTX Map service translates any file that exceeds the input-data-size threshold value against the specified container node, for example, `Node1AC1`. When the file size is less than the threshold value, or when `Threshold=0`, the file is not translated on the container.

Configuring the Translation service to use IBM Transformation Extender maps

When the Sterling B2B Integrator Translation service uses a IBM Transformation Extender map, set the `exhaust_input` setting to YES to enable the service to process all of the input records.

When the `exhaust_input` setting is set to YES, the Sterling B2B Integrator Translation service consumes all of the input records during translation. When the `exhaust_input` parameter is set to NO, the IBM Transformation Extender map processes the entire input, but it returns the output from only the first burst of data. Other processing still occurs, such as calls to the data harness to track duplicate records.

To simulate the `exhaust_input=NO` capabilities of the Sterling B2B Integrator maps, you can define the input type tree on the IBM Transformation Extender map to process only the first block of input data and ignore the remaining input data.

Note: When the Translation service runs a IBM Transformation Extender map, only the first input card and the last output card are associated with the primary document. For more information about the `exhaust_input` setting, see the description of the Translation Service in the [Sterling B2B Integrator documentation](#).

Problem determination

You can enable trace and audit logs to diagnose WTX Map service errors. You can configure the WTX Map service to extract data from a document or a process for tracking purposes.

- **WTX Map service correlations**
A *correlation* is a name/value pair that is extracted from a document or a business process for the purpose of tracking the document or process. The WTX Map service collects a static set of well-known correlations.
- **Diagnostic tools**
Both IBM Transformation Extender and Sterling B2B Integrator provide tools to diagnose an instance of a WTX Map service.

WTX Map service correlations

A *correlation* is a name/value pair that is extracted from a document or a business process for the purpose of tracking the document or process. The WTX Map service collects a static set of well-known correlations.

Correlation data is stored in a Sterling B2B Integrator correlation table. You can use the Sterling B2B Integrator Correlation Search to locate a document or business process by using the data you extracted from it, such as a purchase-order number.

The WTX Map service creates the following correlations with a scope of Translation on the input document:

Map
 Map name
MapVersion
 Version of the map.

The WTX Map service creates the following correlations with a scope of Translation on the output document:

Map
 Map name
MapVersion
 Version of the map
TranslationDateTime
 Current date and time

The well-known correlations are saved in the correlation table, along with any custom correlations that you set by using the data harness function on the map rules or component rules.

See the [Sterling B2B Integrator documentation](#) for details about configuring and searching for correlations.

Diagnostic tools

Both IBM Transformation Extender and Sterling B2B Integrator provide tools to diagnose an instance of a WTX Map service.

You can use these mechanisms for problem determination:

- Sterling B2B Integrator service translation reports
- IBM Transformation Extender trace and audit files
- Sterling B2B Integrator log files
- Sterling B2B Integrator exception handling

You configure map trace and audit files by using the map settings of the WTX Map service.

To enable Sterling B2B Integrator logging and tracing, see the description of setting up event logging in the [Sterling B2B Integrator documentation](#).

- [**Sterling B2B Integrator service translation report**](#)
If a Sterling B2B Integrator service runs a IBM Transformation Extender map and the map fails, the Sterling B2B Integrator translation report contains the results of the map run.
- [**Configuring IBM Transformation Extender logging properties in Sterling B2B Integrator**](#)
In Sterling B2B Integrator, a log file name, location, logging level, and other characteristics are controlled by properties files. You configure IBM Transformation Extender logging by overriding the properties in the log.properties_wtx_ext file with customized settings that you configure in the customer_overrides.properties file.
- [**IBM Transformation Extender logging properties**](#)
The `Sterling_install_dir\properties\log.properties_wtx_ext.in` file defines the properties for IBM Transformation Extender map logs. You can customize the properties of IBM Transformation Extender logs on Sterling B2B Integrator by adding the properties to the `Sterling_install_dir\properties\customer_overrides.properties` file.
- [**Map trace, audit log, and work space directories**](#)
The location of the work space, trace file, and audit file for a map depends on the map settings and the location from which the map is loaded.

Sterling B2B Integrator service translation report

If a Sterling B2B Integrator service runs a IBM Transformation Extender map and the map fails, the Sterling B2B Integrator translation report contains the results of the map run.

Use the Sterling B2B Integrator Administration menu to configure the translation report, such as whether to email the report or save it to a file system. See the [Sterling B2B Integrator documentation](#) for more information.

For each call to the translation service of the map, the translation report includes:

- Translation object name
- Translation service start time
- Translation service end time
- Total map execution time in milliseconds
- IBM Transformation Extender map status code and status message
- Audit data, if enabled in a IBM Transformation Extender map

Unlike other Sterling B2B Integrator services, the WTX Map service does not produce a translation report. For audit data from a IBM Transformation Extender map to be written to the translation report, the map must have audit enabled, the AuditLocation must be `Memory`, and the appropriate data audit settings must be enabled.

When the map enables auditing and the AuditLocation is `File`, the map execution produces an audit log on the file system. The audit log contains additional details.

Configuring IBM Transformation Extender logging properties in Sterling B2B Integrator

In Sterling B2B Integrator, a log file name, location, logging level, and other characteristics are controlled by properties files. You configure IBM Transformation Extender logging by overriding the properties in the `log.properties_wtx_ext` file with customized settings that you configure in the `customer_overrides.properties` file.

In Sterling B2B Integrator, properties files are stored in the `Sterling_install_dir\properties` directory. The `log.properties_wtx_ext.in` file initializes the default properties of the `log.properties_wtx_ext` file. At runtime, the `Sterling_install_dir\log.properties_wtx_ext` file controls IBM Transformation Extender logs that are created in the Sterling B2B Integrator log directory. The `\properties\customer_overrides.properties` file overrides the settings in any properties file when Sterling B2B Integrator is installed or when you run the setup script. To customize the properties of IBM Transformation Extender map logs, you add IBM Transformation Extender logging properties to the `customer_overrides.properties` file.

IBM Transformation Extender logging properties

The `Sterling_install_dir\properties\log.properties_wtx_ext.in` file defines the properties for IBM Transformation Extender map logs. You can customize the properties of IBM Transformation Extender logs on Sterling B2B Integrator by adding the properties to the `Sterling_install_dir\properties\customer_overrides.properties` file.

At runtime, the properties that are configured in the `customer_overrides.properties` file override the default properties defined in the `Sterling_install_dir\properties\log.properties_wtx_ext.in` file. You can override any default property except the `wtxlogger.logkey` property and the `wtxlogger.displayname` property.

Map trace, audit log, and work space directories

The location of the work space, trace file, and audit file for a map depends on the map settings and the location from which the map is loaded.

When the map settings specify that trace, audit or work files are written to the map directory:

- When the map is stored in a file system on the Sterling B2B Integrator server, files are written to the directory where the map is stored.
- When the map is stored in the Sterling B2B Integrator map repository, files are written to the Sterling B2B Integrator `Sterling_install_dir\logs\` directory.
- When a compiled map is dynamically loaded from a document object, files are written to the Sterling B2B Integrator `Sterling_install_dir\logs\` directory.

When the map settings specify that trace, audit or work files are written to a custom directory, they are written to the specified location.

When the map settings specify that the audit data is sent to memory, the audit data is written to the translation status report and instance data of the WTX Map service.

Related reference

- [Running multiple concurrent instances of maps](#)

Performance tuning

The `tx_install_dirconfig.yaml` file contains the `runtime/stream/MaxMemLimit` property and the `runtime/stream/MaxMemDirectory` property. When data passed to or from an adapter exceeds the size (in MB) that is configured on the `StreamMaxMemLimit` property, the data is paged to a temporary file to limit the memory consumption of the process. The `StreamMaxMemDirectory` property specifies the directory where the temporary files are located.

Configure these properties in conjunction with the threshold for running maps in separate Java™ Virtual Machines (JVMs).

Related reference

- [Running maps locally or in a separate JVM](#)
- [Map caching](#)
- [Running multiple concurrent instances of maps](#)

IBM Transformation Extender for IBM Integration Bus

This documentation provides information for IBM® Transformation Extender for IBM Integration Bus.

New users: how to find the information you want

The Start here topic contains other topics that provide an introduction to IBM Transformation Extender for IBM Integration Bus for new users.

Experienced users: how to find the information you want

If you are familiar with what information is available, and where to find it in this documentation, you might have a specific area of information to which you need to reference. Navigate through the information in the table of contents, or take a more direct route to find specific information by using the Search, or Index functionality.

- [Start here](#)
New users: use these topics to help you navigate through the IBM Transformation Extender for IBM Integration Bus documentation:
- [Product overview](#)
These topics provide introductory information to help you get started with IBM Transformation Extender for IBM Integration Bus.

- [Developing a message flow](#)
These topics provide you with the information you need to develop message flows with the TX Map node component.
- [Deploying your message flow](#)
These topics provide you with the information you need to deploy message flows that contain TX Map nodes.
- [Running your message flow](#)
You can run your message flow by sending one or more messages through the flow.
- [Problem determination](#)
Exceptions occur for map failures, TX Map node data errors, or when messages from both input and output are missing or malformed. IBM Transformation Extender passes error messages to IBM Integration Bus to write to its log and trace files.
- [References](#)
Use the reference information to accomplish the tasks that your business needs.

Start here

New users: use these topics to help you navigate through the IBM® Transformation Extender for IBM Integration Bus documentation:

- If you are new to IBM Transformation Extender for IBM Integration Bus, look at the [Getting started](#) topic first.
- If you are ready to start developing IBM Transformation Extender for IBM Integration Bus applications, look at the [Developing message flow applications](#) topic.

To use IBM Transformation Extender for IBM Integration Bus, you should be familiar with both the IBM Transformation Extender and IBM Integration Bus products. To learn more about them, review the IBM Transformation Extender and IBM Integration Bus documentation.

- [Getting started](#)
IBM Transformation Extender for IBM Integration Bus provides an extension to the IBM Integration Bus capability to transform messages with processing nodes, with an additional node that contains the IBM Transformation Extender engine.
- [Developing message flow applications](#)
Use these topics to help you start developing your own IBM Transformation Extender for IBM Integration Bus message flow applications.
- [How can I diagnose problems?](#)
New users: use this topic to find out how to debug message flow applications, and get help with troubleshooting problems.

Related reference

- [Developing message flow applications](#)

Getting started

IBM® Transformation Extender for IBM Integration Bus provides an extension to the IBM Integration Bus capability to transform messages with processing nodes, with an additional node that contains the IBM Transformation Extender engine.

To help you get started with learning more about the IBM Transformation Extender for IBM Integration Bus product, see the product overview topics.

Related concepts

- [Product overview](#)

Developing message flow applications

Use these topics to help you start developing your own IBM® Transformation Extender for IBM Integration Bus message flow applications.

- Developing a message flow.
- Deploying your message flow.
- Running your message flow.

References to other topics are also included to help you understand IBM Transformation Extender for IBM Integration Bus concepts that might be useful when you develop your own message flow applications.

Related tasks

- [Running your message flow](#)

Related reference

- [Developing a message flow](#)
- [Deploying your message flow](#)
- [References](#)

How can I diagnose problems?

New users: use this topic to find out how to debug message flow applications, and get help with troubleshooting problems.

To help you diagnose problems when you run your message flow applications, see the problem determination topics.

Related reference

- [Problem determination](#)
-

Product overview

These topics provide introductory information to help you get started with IBM® Transformation Extender for IBM Integration Bus.

- [Introduction](#)
IBM Transformation Extender for IBM Integration Bus is one of the extenders packaged as IBM Transformation Extender for Integration Servers.
 - [Product uses](#)
IBM Transformation Extender for IBM Integration Bus extends the functionality of IBM Integration Bus so that when you use IBM Integration Bus, you have access to IBM Transformation Extender's universal transformation capabilities through maps and type trees.
 - [Transforming data](#)
A message sender and receiver might have different message format specifications, requiring you to design your message flow to transform the data in the message before delivering it. You might need to design your message flow to reformat your message, enrich your message, or do numeric and string calculations based on the value of certain fields.
 - [TX Map node overview](#)
The TX Map node is a component of the IBM Transformation Extender for IBM Integration Bus.
-

Introduction

IBM® Transformation Extender for IBM Integration Bus is one of the extenders packaged as IBM Transformation Extender for Integration Servers.

Use this extender to trigger and run IBM Transformation Extender transformation maps from within a IBM Integration Bus message flow. Use IBM Transformation Extender for IBM Integration Bus to transform and distribute different messages across your various systems and applications, because this product enhances the enterprise service bus features of IBM Integration Bus with the universal transformation capability of IBM Transformation Extender.

IBM Transformation Extender for IBM Integration Bus complements the native message transformation and message parsing capabilities of IBM Integration Bus. With IBM Integration Bus, not only can you use the set of parsers and transformation options that are ready for you to use as is, but you can also use custom parsers and transformations through a plug-in facility. IBM Transformation Extender for IBM Integration Bus includes a plug-in node, called the TX Map node, for running IBM Transformation Extender maps in your message flows.

For information about IBM Integration Bus, see the [IBM Integration Bus library page](#).

Related concepts

- [TX Map node overview](#)
-

Product uses

IBM® Transformation Extender for IBM Integration Bus extends the functionality of IBM Integration Bus so that when you use IBM Integration Bus, you have access to IBM Transformation Extender's universal transformation capabilities through maps and type trees.

Also, IBM Transformation Extender for IBM Integration Bus enhances the functionality of IBM Transformation Extender so that when you use IBM Transformation Extender, you have access to IBM Integration Bus Enterprise Service Bus capabilities.

- [For IBM Integration Bus users](#)
If you already use IBM Integration Bus, you can develop message flows that use IBM Transformation Extender maps and type trees to complement IBM Integration Bus built-in transformation functionality.
 - [For IBM Transformation Extender users](#)
If you already use IBM Transformation Extender, you can develop message flows that use IBM Integration Bus Enterprise Service Bus functionality to complement IBM Transformation Extender maps and type trees.
-

For IBM Integration Bus users

If you already use IBM® Integration Bus, you can develop message flows that use IBM Transformation Extender maps and type trees to complement IBM Integration Bus built-in transformation functionality.

As a IBM Integration Bus user, use IBM Transformation Extender maps and type trees in the following ways:

- Use IBM Transformation Extender maps if you need transformation features that are not available with native ESQL, Java, Mapping, and XSL transformations such as:
 - Pulling data from sources such as files and queues within a map.
 - Calling external programs and services from within a map.
 - Splitting and routing batches of messages.

- Use IBM Transformation Extender type trees if you need message modeling features that are not available with native message sets such as:
 - Modeling complex binary or text message formats that the MRM is unable to model.
 - Complex validation without having to write code, using component rules on type trees.
 - Use IBM Transformation Extender Industry Packs to integrate a range of industry standard data formats with your enterprise infrastructure through predefined type tree templates and maps that the packs provide.
-

For IBM Transformation Extender users

If you already use IBM® Transformation Extender, you can develop message flows that use IBM Integration Bus Enterprise Service Bus functionality to complement IBM Transformation Extender maps and type trees.

As a IBM Transformation Extender user, use IBM Integration Bus built-in functionality in the following ways:

- Use existing IBM Transformation Extender skills.
 - Use a single transformation technology and skill set across your entire organization.
- Reuse existing IBM Transformation Extender maps and type trees within an Enterprise Service Bus environment.
- Combine IBM Transformation Extender maps with IBM Integration Bus facilities, such as:
 - Message routing, including publish/subscribe applications.
 - Event-driven processing, such as message aggregation and fan-in flows.
 - Complex event processing, which can use configurable patterns of event sequences to trigger maps to run.
 - Integration with Web Services through IBM Integration Bus's support for SOAP, WS-Addressing and WS-Security, and WebSphere® Service Registry and Repository.
 - An integrated environment to move transactional data throughout your organization.
 - Implementation of a complete application integration solution.
 - Enterprise Service Bus-enabled adapters and services.
 - Integrated runtime security model.
 - Built-in accounting, statistics, and monitoring functionality.
- Run maps in a robust, scalable and highly available environment.
 - Use for your applications that have high-availability or failover requirements.
 - Use IBM Integration Bus clustering.

Transforming data

A message sender and receiver might have different message format specifications, requiring you to design your message flow to transform the data in the message before delivering it. You might need to design your message flow to reformat your message, enrich your message, or do numeric and string calculations based on the value of certain fields.

IBM® Integration Bus provides the following four built-in transformation nodes that are ready for you to use as is, in your message flows:

- Compute
- XSLTransform
- Mapping
- JavaCompute

IBM Transformation Extender for IBM Integration Bus extends these data transformation and validation capabilities by enabling you to use these IBM Transformation Extender maps and type trees within the broker environment. You can create IBM Transformation Extender type trees that model complex data, complete with component rules that do syntax and semantic validation of the data in your messages. These type trees can be used in maps, which might have multiple input and output cards, and reused in different message flows, as well as in other applications across your enterprise besides IBM Integration Bus.

For information about the IBM Integration Bus transformation nodes, see the related topics in the IBM Integration Bus product documentation.

Related concepts

- [TX Map node overview](#)
-

TX Map node overview

The TX Map node is a component of the IBM® Transformation Extender for IBM Integration Bus.

Within a IBM Integration Bus message flow, a series of steps is used to process a message. A message flow node performs each step. The TX Map node is a message flow node that you use in message flows. It runs a IBM Transformation Extender map within a message flow. The TX Map node uses one or more broker message trees as its input, runs a IBM Transformation Extender map that you specify, and creates one or more broker message trees as its output.

The map that the TX Map node runs can either be a compiled map that is predeployed to the broker, or a source map that is compiled when the message flow is added to a broker archive (.bar) file and deployed as part of broker archive deployment. The map can have multiple input cards, but the TX Map node only has a single input terminal.

If the message tree that arrives at the input terminal consists of a single message, select which input card receives the message. If the message tree that arrives at the input terminal is a message collection, containing multiple messages, each message in the collection is delivered to the appropriate input card. Then the TX Map node runs the map.

Any cards that did not receive a message obtains data from the adapter that is specified on them. The map can have multiple output cards. Select which output cards to connect to other nodes in the message flow. If the output terminal for an output card is connected, a message tree is built, and propagated down the message flow. If the output terminal for an output card is not connected, the data is sent to the adapter that is specified on the card.

IBM Integration Bus processes many different classes of message, such as fixed-length binary, delimited text, and XML. Each class is known as a message domain. The TX Map node can handle input and output messages belonging to several different message domains. See [Supported message domains](#).

The TX Map node is contained in the IBM Transformation Extender for IBM Integration Bus drawer of the IBM Integration Toolkit message flow node palette. Use the message flow node palette to add TX Map nodes in the message flows you are creating in the Message Flow editor. Drag node objects from the palette onto the IBM Integration Toolkit canvas, and create connections, also called wires, between two nodes.

The TX Map node is not an input node, and can not be used to trigger the execution of the message flow.

Related concepts

- [Supported message domains](#)

Related tasks

- [BLOB domain](#)
- [XMLNSC domain](#)
- [DataObject domain](#)
- [MRM domain](#)

Related reference

- [Using the TX Map node within a message flow](#)
- [References](#)

Developing a message flow

These topics provide you with the information you need to develop message flows with the TX Map node component.

- [Using the TX Map node within a message flow](#)

When you develop a message flow, you can set up a TX Map node that runs a map that has a single input card or multiple input cards.

Related concepts

- [TX Map node overview](#)

Related tasks

- [Deploying IBM Transformation Extender maps](#)

Related reference

- [Deploying your message flow](#)

Using the TX Map node within a message flow

When you develop a message flow, you can set up a TX Map node that runs a map that has a single input card or multiple input cards.

The activities for using a TX Map node within a message flow include:

- Creating a map.
- Creating a message flow that uses the TX Map node.
- Deploying the message flow.
 - Creating and building a broker archive (bar) file.
 - Deploying the broker archive (bar) file.
 - Deploying the map only if you are using a precompiled map.
- Running your message flow.

After the message flow is deployed, IBM® Integration Bus starts the message flow automatically, and the following actions result:

- The message flow runs when an input node receives an input message bit stream.
 - A parser parses the input bit stream into an input message tree.
 - A TX Map node receives the input message tree and runs the map.
 - The map transforms the data in the input message tree, and creates one or more output message trees.
 - The output message trees can be processed by other nodes in the message flow.
- [Using the TX Map node with a source map and a single input](#)
Develop a message flow with a TX Map node specified with a source map, that, at run time, is triggered to run by a single input.
- [Using the TX Map node with a precompiled map and a single input](#)
Develop a message flow with a TX Map node specified with a precompiled map, that, at run time, is triggered to run by a single input.
- [Using the TX Map node with a source map and multiple inputs](#)
Develop a message flow with a TX Map node specified with a source map, that, at run time, is triggered to run by multiple inputs.

- [Using the TX Map node with a precompiled map and multiple inputs](#)

Develop a message flow with a TX Map node specified with a precompiled map, that, at run time, is triggered to run by multiple inputs.

Related tasks

- [Running your message flow](#)

Using the TX Map node with a source map and a single input

Develop a message flow with a TX Map node specified with a source map, that, at run time, is triggered to run by a single input.

The map could have many input cards. Only one nominated card receives data from the message flow, while each of the other cards pull data directly from their adapter.

The map could have many output cards. Cards that are connected to the message flow propagate data to the flow, while each of the unconnected cards send data directly to their adapter.

To develop a simple message flow that demonstrates this scenario, do the following steps. They are based on the assumption that you used Map Designer to create a source map (.mms) file containing an executable map with one or more input cards, and one or more output cards.

1. Create a new message flow project in the IBM® Integration Development perspective.
2. Create a new message flow in the project. Select Use default broker schema, or provide your own schema name to qualify the message flow name.
3. Drag a TX Map node from the palette onto the canvas for your message flow.
The node initially has no output terminals, and one failure terminal.
4. Navigate to the Properties view.
5. On the Basic tab of the TX Map node, select the [source map](#).
6. On the Input tab of the TX Map node, select the map card that receives the input.
7. Drag an input node, for example, a FileInput node or MQInput node, to the canvas, and connect the out terminal of the input node to the in terminal of the TX Map node.
The message from the input node overrides the adapter of the map's selected input card.
8. Configure the input node with the necessary transport-related properties.
9. On the Input Message Parsing tab of the input node, configure the message template properties to describe the format of the input message. Select the domain from the list of [supported message domains](#).
10. Drag an output node, for example, a FileOutputStream node or MQOutput node, to the canvas, and connect one of the output terminals of the TX Map node to the in terminal of the output node.
Connecting the output terminal overrides the adapter of the corresponding output card.
11. Configure the output node with the necessary transport-related properties.
12. On the Outputs tab of the TX Map node, configure properties to describe the format of the output message propagated by the output card.
 - a. Click Add to add a set of properties.
The Add Properties entry dialog appears.
 - b. Enter values for Card Number, Message Domain, Message Set, and Message Type, and click OK.
Message Domain must be one of the [supported message domains](#).
13. Repeat steps 10 through 12 for each output terminal that you want to connect.
14. Press Ctrl+S to save the message flow.

Manually create and build the broker archive (bar) file, and then deploy it to the broker.

When the message flow runs, the message from the input node is passed to the TX Map node, and the TX Map node runs the map. The message overrides the adapter specified on the selected input card. For example, if the input card uses the File adapter, the map does not read from the file; it uses the message from the message flow. Connected output terminals override the adapters of the corresponding output cards. For example, if an output card uses the File adapter, the map does not write to the file; it propagates the message to the message flow.

Related tasks

- [Using source maps](#)
- [Running your message flow](#)

Related reference

- [Deploying your message flow](#)

Using the TX Map node with a precompiled map and a single input

Develop a message flow with a TX Map node specified with a precompiled map, that, at run time, is triggered to run by a single input.

The map could have many input cards. Only one nominated card receives data from the message flow, while each of the other cards pull data directly from their adapter.

The map could have many output cards. Cards that are connected to the message flow propagate data to the flow, while each of the unconnected cards send data directly to their adapter.

To develop a simple message flow that demonstrates this scenario, do the following steps. They are based on the assumption that you used Map Designer to create a source map (.mms) file, containing an executable map with one or more input cards, and one or more output cards, and that you compiled the map for a target platform, creating a compiled map (.mmc) file.

1. Create a new message flow project in the IBM® Integration Development perspective.
2. Create a new message flow in the project. Select Use default broker schema, or provide your own schema name to qualify the message flow name.

3. Drag a TX Map node, from the palette onto the canvas for your message flow.
The node initially has no output terminals and one failure terminal.
4. Navigate to the Properties view.
5. On the Basic tab of the TX Map node, select the [precompiled map](#).
6. On the Input tab of the TX Map node, select the map card that receives the input.
7. Drag an input node, for example, a FileInput node or MQInput node, to the canvas, and connect the out terminal of the input node to the in terminal of the TX Map node.
The message from the input node overrides the adapter of the map's selected input card.
8. Configure the input node with the necessary transport-related properties.
9. On the Input Message Parsing tab of the input node, configure the message template properties to describe the format of the input message. Select the domain from the list of [supported message domains](#).
10. Drag an output node, for example, a FileOutputStream node or MQOutput node, to the canvas, and connect one of the output terminals of the TX Map node to the in terminal of the output node.
Connecting the output terminal overrides the adapter of the corresponding output card.
11. Configure the output node with the necessary transport-related properties.
12. On the Outputs tab of the TX Map node, configure properties to describe the format of the output message propagated by the output card.
 - a. Click Add to add a set of properties.
The Add Properties entry dialog appears.
 - b. Enter values for Card Number, Message Domain, Message Set, and Message Type, and click OK.
Message Domain must be one of the [supported message domains](#).
13. Repeat steps 10 through 12 for each output terminal that you want to connect.
14. Press Ctrl+S to save the message flow.

Manually create and build the broker archive (bar) file, and then deploy it to the broker. You must also manually deploy the compiled map (.mmc) file to the broker's file system.

When the message flow runs, the message from the input node is passed to the TX Map node, and the TX Map node runs the map. The message overrides the adapter specified on the selected input card. For example, if the input card uses the File adapter, the map does not read from the file; it uses the message from the message flow. Connected output terminals override the adapters of the corresponding output cards. For example, if an output card uses the File adapter, the map does not write to the file; it propagates the message to the message flow.

Related tasks

- [Using precompiled maps](#)
- [Running your message flow](#)

Related reference

- [Deploying your message flow](#)

Using the TX Map node with a source map and multiple inputs

Develop a message flow with a TX Map node specified with a source map, that, at run time, is triggered to run by multiple inputs.

The TX Map node is used with a native Collector node, which gathers the multiple inputs and, when specified event criteria are satisfied, propagates the messages to the TX Map node in the form of a message collection. For information about the Collector node and message collection, see the specific IBM® Integration Bus product documentation.

The map could have many input cards. Any number of cards can receive data from the message collection, while any cards that do not receive data pull data directly from their adapter.

The map could have many output cards. Cards that are connected to the message flow propagate data to the flow, while each of the unconnected cards send data directly to their adapter.

To develop a simple message flow that demonstrates this scenario, do the following steps. They are based on the assumption that you used Map Designer to create a source map (.mms) file containing an executable map with two input cards, and one or more output cards.

1. Create a new message flow project in the IBM Integration Development perspective.
2. Create a new message flow in the project. Select Use default broker schema, or provide your own schema name to qualify the message flow name.
3. Drag a TX Map node from the palette onto the canvas for your message flow.
The node initially has no output terminals and one failure terminal.
4. Navigate to the Properties view.
5. On the Basic tab of the TX Map node, select the [source map](#).
6. Drag a Collector node to the canvas.
7. Drag an input node, for example, a FileInput node or MQInput node, to the canvas.
8. Configure the input node with the necessary transport-related properties.
9. On the Input Message Parsing tab of the input node, configure the message template properties to describe the format of the input message. Select the domain from the list of [supported message domains](#).
10. Right-click the Collector node, and select Add Input Terminal. Enter the name of a new input terminal, which must match the card name of the input card in the map that receives the input message. The new input terminal is added to the Collection Definition table on the Basic tab.
11. On the Basic tab, configure the required event criteria properties for the new input terminal, namely Quantity, Timeout, Correlation Path, and Correlation Pattern.
12. Connect the out terminal of the input node to the new input terminal on the Collector node.
13. Repeat steps 7 through 12 for each different input the map requires from the message flow.
14. On the Basic tab of the Collector node, configure the Collection Name and Collection Expiry properties. In particular, you should specify a Collection Expiry value.
15. Connect the out terminal from the Collector node to the in terminal of the TX Map node.
Messages from the propagated message collection override the adapters of the map's input cards.
16. Connect the expiry terminal from the Collector node to the in terminal of the TX Map node, to handle incomplete message collections.

17. Drag an output node, for example, a FileOutput node or MQOutput node, to the right of the TX Map node, and connect one of the output terminals of the TX Map node to the in terminal of the output node.

Connecting the output terminal overrides the adapter of the corresponding output card.

18. Configure the output node with the necessary transport-related properties.

19. On the Outputs tab of the TX Map node, configure properties to describe the format of the output message propagated by the output card.

a. Click Add to add a set of properties.

The Add Properties entry dialog appears.

b. Enter values for Card Number, Message Domain, Message Set, and Message Type, and click OK.

Message Domain must be one of the [supported message domains](#).

20. Repeat steps 17 through 19 for each output terminal that you want to connect.

21. Press Ctrl+S to save the code.

An Unconnected catch terminals warning message appears, which you can ignore.

Manually create and build the broker archive (bar) file, and then deploy it to the broker.

When the message flow runs, the Collector node gathers messages into message collections. As each collection completes, it is propagated to the TX Map node, and the TX Map node runs the map. Each message in the collection is delivered to the matching input card. For example, if a message in the collection arrived at the invoice Collector node input terminal, it would be delivered to the invoice input card. The messages in the collection override the adapter specified on the input cards. For example, if the input cards use the File adapter, the map does not read from the file; it uses the message from the message collection. Connected output terminals override the adapters of the corresponding output cards. For example, if an output card uses the File adapter, the map does not write to the file; it propagates the message to the message flow.

Related concepts

- [Collector node](#)

Related tasks

- [Using source maps](#)
- [Running your message flow](#)

Related reference

- [Deploying your message flow](#)

Using the TX Map node with a precompiled map and multiple inputs

Develop a message flow with a TX Map node specified with a precompiled map, that, at run time, is triggered to run by multiple inputs.

The TX Map node is used with a native Collector node, which gathers the multiple inputs and, when specified event criteria are satisfied, propagates the messages to the TX Map node in the form of a message collection. For information about the Collector node and message collection, see the specific IBM® Integration Bus product documentation.

The map could have many input cards. Any number of cards can receive data from the message collection, while any cards that do not receive data pull data directly from their adapter.

The map could have many output cards. Cards that are connected to the message flow propagate data to the flow, while each of the unconnected cards send data directly to their adapter.

To develop a simple message flow that demonstrates this scenario, do the following steps. They are based on the assumption that you used Map Designer to create a source map (.mms) file, containing an executable map with one or more input cards, and one or more output cards, and that you compiled the map for a target platform, creating a compiled map (.mmc) file.

1. Create a new message flow project in the IBM Integration Development perspective.
2. Create a new message flow in the project. Select Use default broker schema, or provide your own schema name to qualify the message flow name.
3. Drag a TX Map node from the palette onto the canvas for your message flow.
The node initially has no output terminals and one failure terminal.
4. Navigate to the Properties view.
5. On the Basic tab of the TX Map node, select the [precompiled map](#).
6. Drag a Collector node to the canvas.
7. Drag an input node, for example, a FileInput node or MQInput node, to the canvas.
8. Configure the input node with the necessary transport-related properties.
9. On the Input Message Parsing tab of the input node, configure the message template properties to describe the format of the input message. Select the domain from the list of [supported message domains](#).
10. Right-click the Collector node, and select Add Input Terminal. Enter the name of a new input terminal, which must match the card name of the input card in the map that receives the input message. The new input terminal is added to the Collection Definition table on the Basic tab.
11. On the Basic tab, configure the required event criteria properties for the new input terminal, namely Quantity, Timeout, Correlation Path, and Correlation Pattern.
12. Connect the out terminal of the input node to the new input terminal on the Collector node.
13. Repeat steps 7 through 12 for each different input the map requires from the message flow.
14. On the Basic tab of the Collector node, configure the Collection Name and Collection Expiry properties. In particular, you should specify a Collection Expiry value.
15. Connect the out terminal from the Collector node to the in terminal of the TX Map node.
Messages from the propagated message collection override the adapters of the map's input cards.
16. Connect the expiry terminal from the Collector node to the in terminal of the TX Map node, to handle incomplete message collections.
17. Drag an output node, for example, a FileOutput node or MQOutput node, to the right of the TX Map node, and connect one of the output terminals of the TX Map node to the in terminal of the output node.
Connecting the output terminal overrides the adapter of the corresponding output card.
18. Configure the output node with the necessary transport-related properties.

19. On the Outputs tab of the TX Map node, configure properties to describe the format of the output message propagated by the output card.
 - a. Click Add to add a set of properties.
The Add Properties entry dialog appears.
 - b. Enter values for Card Number, Message Domain, Message Set, and Message Type, and click OK.
Message Domain must be one of the [supported message domains](#).
20. Repeat steps 17 through 19 for each output terminal that you want to connect.
21. Press Ctrl+S to save the code.
An Unconnected catch terminals warning message appears, which you can ignore.

Manually create and build the broker archive (bar) file, and then deploy it to the broker. You must also manually deploy the compiled map (.mmc) file to the broker's file system.

When the message flow runs, the Collector node gathers messages into message collections. As each collection completes, it is propagated to the TX Map node, and the TX Map node runs the map. Each message in the collection is delivered to the matching input card. For example, if a message in the collection arrived at the invoice Collector node input terminal, it would be delivered to the invoice input card. The messages in the collection override the adapter specified on the input cards. For example, if the input cards use the File adapter, the map does not read from the file; it uses the message from the message collection. Connected output terminals override the adapters of the corresponding output cards. For example, if an output card uses the File adapter, the map does not write to the file; it propagates the message to the message flow.

Related concepts

- [Collector node](#)

Related tasks

- [Using precompiled maps](#)
- [Running your message flow](#)

Related reference

- [Deploying your message flow](#)

Deploying your message flow

These topics provide you with the information you need to deploy message flows that contain TX Map nodes.

Message flows are hosted in execution groups. An execution group is a process. On System z environments, it is an address space. A broker can have one or more execution groups. Message flows are deployed to execution groups in broker archive (.bar) files.

When you deploy your message flow, select the execution group into which the broker archive (bar) file is deployed.

After you deploy your message flow, it is started automatically.

For details about IBM® Integration Bus concepts, see the IBM Integration Bus documentation.

- [Deploying IBM Transformation Extender maps](#)

When you deploy message flows that contain TX Map nodes, you must also deploy the associated IBM Transformation Extender maps, and any files that the maps require.

Deploying IBM Transformation Extender maps

When you deploy message flows that contain TX Map nodes, you must also deploy the associated IBM® Transformation Extender maps, and any files that the maps require.

There are two alternative map deployment mechanisms when you use the TX Map node in a message flow.

For all deployments:

1. Create a broker archive (bar) file.
This opens the bar file editor.
2. Switch to the Prepare tab, and check the box against your message flow.
3. Click Build broker archive to build the bar file, confirm that the build operation was successful, and click Save to save the updated bar file.
After you click Build broker archive, you might be prompted to respond in the Override configurable properties dialog as to whether or not you want to continue to allow the build process to override the configurable properties in the bar file.
4. If you need to set any message flow configurable properties, switch to the Manage tab, and set the properties accordingly in the Configure tab of the Properties view.

The steps that you do next, depend on the kind of map you are using:

- [source](#)
- [precompiled](#)
- [Using a source map with a TX Map node](#)
- [Using a precompiled map with a TX Map node](#)

Related tasks

- [Using the TX Map node with a source map and a single input](#)
 - [Using the TX Map node with a precompiled map and a single input](#)
 - [Using the TX Map node with a source map and multiple inputs](#)
 - [Using the TX Map node with a precompiled map and multiple inputs](#)
-

Using a source map with a TX Map node

To use a source map with a TX Map node:

1. In the Prepare tab, click Build broker archive.
When IBM® Integration Bus builds the bar file, it compiles the message flow, and adds it to the bar file. As part of this compilation, the source map is compiled automatically, and the resulting compiled maps, and [other resources](#) in the project, are added to a map archive (mar) file, which is then added to the bar file. When a source map is compiled, IBM Integration Bus adds three compiled maps to the map archive (mar) file, one each for Windows, UNIX, and z/OS operating systems. This means that the resulting bar file can be deployed to any supported IBM Integration Bus runtime environment.
2. When the bar file is complete, deploy it using one of the typical IBM Integration Bus deployment techniques.
Deployment unpacks the bar file and forwards the content to the broker runtime environment. Any mar files are placed into locations from where they are loaded when the TX Map node requires them.

Related tasks

- [Deploying map resources](#)
-

Using a precompiled map with a TX Map node

To use a precompiled map with a TX Map node:

1. In the Prepare tab, click Build broker archive.
When the IBM® Integration Bus builds the bar file, it compiles the message flow. The compiled map does not get deployed in the bar file.
2. Compile the map for the IBM Integration Bus runtime environment that you want.
3. Manually deploy the map, and [other required resources](#), to the runtime directory specified by the Map Server location property of the TX Map node.
4. When the bar file is complete, deploy it using one of the typical IBM Integration Bus deployment techniques.
Deployment unpacks the bar file and forwards the content to the broker runtime environment.

Related tasks

- [Deploying map resources](#)
-

Running your message flow

You can run your message flow by sending one or more messages through the flow.

The message flow receives a message in an input node. The input node parses the message bit stream using the specified parser, creating a broker message tree. The resulting message tree is propagated down the flow. When a TX Map node receives a propagated message tree, it runs the associated map and creates output messages.

If your map is expecting multiple inputs, use a Collector node to gather those inputs. In this case, you must send as many messages as are needed to complete the message collection that the Collector node defined.

The body of a message tree, propagated from an output terminal of a TX Map node, is owned by the message domain that you specified on the output terminal. Any headers present in the input message tree that the map received, such as MQMD, will be preserved into the propagated message tree, unchanged. If the map received a message collection as its input, the propagated headers will be those of the first message in the collection. Headers from other messages in the collection are discarded. The Environment, Local Environment and Exception List trees from the input message are also propagated, unchanged.

If a failure occurs when running a map, the failure is added to the Exception List, and the input message, Environment, Local Environment and Exception List are propagated from the failure terminal, if connected. Otherwise, an exception is thrown.

When the message tree reaches an output node, it is serialized into a bit stream by the owning parser.

You can override the name of the map to be run by a TX Map node, dynamically at run time, using the message flow's LocalEnvironment. See [Overriding map properties at run time](#).

To run your message flow:

1. If your message flow has a TX Map node with a single input:
 - a. Put a message on the queue.
The TX Map node passes the message to the map, and the map writes it to the output.
2. If your message flow has a Collector node:
 - a. Put a message on the first queue.
 - b. Put a message on the second queue.
The Collector node passes both messages to the map, and the map writes them to the output.

Problem determination

Exceptions occur for map failures, TX Map node data errors, or when messages from both input and output are missing or malformed. IBM® Transformation Extender passes error messages to IBM Integration Bus to write to its log and trace files.

- [IBM Transformation Extender map failures](#)

When IBM Integration Bus runs your message flow that uses the TX Map node to run your map, and the map fails, there are tools that you can use to troubleshoot the problem.

- [TX Map node exceptions](#)

When IBM Integration Bus runs your message flow that uses the TX Map node, and it fails, there are tools that you can use to troubleshoot the problem.

- [Using IBM Integration Bus log and trace files](#)

When the TX Map node detects a problem and raises an exception, it sends information to event log and trace files through the IBM Integration Bus logging and tracing mechanisms.

- [Sending audit data to output and failure terminals through the local environment tree](#)

You can configure a IBM Transformation Extender map to send audit-log data across IBM Transformation Extender message flow wires by using the IBM Integration Bus local environment (LocalEnvironment) tree. When the IBM Integration Bus message flow runs successfully, the message flow sends the audit-log data to the output terminal of the IBM Integration Bus node. When the IBM Integration Bus message flow fails, the audit log data is sent to the failure terminal of the IBM Integration Bus node (if a failure terminal is wired to the map node).

Related reference

- [Using IBM Integration Bus log and trace files](#)

IBM Transformation Extender map failures

When IBM® Integration Bus runs your message flow that uses the TX Map node to run your map, and the map fails, there are tools that you can use to troubleshoot the problem.

As a result of running a map, if the audit log map settings are set to ON, IBM Transformation Extender writes information about the map run, including the map return codes and messages, to the log file. For the list of all the return codes and messages, see the topic about error and warning messages that is in the IBM Transformation Extender Map Designer documentation.

If the trace map settings are set to ON, IBM Transformation Extender writes the map run steps to the trace file. For information about the trace file, see the specific topics in the IBM Transformation Extender Map Designer documentation.

All log and trace entries include time stamps.

By default, IBM Transformation Extender generates unique log and trace file names at run time, equivalent to when you use the Unique setting for the log and trace file names. You can also select the Custom setting to specify a custom file name and location.

If you specified Use external map in the TX Map node, IBM Transformation Extender creates the log and trace files with the same names and in the same directory locations as specified in the external map.

If you specified Use map from project in the TX Map node, and selected Default for the audit log or trace file name map setting, IBM Transformation Extender creates the log file as *map_name.log*, and the trace file as *map_name.mtr*, where *map_name* is the name of the executable map. If you selected Default for Directory, which specifies the audit log or trace file location, IBM Transformation Extender creates the files in the IBM Integration Bus's deployment directory.

Use the information that is in the IBM Transformation Extender log and trace files to help you troubleshoot the problem.

For information about the log and trace files, and troubleshooting maps, see the IBM Transformation Extender Map Designer documentation.

Related reference

- [TX Map node exceptions](#)

TX Map node exceptions

When IBM® Integration Bus runs your message flow that uses the TX Map node, and it fails, there are tools that you can use to troubleshoot the problem.

If any of the data that the TX Map node receives from an input message is missing or malformed, the node raises an exception.

If a Map Node failure terminal is connected to the message flow, IBM Transformation Extender for IBM Integration Bus propagates the following information through the failure terminal when the Map node fails:

- A message that contains NodeErrorInfo
- The exception list

In addition, IBM Integration Bus sends the original input message through the failure terminal as a separate message.

If the terminal is not connected, the exception is thrown back up the message flow, according to standard IBM Integration Bus failure behavior.

If the TX Map node fails, IBM Integration Bus logs the failure exception, including the standard error message, in the broker's event log and, optionally, trace files. For details about the standard error message, see the topic about problem determination.

Use the information that is in the IBM Integration Bus log and trace files to help you troubleshoot the problem.

Related reference

- [Using IBM Integration Bus log and trace files](#)
 - [IBM Transformation Extender map failures](#)
-

Using IBM Integration Bus log and trace files

When the TX Map node detects a problem and raises an exception, it sends information to event log and trace files through the IBM® Integration Bus logging and tracing mechanisms.

It sends information to local error log files through the IBM Integration Bus logging mechanism, which is the Event Viewer on Windows, the system log on UNIX, and the console log on z/OS® operating systems. Logging information includes event status, and node return code, which is either the map return code or the node failure code.

If you activated the user trace options, IBM Integration Bus sends information to trace files through the IBM Integration Bus tracing mechanism. Tracing information includes more information than the event log files provide, such as formatted messages.

All log and trace entries include time stamps.

Use the information that is in these IBM Integration Bus log and trace files to help you troubleshoot the problem.

For information about the IBM Integration Bus logging and tracing mechanisms, see the specific IBM Integration Bus product documentation.

Related reference

- [Problem determination](#)
 - [TX Map node exceptions](#)
-

Sending audit data to output and failure terminals through the local environment tree

You can configure a IBM® Transformation Extender map to send audit-log data across IBM Transformation Extender message flow wires by using the IBM Integration Bus local environment (LocalEnvironment) tree. When the IBM Integration Bus message flow runs successfully, the message flow sends the audit-log data to the output terminal of the IBM Integration Bus node. When the IBM Integration Bus message flow fails, the audit log data is sent to the failure terminal of the IBM Integration Bus node (if a failure terminal is wired to the map node).

To enable this feature, set the Audit Location to Memory in the IBM Transformation Extender map settings. Click Map > Map Settings > MapAudit > AuditLocation > Memory. The Sized setting can be either ON or OFF.

Map audit information is propagated to the LocalEnvironment.WTX.Audit.MapInst_x location in the IBM Integration Bus local environment tree, where x is the instance of the map run from the IBM Integration Bus. Map instances start at 1, are unique, and are sequential.

The type of audit data information that is propagated to the local environment tree depends on the IBM Transformation Extender Map settings. For example, to send the map execution summary, set MapAudit Switch setting to ON, and set the SummaryAudit Execution setting to Always.

References

Use the reference information to accomplish the tasks that your business needs.

- [Product requirements](#)
IBM Transformation Extender for IBM Integration Bus is packaged with the IBM Transformation Extender for Integration Servers product installation.
- [Supported message domains](#)
A *parser* is a program that interprets the bit stream of an incoming message and creates an internal representation of the message in a tree structure. The parser also regenerates a bit stream from the internal message tree representation for an outgoing message. Each parser is suited to a particular message domain. A *message domain* is class of messages, such as fixed-length binary messages or XML messages. IBM Integration Bus supplies a range of parsers to parse and write message formats.
- [Deploying map resources](#)
When you deploy message flows that contain TX Map nodes, you must also deploy the associated IBM Transformation Extender maps, and any files that the maps require.
- [Triggering maps to run](#)
When a message is propagated to the TX Map node from the preceding node in a message flow, the TX Map node responds to this event by triggering the map to run.
- [Input cards in maps](#)
The input cards of a map are all associated with a single input terminal of a TX Map node.
- [Output cards in maps](#)
The output cards of a map are associated with the output terminals of a TX Map node.
- [Sending output data from a map to an IBM Integration Bus node](#)
You can use a IBM Transformation Extender PUT map rule to send output data to another IBM Integration Bus node, either immediately, or based on the outcome of the map.
- [Specifying the map for the TX map node](#)
There are several different ways to specify the map that a TX Map node runs.

- **[Map caching](#)**
At design time, set map caching to optimize runtime performance.
- **[LocalEnvironment DynamicMap and MapServerLocation caching](#)**
When the TX Map node Cache map property is enabled, the TX Map node caches the path to a map and the dynamic map bytes that are configured in the local environment tree.
- **[Wildcards](#)**
Wildcards are represented by the asterisk (*) and question mark (?) characters, and are used to substitute specific portions of the file name property of the FileInput node, and the FileOutput node when you want to use a pattern instead of a specific name.
- **[Map Settings interface](#)**
Use the Map Settings interface to override map and card properties when you configured your TX Map node properties in the Basic tab with either Use map from project selected, or both Use external map and Local compiled map selected.
- **[Resource Registry](#)**
In IBM Integration Bus, you can dynamically modify and configure the location of the IBM Transformation Extender resource configuration file by using the IBM Transformation Extender property in the execution group to which the bar file is deployed.
- **[Samples](#)**
Use the samples that are included in the IBM Transformation Extender for IBM Integration Bus installation to demonstrate how you use the product.

Product requirements

IBM® Transformation Extender for IBM Integration Bus is packaged with the IBM Transformation Extender for Integration Servers product installation.

To use IBM Transformation Extender for IBM Integration Bus, other products must be installed and running. Product requirements are detailed in the [release notes](#).

Supported message domains

A *parser* is a program that interprets the bit stream of an incoming message and creates an internal representation of the message in a tree structure. The parser also regenerates a bit stream from the internal message tree representation for an outgoing message. Each parser is suited to a particular message domain. A *message domain* is class of messages, such as fixed-length binary messages or XML messages. IBM® Integration Bus supplies a range of parsers to parse and write message formats.

The following table lists the domain that you should use for the specified scenario.

Table 1. Message scenarios and domain types

Scenario	Domain
Message is modeled by type tree (.mtt), or XML Schema (.xsd), or broker message definition (.mxsd). Only the map processes the message content. The message is opaque to native broker nodes in the message flow.	BLOB
Message is a business object from a WebSphere® adapter, and is modeled by broker message definition (.mxsd).	DataObject
Message is not XML or a business object, and is modeled by Data Format Description Language (DFDL) schema.	DFDL
Message is not XML or a business object, and is modeled by broker message definition (.mxsd).	MRM
Message is XML, and is modeled by XML Schema (.xsd) or broker message definition (.mxsd).	XMLNSC

The TX Map node handles messages that belong in the following message domains, and these messages can be used as input to, and output from, map cards. The steps you take to create your message flow, and the runtime behavior of the TX Map node, depend on the domain that you selected.

- **[BLOB domain](#)**
IBM Integration Bus uses the BLOB parser to process messages that belong to the BLOB domain.
- **[DataObject domain](#)**
IBM Integration Bus uses the DataObject parser to read and write messages that belong to the DataObject domain.
- **[DFDL domain](#)**
IBM Integration Bus uses the DFDL parser to read and write messages that belong to the DFDL domain.
- **[MRM domain](#)**
IBM Integration Bus uses the MRM parser to read and write messages that belong to the MRM domain.
- **[XMLNSC domain](#)**
IBM Integration Bus uses the XMLNSC parser to read and write messages that belong to the XMLNSC domain.
- **[Other domains](#)**
IBM Integration Bus provides several other domains, such as XMLNS, MIME, JMSMap and JMSStream, and accompanying parsers to parse and serialize messages that belong to them.

BLOB domain

IBM® Integration Bus uses the BLOB parser to process messages that belong to the BLOB domain.

A BLOB message is handled as a single string of bytes, and although you can manipulate it, you cannot identify specific pieces of the byte string in the way that you can with messages in other domains. The BLOB parser does not create a message tree structure in the same way that other parsers do. It creates a simple tree with a root element BLOB that has a child element, also called BLOB, that contains the message bit stream.

The TX Map node can consume and create messages in the BLOB domain. However, the message must be modeled using a type tree so the map can interpret the structure of the BLOB message.

To create a message flow that uses the TX Map node to create or consume messages in the BLOB domain, do the following steps within the IBM Integration Toolkit:

1. In the Transformation Extender Development perspective, use Type Designer to model the message format as a type in a type tree (.mtt) file.
2. In the Transformation Extender Development perspective, use Map Designer to create a map, and add input or output cards as required. In the New Card dialog, select the type tree (.mtt) file as the Type tree, and the type as the Type.
3. Switch to the IBM Integration Development perspective, and create a message flow containing a TX Map node that uses your map.

- a. If the message is consumed by an input card, configure the input node of the message flow to specify the Message Domain as **BLOB**.
 - b. If the message is created by an output card, configure the corresponding TX Map node output terminal to specify the Message Domain as **BLOB**, complete the Message Set property, and set the Encoding and Coded Char Set ID properties to match the byte order (endianness) and character set of the output data from the map.
4. Complete the message flow, and deploy it to the broker.

At run time, if the message is to be consumed by an input card, the input node of the message flow starts the BLOB parser, which creates a simple message tree from the message bit stream. The TX Map node receives the BLOB message tree, extracts the BLOB bit stream element, and passes it to the input card. When the map runs, it parses the bit stream using the type tree.

At run time, if the message is to be created by an output card, when the map runs, it creates a bit stream. The TX Map node starts the BLOB parser to create a simple message tree from the bit stream, and propagates it to the next node in the message flow. The output node of the message flow starts the BLOB parser, which extracts the BLOB bit stream element.

The map is built using the type tree, and at run time, the map itself parses the message bit stream on input, and serializes it on output, using the type tree.

Related reference

- [Configuring connected output terminals](#)

DataObject domain

IBM® Integration Bus uses the DataObject parser to read and write messages that belong to the DataObject domain.

You must use the DataObject domain when you use the WebSphere® Adapter nodes in your message flow. IBM Integration Bus uses the DataObject parser to read and write messages, or business objects, from Enterprise Information Systems (EIS) such as SAP. The DataObject parser uses either XML schema files (.xsd) from an IBM Integration Bus library, or message definitions in a message set when reading and writing messages. A message definition (.mxsd) file is an annotated XML schema file.

The TX Map node can consume and create messages in the DataObject domain.

To create a message flow that uses the TX Map node to create or consume messages in the DataObject domain, do the following steps within the IBM Integration Toolkit:

1. In the IBM Integration Development perspective, use the Adapter Connection wizard to model the business object as either:
 - A global element in an XML schema file (.xsd) within a broker library
 - A message in a message definition (.mxsd) file, within a broker message set
- For detailed instructions, see the IBM Integration Bus product documentation.
2. In the Transformation Extender Development perspective, use Map Designer to create a map, and add input or output cards as required. In the New Card dialog, select the message definition (.mxsd) file or the XML schema (.xsd) file as the Type tree, and the group corresponding to the business object message as the Type.
 3. Switch to the IBM Integration Development perspective. Create a message flow that contains a TX Map node that uses your map.
 - a. If the message is consumed by an input card, configure the input node of the message flow to specify the Message Domain as **DataObject**.
 - b. If the message is created by an output card, configure the corresponding TX Map node output terminal to specify the Message Domain as **DataObject**. Set the Encoding and Coded Char Set ID properties to match the byte order (endianness) and the character set of the output data from the map.
 4. Complete the message flow and deploy it to the broker, along with the message set.

If the message is to be consumed by an input card, at run time the input node of the message flow starts the DataObject parser. The DataObject parser uses the XML schema or the message set to create a message tree from the EIS business object. The TX Map node receives the DataObject domain message tree and passes it to the input card. The map uses the tree directly when it runs.

If the message is to be created by an output card, at run time the map creates the DataObject domain message tree directly. The TX Map node propagates the message tree to the next node in the message flow. The output node of the message flow starts the DataObject parser, which uses the XML schema or the message set to create the EIS business object.

You do not have to create an equivalent type tree for the business object message. The map is built with the message definition file. At run time, the map operates directly on the DataObject domain message tree, and not on a bit stream.

Related reference

- [Configuring connected output terminals](#)

DFDL domain

IBM® Integration Bus uses the DFDL parser to read and write messages that belong to the DFDL domain.

Use the DFDL domain to parse and write a wide variety of message formats. The DFDL domain is intended for text and binary message formats that are described by Data Format Description Language (DFDL) schema files. The DFDL parser uses DFDL schema files (.xsd) from an IBM Integration Bus library when it parses and serializes messages. A DFDL schema (.xsd) file is an annotated XML Schema file. The TX Map node can consume and create messages in the DFDL domain.

Use the IBM Integration Toolkit to create a message flow that uses the TX Map node to create or consume messages in the DFDL domain.

1. In the Integration Development perspective, model the message format as a message in a DFDL (.xsd) file, within a broker library. See the IBM Integration Bus product documentation for details.
2. In the Transformation Extender Development perspective:
 - a. Use Map Designer to create a map and add input or output cards as required.
 - b. In the New Card dialog, select the DFDL (.xsd) file as the Type tree, and select XSD as the Type.
3. Switch to the Integration Development perspective, and create a message flow containing a TX Map node that uses your map.
 - If the message is consumed by an input card, configure the input node of the message flow to specify the Message Domain property as DFDL, and the Message property as the name of the message in the DFDL schema.

- If the message is created by an output card, configure the corresponding TX node output terminal to specify the Message Domain as DFDL, and the Message property as the name of the message in the DFDL schema. Set the Encoding and Coded Char Set ID properties to match the byte order (endianness) and character set of the output data from the map.
4. Complete the message flow and deploy it to the broker along with the message set.

When the message is to be consumed by an input card, at run time the input node of the message flow starts the DFDL parser. The DFDL parser uses the DFDL schema to create a message tree from the message bit stream. The TX Map node receives the DFDL domain message tree and passes it to the input card. When the map runs, it uses the tree directly.

When the message is to be created by an output card, when the map runs, it creates the DFDL domain message tree directly. The TX Map node propagates the message tree to the next node in the message flow. The output node of the message flow starts the DFDL parser. The DFDL parser uses the DFDL schema to serialize the message tree.

You do not have to create an equivalent type tree for the message. The map is built using the DFDL schema. At run time, the map operates directly on the DFDL domain message tree, and not on a bit stream.

MRM domain

IBM® Integration Bus uses the MRM parser to read and write messages that belong to the MRM domain.

Use the MRM domain to parse and write a wide variety of message formats. It is primarily intended for non-XML message formats, but it can also parse and write XML. The MRM parser always uses message definitions in a message set when parsing and serializing messages. A message definition (.mxsd) file is an annotated XML Schema file.

The TX Map node can consume and create messages in the MRM domain.

To create a message flow that uses the TX Map node to create or consume messages in the MRM domain, do the following steps within the IBM Integration Toolkit:

1. In the IBM Integration Development perspective, model the message format as a message in a message definition (.mxsd) file, within a broker message set. For detailed instructions, see the specific IBM Integration Bus product documentation.
2. In the Transformation Extender Development perspective, use Map Designer to create a map, and add input or output cards as required. In the New Card dialog, select the message definition (.mxsd) file as the Type tree, and the group corresponding to the message as the Type.
3. Switch to the IBM Integration Development perspective, and create a message flow containing a TX Map node that uses your map.
 - a. If the message is consumed by an input card, configure the input node of the message flow to specify the Message Domain as `MRM`, and complete the Message Set, Type, and Format properties.
 - b. If the message is created by an output card, configure the corresponding TX Map node output terminal to specify the Message Domain as `MRM`, complete the Message Set, Type and Format properties, and set the Encoding and Coded Char Set ID properties to match the byte order (endianness) and character set of the output data from the map.
4. Complete the message flow, and deploy it to the broker along with the message set.

At run time, if the message is to be consumed by an input card, the input node of the message flow starts the MRM parser, which creates a message tree from the message bit stream, using the message set. The TX Map node receives the MRM domain message tree, and passes it to the input card. When the map runs, it uses the tree directly. At run time, if the message is to be created by an output card, when the map runs, it creates the MRM domain message tree directly. The TX Map node propagates the message tree to the next node in the message flow. The output node of the message flow starts the MRM parser, which serializes the message tree, using the message set.

You do not have to create an equivalent type tree for the message. The map is built using the message definition file, and at run time, the map operates directly on the MRM domain message tree, and not on a bit stream.

Related reference

- [Configuring connected output terminals](#)

XMLNSC domain

IBM® Integration Bus uses the XMLNSC parser to read and write messages that belong to the XMLNSC domain.

The XMLNSC parser is a flexible, general-purpose XML parser that offers high performance XML parsing and optional XML schema validation. The XMLNSC parser has a range of options that make it suitable for most XML processing requirements. The XMLNSC parser uses either XML schema files (.xsd) from an IBM Integration Bus library, or message definitions in a message set to parse and serialize messages. A message definition (.mxsd) file is an annotated XML schema file.

The TX Map node can consume and create messages in the XMLNSC domain.

To create a message flow that uses the TX Map node to create or consume messages in the XMLNSC domain, do the following steps within the IBM Integration Toolkit:

1. In the IBM Integration Development perspective, model the message format as either:
 - A global element in an XML schema file (.xsd) within a broker library
 - A message in a message definition (.mxsd) file within a broker message set, typically by importing an XML schema (.xsd) file.
 For detailed instructions, see the IBM Integration Bus product documentation.
2. In the Transformation Extender Development perspective, use Map Designer to create a map and add input or output cards as required. In the New Card dialog, select either the message definition (.mxsd) file or the XML schema (.xsd) file as the Type tree. Select the group that corresponds to the message as the Type.
3. Switch to the IBM Integration Development perspective, and create a message flow that contains a TX Map node that uses your map.
 - a. If the message is consumed by an input card, configure the input node of the message flow to specify the Message Domain as `XMLNSC`.
 - b. If the message is created by an output card, configure the corresponding TX Map node output terminal to specify the Message Domain as `XMLNSC`. Set the Encoding and Coded Char Set ID properties to match the byte order (endianness) and character set of the output data from the map.
4. Complete the message flow, and deploy it to the broker along with the message set.

If the message is to be consumed by an input card, at run time the input node of the message flow starts the XMLNSC parser. The XMLNSC parser uses the XML schema or the message set to create a message tree from the message bit stream. The TX Map node receives the XMLNSC domain message tree and passes it to the input card. When the map runs, it uses the tree directly.

If the message is to be created by an output card, at run time the map creates the XMLNSC domain message tree directly. The TX Map node propagates the message tree to the next node in the message flow. The output node of the message flow starts the XMLNSC parser. The XMLNSC parser uses the XML schema or the message set to serialize the message tree.

You do not have to create an equivalent type tree for the XML message. The map is built with the message definition file or XML schema. At run time, the map operates directly on the XMLNSC domain message tree, and not on a bit stream.

Related reference

- [Configuring connected output terminals](#)
-

Other domains

IBM® Integration Bus provides several other domains, such as XMLNS, MIME, JMSMap and JMSStream, and accompanying parsers to parse and serialize messages that belong to them.

The TX Map node can consume and create messages in these domains.

To create a message flow that uses the TX Map node to create or consume messages in these domains, do the following steps within the IBM Integration Toolkit:

1. In the Transformation Extender Development perspective, use Type Designer to model the message format as a type in a type tree (.mtt) file.
2. In the Transformation Extender Development perspective, use Map Designer to create a map, and add input or output cards as required. In the New Card dialog, select the type tree (.mtt) file as the Type tree, and the type as the Type.
3. Switch to the IBM Integration Development perspective, and create a message flow containing a TX Map node that uses your map.
 - a. If the message is consumed by an input card, configure the input node of the message flow to specify the selected Message Domain.
 - b. If the message is created by an output card, configure the corresponding TX Map node output terminal to specify the selected Message Domain. Set the Encoding and Coded Char Set ID properties to match the byte order (endianness) and character set of the output data from the map.
4. Complete the message flow, and deploy it to the broker.

If the message is to be consumed by an input card, at run time the input node of the message flow starts the selected parser. The parser creates a message tree from the message bit stream. The TX Map node receives the message tree, serializes it back to a bit stream, and passes it to the input card. The map runs and uses the type tree to parse the bit stream.

If the message is to be created by an output card, at run time the map creates a bit stream. The TX Map node starts the selected parser to create a message tree from the bit stream, and propagates it to the next node in the message flow. The output node of the message flow starts the selected parser, which serializes the message tree.

The map is built by using the type tree. At run time, the map uses the type tree to parse the message bit stream on input, and serialize it on output.

Related reference

- [Configuring connected output terminals](#)
-

Deploying map resources

When you deploy message flows that contain TX Map nodes, you must also deploy the associated IBM® Transformation Extender maps, and any files that the maps require.

If the map is deployed to the broker in a map archive (mar) file, you must make sure that any required files reside in the same project as the map, so they are automatically included in the mar file.

If the map is manually deployed to the broker, you must also manually deploy any required files to the broker.

The two alternative map deployment mechanisms are described in [Deploying IBM Transformation Extender maps](#).

In particular, if your data is XML, and you have imported an XML Schema (.xsd file) when creating a type tree validated using Xerces, or used an XML Schema directly when creating a map card, the .xsd file is required by the map and must be deployed to the broker for use at run time in either of the following cases:

- if the data is being received from the message flow in the BLOB domain
- if the data is being received from a IBM Transformation Extender adapter

Related tasks

- [Deploying IBM Transformation Extender maps](#)
 - [Using a source map with a TX Map node](#)
 - [Using a precompiled map with a TX Map node](#)
-

Triggering maps to run

When a message is propagated to the TX Map node from the preceding node in a message flow, the TX Map node responds to this event by triggering the map to run.

The run unit within a IBM® Integration Bus execution group is a message flow. A message flow contains one or more input nodes that wait for input. After the nodes receive the input, they run the rest of the message flow.

When you use a IBM Transformation Extender map with IBM Integration Bus, the map can only be triggered to run within the context of a message flow, by a TX Map node. The message flow must also contain one or more input nodes. The TX Map node is not an input node, and can not start a message flow.

To configure a message flow that triggers a map to run, using the Message flow editor of the IBM Integration Toolkit, connect a TX Map node into a message flow, and then connect the input terminal of the TX Map node to the output terminal of a preceding node. Connections between objects in your message flows are called wires.

When the message flow is running, a message, which was propagated from the output terminal of the preceding node, arrives at the input terminal of the node in the form of a broker message tree. The TX Map node uses the content of the message tree to run the map.

- [Input nodes and events](#)

Input nodes and events trigger a message flow to run.

- [Triggering maps to run by a single event](#)

A single event can trigger a IBM Transformation Extender map to run.

- [Triggering maps to run by multiple events](#)

Multiple events can trigger a IBM Transformation Extender map to run.

- [Triggering maps to run in complex message flows](#)

A message flow can trigger systems of IBM Transformation Extender maps to run under complex triggering conditions.

Related reference

- [TX Map node exceptions](#)
-

Input nodes and events

Input nodes and events trigger a message flow to run.

You can use the following input nodes and events as triggers to run your message flow containing a TX Map node:

Input node or event	Trigger
FileInput	a file that appears in a file system folder or from an FTP server
MQInput	a message that arrives on a WMQ queue
HTTPInput	a message that arrives at an HTTP URI
SOAPInput	a Web Services SOAP that arrives at an HTTP URI
JMSInput	a message that arrives at a JMS destination
TCPIPCClientInput	a message that arrives at a TCPIP port
TCPIPServerInput	a message arriving at a TCPIP port
SCADAInput	a message that arrives from a SCADA input device
SAPInput	a business object that arrives from the WebSphere® Adapter for SAP
TimeoutNotification	a timer event that is signaled

Triggering maps to run by a single event

A single event can trigger a IBM Transformation Extender map to run.

When you have a message flow that contains a TX Map node that is preceded by a node other than a Collector node, a single event can trigger the IBM Transformation Extender map to run.

A single event can be a message that arrives at its destination. The node sends the message, in the form of a broker message tree, to the IBM Transformation Extender map input card that you configured on the Card number to wire property of the TX Map node.

The TX Map node runs the map. If the map has other input cards, it pulls data from each card's adapter.

Triggering maps to run by multiple events

Multiple events can trigger a IBM Transformation Extender map to run.

When you have a message flow that contains a TX Map node that is preceded by a Collector node, multiple events can trigger the IBM Transformation Extender map to run.

The Collector node in a message flow groups together multiple messages that arrive at its input terminals, and runs the rest of the flow when triggering criteria that you have defined are met. It performs event triggering functions similar to the IBM Transformation Extender with Launcher product.

The node propagates all the messages it receives in a message collection, which is a type of broker message tree. Each message is contained within a folder in the message collection. The folder names are taken from the names of the input terminals of the Collector node. When you use the Collector node and TX Map nodes together in your message flow, you must add Collector input terminals with names that match the card names, not the card numbers, of the map's input cards.

If the IBM Transformation Extender map has an input card with a card name that matches the name of a folder in the message collection, the Collector node delivers the message in that folder to that input card. After the Collector node delivers the messages within all of the folders in the collection to the map's input cards with matching names, the TX Map node runs the map.

If the IBM Transformation Extender map has other input cards that do not match any folder names in the message collection, these input cards do not receive messages, and the map pulls data from each of these card's adapter.

Triggering maps to run in complex message flows

A message flow can trigger systems of IBM® Transformation Extender maps to run under complex triggering conditions.

A more complex example of a message flow might contain multiple Connector and TX Map nodes connected together. For example, it can contain several input nodes that are connected to a Collector node, which is configured to trigger the rest of the message flow after it receives a message from all the input nodes. The Collector node generates the message collection, propagates it to a TX Map node, which triggers a IBM Transformation Extender map that has multiple output cards to run. Each output card propagates an output message to a separate branch of the flow, each of which contains additional Collector and TX Map nodes.

Through this network of message flows containing complex triggering conditions, systems of maps can be triggered to run.

Input cards in maps

The input cards of a map are all associated with a single input terminal of a TX Map node.

When you drag a blank TX Map node from the IBM® Integration Toolkit palette onto the canvas for your message flow, it contains a single input terminal. All cards that require data from the message flow will receive their data through this single input terminal.

- [Map triggered by multiple inputs](#)

If your map is triggered to run by multiple inputs, you must use a Collector node in your message flow before the TX Map node.

- [Wired input cards](#)

Wired input card is the term used when a map input card receives its data from another node in the message flow. The TX Map node receives the data in the form of a broker message tree.

- [Unwired input cards](#)

Unwired input card is the term used when a map input card does not receive its data from another node in the message flow, but instead gets its data from the IBM Transformation Extender adapter specified on the input card.

Map triggered by multiple inputs

If your map is triggered to run by multiple inputs, you must use a Collector node in your message flow before the TX Map node.

Configure the Collector node to gather the required inputs together in a message collection. When the conditions that you set for the message collection have been met, the message collection is complete, and the inputs are propagated as a message collection tree. When a broker message collection tree arrives at the input terminal of the TX Map node, the individual messages it contains are passed to the corresponding input cards, and the map runs. Any input cards that did not receive data get data from their adapter.

Any card number specified on the Input tab of the TX Map node properties is ignored when using a message collection tree.

If you need to wire, or connect, only a single input card to run the map, use a non-Collector node type. Wire the input node directly to the input terminal of the TX Map node, and specify the card to wire in the node's properties. This is a case where you have a map with multiple input cards, but only one of the cards is required for the message to trigger the map to run.

IBM® Transformation Extender for IBM Integration Bus adds input card properties for the TX Map node, based on the number of input cards in the compiled map. If the input terminal is wired to a Collector node, the Collector node determines which input cards are physically wired in the TX Map node. The TX Map node input card property determines which input card is wired for a non-Collector node message. For compatibility to previous versions, the default is the first input card.

- [Collector node](#)

You can use a Collector node to gather multiple input messages from IBM Integration Bus into a single TX Map node. This is also known as fan-in.

Collector node

You can use a Collector node to gather multiple input messages from IBM® Integration Bus into a single TX Map node. This is also known as fan-in.

Use a Collector node where multiple messages are required to trigger the map to run.

The Collector node gathers input messages, and when specified event criteria have been met, propagates the input messages in the form of a message collection tree. Each message is contained in a folder in the tree. The name of the folder is the same as the name of the Collector node input terminal that received that message.

The TX Map node interprets broker message collection trees. At run time, when a message collection arrives at the input terminal of a TX Map node, it disassembles it, and passes the individual input messages to the input cards of the IBM Transformation Extender map. The name of the folder in the message collection determines the input card that receives that message.

For example, a map has two input cards. The name of card #1 is PurchaseOrder, and the name of card #2 is AddressDetails. You want the map to run only when both cards have data. Configure the Collector node at design time to create message collections that consist of one purchase order message, and one address details message. If you dynamically add two input terminals to the Collector node, one named PurchaseOrder, and one named AddressDetails, to match the map input card names, the propagated message collection will contain folders named PurchaseOrder and AddressDetails.

Input cards, which do not have associated folders in the message collection, get their data from the IBM Transformation Extender adapter specified on the card.

The Collector node input terminal names and the map input card names are case sensitive. If you do not name them the same, the message collection will have a folder name that does not match the name of the input card. This condition causes the message in the folder to be discarded, and the input card to get its data from the IBM Transformation Extender adapter specified on the card. The name of the Collector node input terminal can not contain dots (.). Therefore, if you create a map that is used in a IBM Integration Bus message flow, the names of the map input cards that are wired to the previous Collector node in the flow can not contain dots (.).

The Collector node provides a comprehensive set of controls that enable you to set complex event criteria.

For more information about the Collector node, and message collection, see the specific IBM Integration Bus product documentation.

Related tasks

- [Using the TX Map node with a source map and multiple inputs](#)
 - [Using the TX Map node with a precompiled map and multiple inputs](#)
-

Wired input cards

Wired input card is the term used when a map input card receives its data from another node in the message flow. The TX Map node receives the data in the form of a broker message tree.

If the tree represents a single message, the TX Map node passes the message to a single, specified input card.

If the tree is a message collection, containing several messages, the TX Map node passes each message to the appropriate input card.

Unwired input cards

Unwired input card is the term used when a map input card does not receive its data from another node in the message flow, but instead gets its data from the IBM® Transformation Extender adapter specified on the input card.

This occurs when the broker message tree arriving at the input terminal of the TX Map node does not contain a message for the input card.

Output cards in maps

The output cards of a map are associated with the output terminals of a TX Map node.

When you drag a blank TX Map node from the IBM® Integration Toolkit palette onto the canvas for your message flow, it contains only a failure terminal. After you associate the node with a map, terminals appear on the node icon.

- [Creating the output terminals](#)
The TX Map node creates the output terminals dynamically, where possible. Otherwise, you create them manually.
 - [Connecting output terminals to nodes](#)
You can connect an output terminal to a subsequent node, although not all output terminals need to be connected.
 - [Configuring connected output terminals](#)
You can configure additional properties for any output terminals that you connect to nodes when you want to override the default behavior.
 - [Wired output cards](#)
Wired output card is the term used when the TX Map node output terminal corresponding to a map output card is connected, or wired, to another node in the message flow.
 - [Unwired output cards](#)
Unwired output card is the term used when the TX Map node output terminal corresponding to a map output card is not connected, or wired, to another node in the message flow.
-

Creating the output terminals

The TX Map node creates the output terminals dynamically, where possible. Otherwise, you create them manually.

The TX Map node creates the output terminals dynamically when the TX Map node in your message flow is configured, in the Basic tab of the Properties view, to reference a map on your local system:

- For a precompiled map, click Browse to set the Local compiled map property.
- For a source map, click Browse to set the Executable and Source map properties.

The node creates one output terminal for each output card of the map.

The node cannot create the output terminals dynamically when you select a precompiled map, which is located remotely. IBM® Integration Toolkit cannot determine how many output cards the map has. You have to manually add the correct number of output terminals by right-clicking the TX Map node, and selecting the Add Output Terminal option.

Related tasks

- [Using source maps](#)
 - [Using precompiled maps](#)
-

Related reference

- [Sending output data from a map to an IBM Integration Bus node](#)
-

Connecting output terminals to nodes

You can connect an output terminal to a subsequent node, although not all output terminals need to be connected.

Where the output terminal is connected, the data flows from the map output card to the output terminal, and is propagated down the connection to the next node in the message flow. Where the output terminal is not connected, the data flows from the map output card to the IBM® Transformation Extender adapter specified on the card.

The Terminal Selection dialog is displayed when you right-click and select Create connection, or if the node has so many terminals that they have been grouped together in the user interface, for example, a Compute node or TX Map node with many output cards.

When you connect a TX Map node output terminal to a subsequent node, IBM Integration Toolkit displays the Terminal Selection dialog if the map has four or more output cards. The Terminal Selection dialog displays the output terminals, plus an additional failure terminal, for you to select. As an example, if your map has four output cards, the following terminals are available for you to select from the list of terminals:

- failure
- out1
- out2
- out3
- out4

Related reference

- [Sending output data from a map to an IBM Integration Bus node](#)

Configuring connected output terminals

You can configure additional properties for any output terminals that you connect to nodes when you want to override the default behavior.

You must configure the message domain. The domain that you use determines which message template properties, message set, message type, and message format, you need to configure. See [Supported message domains](#). Configure the message template properties for the output terminals through the Outputs tab in the Properties view.

You must configure the Encoding and Coded Char Set ID properties. These properties must match the byte order (endianness) and character set of the output data from the map. Configure these properties for the output terminals through the Outputs tab in the Properties view. The values you need to use follow IBM Integration Bus conventions, described in the help topic about converting data with message flows in the IBM Integration Bus documentation.

If you do not configure these properties for an output terminal, IBM Transformation Extender for IBM Integration Bus uses the corresponding properties from the input tree. As an example, if the input belongs to the XMLNSC message domain, and you do not configure the message template properties for the output terminal, TX Map node creates the output message using the XMLNSC domain. If the message flow uses a Collector node, TX Map node creates the output message using the domain to which the tree on the first input message belongs.

When the Outputs tab initially opens, there are no entries in the Properties table. Add the properties for each of the connected output terminals. After you configure additional properties for a connected output terminal, the Properties table displays the card's number in the following format:

card-number

Examples are:

1
2

Related concepts

- [Supported message domains](#)

Related tasks

- [BLOB domain](#)
- [XMLNSC domain](#)
- [DataObject domain](#)
- [MRM domain](#)
- [Other domains](#)

Related reference

- [Sending output data from a map to an IBM Integration Bus node](#)

Wired output cards

Wired output card is the term used when the TX Map node output terminal corresponding to a map output card is connected, or wired, to another node in the message flow.

The data from that card is propagated to the connected node in the form of a broker message tree.

Related reference

- [Sending output data from a map to an IBM Integration Bus node](#)

Unwired output cards

Unwired output card is the term used when the TX Map node output terminal corresponding to a map output card is not connected, or wired, to another node in the message flow.

The data from that card is propagated to the IBM® Transformation Extender adapter specified on the output card. No data is propagated to the message flow.

Sending output data from a map to an IBM Integration Bus node

You can use a IBM Transformation Extender PUT map rule to send output data to another IBM® Integration Bus node, either immediately, or based on the outcome of the map.

To send the output data, wire an output terminal to a IBM Transformation Extender map node or RUN map that uses the PUT rule. You wire a map node to the output terminal specified in the map's message flow. The map must use the PUT rule and specify WIRE as the adapter name:

```
=PUT("WIRE", "output_terminal_name", data)
```

Optionally, you can specify that the message is to be sent immediately. The -NOW argument sends the message to the output terminal as soon as the PUT rule is processed. When you omit the -NOW argument, a map node sends a message to the output terminal when the map completes, and a RUN map sends a message to the output terminal when the top map completes. If you omit the -NOW argument and the map node or RUN map fails, the message is not sent.

```
=PUT("WIRE", "output_terminal_name -NOW", data)
```

Related reference

- [Wired output cards](#)
- [Creating the output terminals](#)
- [Connecting output terminals to nodes](#)
- [Configuring connected output terminals](#)

Specifying the map for the TX map node

There are several different ways to specify the map that a TX Map node runs.

At design time, you can specify the following map information on the TX Map node:

- Name of an executable map within a source map (.mms) file.
 - When IBM® Integration Bus adds the message flow containing the TX Map node to a broker archive (.bar) file, the specified executable map is compiled automatically to create a compiled map (.mmc) file.
 - The compiled map, and other files in the project, are zipped into a map archive (.mar) file, which is added to the broker archive file. The map is, therefore, automatically deployed to the broker when you deploy the message flow.
- Location of a precompiled map (.mmc) file on the file system of the broker's machine.
 - When IBM Integration Bus adds the message flow containing the TX Map node to a broker archive (.bar) file, the map is not automatically compiled, and is not added to the broker archive file.
 - You must manually deploy the map to the location that you specified.

At deploy time, you can specify the location of a compiled map by using the Configure tab in the Properties view, which is under the Manage tab in the Broker Archive editor. This overrides the map that you specified at design time.

At run time, your message flow can dynamically specify the location of a compiled map, and pass this to the TX Map node, by using the Local Environment message tree. This takes precedence over any of the other methods.

- [Using source maps](#)
Configure a TX Map node in your message flow to use a source map.
- [Using precompiled maps](#)
Configure the TX Map node in your message flow to use a precompiled map.
- [Changing a source map compared to changing a compiled map](#)
When you design a message flow to use either a source map or a compiled map, consider whether you might want to change the map later. There are fewer steps required to change a compiled map than to change a source map.
- [Changing a message flow to point to a different map](#)
When you design a message flow to use either a source map or a compiled map, consider whether you might change the message flow later to point to a different map. There are fewer steps required to point to a different a compiled map than to point to a different source map. To dynamically change a message flow to point to a different map, design the message flow to use compiled maps.
- [Deploy-time TX Map node configuration](#)
You can modify certain properties of a TX Map node, when the containing message flow has been added to a bar file, using the Configure tab of the Broker Archive editor.
- [Overriding map properties at run time](#)
Override map properties at run time by using the IBM Integration Bus LocalEnvironment.

- [Dynamically overriding a map with a compiled map](#)

An upstream node in a message flow can dynamically override a map that is configured in the properties of a TX Map node. The upstream node can populate the IBM Integration Bus local environment tree with the binary data of a compiled map. The map bytes in the local environment tree override the map that is configured on the TX Map node. Ensure that the map in the local environment tree is compiled for the appropriate operating system platform.

Using source maps

Configure a TX Map node in your message flow to use a source map.

To use a source map:

1. Switch to the IBM® Integration Development perspective, and open your message flow in the project.
2. Select a TX Map node, and navigate to the Basic tab of the Properties view.
3. Select Use map from project.
4. Browse for the source map (.mms) file by clicking Browse, and use the wizard to locate the source map.
5. Select an executable map from the source map.

If you do not see your map inside the source map file, it might be set up as a functional map. For detailed information about functional maps, see the Map Designer documentation.

6. The number of output cards in the executable map determines the number of output terminals that are automatically added to the TX Map node.
7. To override any of the map or card settings, switch to the Map Settings tab, and click Map Settings.

Automatically, the map is compiled when IBM Integration Bus adds the message flow to a broker archive (bar) file. A map archive (mar) file is created containing compiled map (.mmc) files, which is added to the current project in the workspace and to the bar file. Then you deploy the bar file to the broker.

Related reference

- [Changing a source map compared to changing a compiled map](#)
- [Changing a message flow to point to a different map](#)

Using precompiled maps

Configure the TX Map node in your message flow to use a precompiled map.

To use a compiled map:

1. Switch to the IBM® Integration Development perspective, and open your message flow in the project.
2. Select a TX Map node, and navigate to the Basic tab of the Properties view.
3. To select the compiled map when it is located on an external file system, which is on the same machine as where you develop the message flow and where the broker runs:
 - a. Select Use external map.
External maps are compiled outside any projects. They must exist at design time for the map node to determine the number of output cards, and create the output terminals. They would not be transferred to the server because this scenario is using compiled maps that are on the same machine.
 - b. Enter the fully qualified path to the compiled map that is located on your local machine in the Local compiled map field, or click Browse [...], select the compiled map from the file system in the Browse Compiled Map dialog, and click OK.
IBM Integration Toolkit creates one output terminal for each output card in the map on the TX Map node.
The TX Map node is configured to use the compiled map from an external location.
4. To select the compiled map when you have a copy of the map in the machine where you are developing the message flow, and have to import it into your workspace project:
 - a. Import the map and type trees into your workspace project.
 - i. Select File > Import. The Import wizard starts.
 - ii. Expand the Other folder, select Project Interchange, and click Next.
 - iii. Navigate to the directory in which your map and type trees are located, select them, and click Open.
 - iv. Select the check box next to the displayed project, and click Finish. The map and type trees are imported into your workbench project.
 - b. Select Use map from project.
 - c. Enter the map in the Executable map field, or click Browse [...], select the executable map from the file system in the Select executable map dialog, and click OK.
This populates the Executable map and Source map edit fields. The source map is read only and is derived from the executable map selection. IBM Integration Toolkit also creates output terminals automatically for each output card of the map on the TX Map node.
The TX Map node is configured to use the compiled map from your workspace.
5. To select the compiled map when you do not have a copy of the map in the machine where you are developing the message flow, and the map and broker are both on the same remote machine:
 - a. Select Use external map.
External maps are compiled outside any projects. An external map must exist at design time and be compiled for the correct platform. If the server is a different machine, you must manually transfer the map to the server.
 - b. Enter the fully qualified name of the compiled map in the Map Server Location field.
The TX Map node is configured to use the compiled map from a remote location.

The TX Map node does not use the bar file mechanism to deploy the map. You deploy the map outside of the bar file deployment. You must manually create and build the bar file, without a mar file, and then deploy it to the broker. At run time, IBM Integration Bus finds and runs the compiled map.

If you want to use a different compiled map than the one you specified in the TX Map node, specify it in the LocalEnvironment. At run time, the compiled map specified in the LocalEnvironment overrides the map that you specified in the TX Map node. For more information about the LocalEnvironment, see the specific IBM Integration Bus product documentation.

Related reference

- [Overriding map properties at run time](#)
- [Dynamically overriding a map with a compiled map](#)
- [Changing a source map compared to changing a compiled map](#)
- [Changing a message flow to point to a different map](#)

Changing a source map compared to changing a compiled map

When you design a message flow to use either a source map or a compiled map, consider whether you might want to change the map later. There are fewer steps required to change a compiled map than to change a source map.

To change a source map:

1. Use IBM® Transformation Extender Design Studio or the IBM Transformation Extender perspective of IBM Integration Bus to make the map changes.
2. Recompile the broker archive (BAR) file.
3. Deploy the new BAR file.

To change a compiled map:

1. Use IBM Transformation Extender Design Studio or the IBM Transformation Extender perspective of IBM Integration Bus to make the map changes.
2. Replace the original map with the revised map in the database or file system where the BAR file is deployed.

Map Type	Edit Map	Recompile BAR File	Deploy New BAR File
Source map	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
Compiled map	<input checked="" type="checkbox"/>		

See the IBM Integration Bus documentation for additional information about compiling and deploying a broker archive file.

Related tasks

- [Using precompiled maps](#)
- [Using source maps](#)

Related reference

- [Changing a message flow to point to a different map](#)

Changing a message flow to point to a different map

When you design a message flow to use either a source map or a compiled map, consider whether you might change the message flow later to point to a different map. There are fewer steps required to point to a different a compiled map than to point to a different source map. To dynamically change a message flow to point to a different map, design the message flow to use compiled maps.

To change a message flow to point to a different source map:

1. In the TX map node, select a new source map or executable map.
2. Recompile the broker archive (BAR) file.
3. Deploy the new BAR file.

You cannot dynamically change the TX map node's executable map or source map properties by using the the BAR file override properties in IBM® Integration Toolkit or by using the IBM Integration Bus **mqsiapplybaroverride** command.

To change a message flow to point to a different compiled map, you can do one of the following:

- Change the map server location property:
 1. In the TX map node, change the Map Server Location property to point to a different compiled map.
 2. Recompile the broker archive (BAR) file.
 3. Deploy the new BAR file.
- Override the BAR file:
 1. Use the BAR file override properties in IBM Integration Toolkit or the IBM Integration Bus **mqsiapplybaroverride** command to change the map server location.
 2. Redeploy the BAR file. You do not have to recompile the BAR file.
- Specify a dynamic binary map or a dynamic path to a map in the IBM Integration Bus local environment tree
This method takes precedence over the map name or map location specified in the IBM Integration Toolkit or by the BAR file. When you use the local environment tree to specify the map, you do not have to recompile or redeploy the BAR file.

Related tasks

- [Using precompiled maps](#)
- [Using source maps](#)

Related reference

- [Changing a source map compared to changing a compiled map](#)

Deploy-time TX Map node configuration

You can modify certain properties of a TX Map node, when the containing message flow has been added to a bar file, using the Configure tab of the Broker Archive editor.

The properties you can override are the Map Server Location for a precompiled map that you specified outside of a project, the Card Number To Wire, and the Cache Map.

In the Broker Archive editor, select the Manage tab, and in the Configure tab of the Properties view, modify the properties of the TX Map node. If you override the map with an invalid configuration with other node properties, such as a wired input or output that does not exist, the node fails.

Overriding map properties at run time

Override map properties at run time by using the IBM® Integration Bus LocalEnvironment.

The message flow derives the map properties from different sources, and searches for the properties in the order listed:

1. Run time
 - Source: LocalEnvironment
 - Override the map server location and input card to wire map properties at run time in the LocalEnvironment.
 - For information about how the LocalEnvironment works, see the related topics in the IBM Integration Bus product documentation.
2. Deploy time
 - Source: broker archive (bar) file
 - Override the cache map, map server location, and input card to wire map properties at deploy time in the bar file using the Configure tab of the Properties view, which is under the Manage tab in the Broker Archive editor.
3. Design time
 - Source: TX Map node
 - Override the map and card properties at design time in the Map Settings tab of the TX Map node.

At run time, override the following map properties by using the IBM Integration Bus LocalEnvironment tree from a prior node in the message flow, such as a Compute node, which can change the LocalEnvironment to override parameters in the tree that is passed to the TX Map node:

MapServerLocation property

LocalEnvironment tree path: LocalEnvironment.WTX.MapServerLocation

Purpose: To run the compiled map located in a different location than the location configured on the TX Map node.

Result: IBM Integration Bus overrides the map server location with the location you specified in the LocalEnvironment tree, and the TX Map node runs the map from this location instead.

CardNumberToWire property

LocalEnvironment tree path: LocalEnvironment.WTX.InputCardNumberToWire

Purpose: To specify which input card of the map should be wired.

Result: The specified map input card receives its data from the prior node in the message flow.

Related tasks

- [Using precompiled maps](#)

Dynamically overriding a map with a compiled map

An upstream node in a message flow can dynamically override a map that is configured in the properties of a TX Map node. The upstream node can populate the IBM® Integration Bus local environment tree with the binary data of a compiled map. The map bytes in the local environment tree override the map that is configured on the TX Map node. Ensure that the map in the local environment tree is compiled for the appropriate operating system platform.

LocalEnvironment subtree

LocalEnvironment.WTX.DynamicMap

Contains the binary data of a compiled map. The map bytes dynamically override the map that is configured on a downstream TX Map node.

LocalEnvironment.WTX.DynamicMapName

Assigns a name to the binary data of the compiled map in the local environment tree. This name identifies the binary map for purposes such as map caching.

LocalEnvironment.WTX.MapServerLocation

Specifies the path to a map on the file system. This map runs instead of the map that is configured in the TX Map node.

Order of precedence

The order of precedence for map overrides is:

1. A dynamic binary map in the local environment tree (`LocalEnvironment.WTX.DynamicMap`)
2. A dynamic path to a map in the local environment tree (`LocalEnvironment.WTX.MapServerLocation`)
3. A map that is configured through the TX Map node properties, whether it is configured through the map settings interface or by a WebSphere® IBM Integration Bus broker archive (BAR) editor.

Default directory locations for dynamic maps

The default location of a dynamic map's log, trace, and work files is:

`MQSI_WORKPATH/common/log/WTX/ExeGrpName/FlowName/NodeName/DynamicMapName`

If you do not provide a dynamic map name by using `LocalEnvironment.WTX.DynamicMapName`, the dynamic map bytes are in the directory:

`MQSI_WORKPATH/common/log/WTX/ExeGrpName/FlowName/NodeName/NodeName`

You can find the value of `MQSI_WORKPATH` in a WebSphere IBM Integration Bus command shell environment.

If the dynamic map requires static files or RUN maps, put them in the same directory as the dynamic map's log, trace, and work files.

Related concepts

- [Map caching](#)

Related tasks

- [Using precompiled maps](#)
-

Map caching

At design time, set map caching to optimize runtime performance.

Map caching optimizes the runtime performance because the process does not load the compiled map for every iteration of the running of the map. The compiled map is loaded only on the first invocation of the map.

When map caching is set on, any changes that you make to the compiled map are used by the message after you stop and restart the execution group. When map caching is set off, any changes that you make to the compiled map are used by the next message that passes through the message flow.

Set up map caching for each TX Map node to enable individual control for each map node instance. Set map caching on in the Basic tab under Properties in the IBM® Integration Toolkit.

Even after a map is cached, the physical file in the path for the cached map must exist at run time.

A TX Map node can use a map that is cached by a different node if the TX Map node specifies the exact path and name of the cached map.

You can enable RUN map caching in the `/runtime/RUN Maps/RunMapMaxCacheNum` path of the `config.yaml` configuration file. Set `RunMapMaxCacheNum` to the maximum number of maps to cache.

Related reference

- [Dynamically overriding a map with a compiled map](#)
-

LocalEnvironment DynamicMap and MapServerLocation caching

When the TX Map node Cache map property is enabled, the TX Map node caches the path to a map and the dynamic map bytes that are configured in the local environment tree.

Dynamic map caching

When Cache map is enabled and a dynamic map has a name (`LocalEnvironment.WTX.DynamicMapName`), the node caches the bytes of a dynamic map (`LocalEnvironment.WTX.DynamicMap`). The node cannot cache dynamic map bytes that do not have a dynamic map name.

After you cache a dynamic map, a TX Map node that invokes the dynamic map name does not need to retrieve the map bytes from the local environment tree. If the TX Map node receives the dynamic map name (`LocalEnvironment.WTX.DynamicMapName`) together with at least one placeholder byte in `LocalEnvironment.WTX.DynamicMap`, the node runs the cached map.

For example, if you cache a map named PreProcessEDI and a TX Map node receives the following input from the local environment tree on a subsequent invocation, the TX Map node runs the cached PreProcessEDI map:

```
SET OutputLocalEnvironment.WTX.DynamicMapName = 'PreProcessEDI';
SET OutputLocalEnvironment.WTX.DynamicMap = X'00';
```

To enable the TX Map node to detect changes to a cached dynamic map, use a new map name (`LocalEnvironment.WTX.DynamicMapName`) to trigger the node to load the map bytes (`LocalEnvironment.WTX.DynamicMap`).

MapServerLocation caching

When you set the TX Map node Cache map property, the TX Map node caches each unique MapServerLocation path in the local environment tree. The TX Map node does not need to load a cached map path for subsequent invocations of the map, but the map must exist.

To enable the TX Map node to detect changes to a cached map path, use a new map server location (`LocalEnvironment.WTX.MapServerLocation`) to trigger the node to load a new map path.

Wildcards

Wildcards are represented by the asterisk (*) and question mark (?) characters, and are used to substitute specific portions of the file name property of the FileInput node, and the FileOutput node when you want to use a pattern instead of a specific name.

The FileInput node is the only input node that handles wildcards.

The wildcard character represented by the asterisk (*) is used to substitute 0 or more characters. The wildcard character represented by the question mark (?) is used to substitute 1 character.

For information about how wildcards are resolved on the Collector node, the FileInput node, and the FileOutput node, see the related topics in the IBM® Integration Bus product documentation.

- [Using wildcards](#)

You can use the wildcard characters for IBM Transformation Extender resources that are not wired, which includes non-wired sources and targets, only if you use wildcards on the IBM Integration Bus FileInput node that you selected as the input to the TX Map node.

Using wildcards

You can use the wildcard characters for IBM® Transformation Extender resources that are not wired, which includes non-wired sources and targets, only if you use wildcards on the IBM Integration Bus FileInput node that you selected as the input to the TX Map node.

The FileInput node resolves any wildcards in its file name or pattern properties, and places the evaluated wildcard expression at \$LocalEnvironment/Wildcard/WildcardMatch. The TX Map node uses this result for non-wired input and output cards that contain a wildcard to evaluate the correct source and target name, for example, filename and command line.

If you want more than one input card of the IBM Transformation Extender map to receive data from IBM Integration Bus, you need to use the Collector node. The Collector node can pass wildcard values from the FileInput node to the TX Map node when you set the Correlation path property of the Collector node to \$LocalEnvironment/Wildcard/WildcardMatch in the Basic tab of the Properties view.

One way to use wildcards is to represent a portion of a name for an output file that you want IBM Transformation Extender to create, so that it includes the corresponding characters from a message correlation ID that IBM Transformation Extender receives.

One example is if you configured a FileInput node with the c:*inputdata.txt file name, and a IBM Transformation Extender non-wired output file with the DATA*.txt name. If the FileInput node receives an input file with the 123inputdata.txt name, the wildcard expression evaluates to 123, and the result is that IBM Transformation Extender produces an output file with the DATA123.txt name.

Another example is if you configured a FileInput node with the c:*input??data.txt file name, and a IBM Transformation Extender non-wired output file with the DATA*.txt name. If the FileInput node receives an input file with the 123inputHQdata.txt name, the wildcard expression evaluates to 123inputHQ, and the result is that IBM Transformation Extender produces an output file with the DATA123inputHQ.txt name.

Map Settings interface

Use the Map Settings interface to override map and card properties when you configured your TX Map node properties in the Basic tab with either Use map from project selected, or both Use external map and Local compiled map selected.

If you configured your TX Map node properties with Use external map selected, and only Map server location selected, without Local compiled map selected as well, the map settings are disabled.

Override the map and card settings in the Map Settings tab of the TX Map node.

Through the Map Settings tab, override card settings, such as audit, trace, paging, and so forth, for all sources and targets of the map. Only unwired cards from IBM® Integration Bus use the card settings that you changed in the Map Settings interface.

Resource Registry

In IBM® Integration Bus, you can dynamically modify and configure the location of the IBM Transformation Extender resource configuration file by using the IBM Transformation Extender property in the execution group to which the bar file is deployed.

The Resource Registry can be used only with IBM Transformation Extender resources. See the Resource Registry documentation for details.

- [Modifying resource names](#)

Use the IBM Transformation Extender editor to create resource aliases to be used with IBM Transformation Extender resources in IBM Transformation Extender for IBM Integration Bus running under IBM Integration Bus.

- [Setting resource registry on execution groups](#)

Configure IBM Transformation Extender for IBM Integration Bus to use the IBM Transformation Extender Resource Registry by setting the resource configuration (.mrc) file on the IBM Integration Bus execution groups.

Modifying resource names

Use the IBM® Transformation Extender editor to create resource aliases to be used with IBM Transformation Extender resources in IBM Transformation Extender for IBM Integration Bus running under IBM Integration Bus.

You can use resource aliases only on TX Map nodes. The IBM Transformation Extender resources used in TX Map nodes that can be aliased include:

- non-wired source and target resources that are specified in map cards
- GET and PUT functions in map rules
- RUN map overrides
- audit file names
- trace file names
- back up file names
- work space locations

For a complete list of the resources, see the Resource Registry documentation.

Setting resource registry on execution groups

Configure IBM® Transformation Extender for IBM Integration Bus to use the IBM Transformation Extender Resource Registry by setting the resource configuration (.mrc) file on the IBM Integration Bus execution groups.

Update the default settings of the execution groups to use the Resource Registry. All command properties that are listed in the following steps are case-sensitive. The sample commands are shown for the Windows operating system.

To set the resource configuration file on execution groups:

1. Verify that the broker is running.
2. Open a Command Console that has the IBM Integration Bus environment configured for your current installation. Select Start>IBM Integration Bus *n.n>* Command Console, where *n.n* represents the product version number.
3. To find out where the current execution group's resource configuration file is located, use the IBM Integration Bus **mqswireportproperties** command:

```
C:\Program Files\IBM\MQSI\n.n>
mqswireportproperties brokername
-e executiongroup
-o ComIbmWTXManager
-n resourceConfigFile
```

An example of what your default setting might be:

```
C:\Program Files\IBM\MQSI\6.1>
mqswireportproperties WBRK61_DEFAULT_BROKER
-e default
-o ComIbmWTXManager
-n resourceConfigFile
```

The Command Console returns the following response:

```
resource.mrc
BIP8071I: Successful command completion.
```

4. To set the location of the resource configuration file for an execution group, use the IBM Integration Bus **mqsicchangeproperties** command:

```
C:\Program Files\IBM\MQSI\n.n>
mqsicchangeproperties brokername
-e executiongroup
-o ComIbmWTXManager
-n resourceConfigFile
-v locaton_of_your_resource_registry_configuration_file
```

An example of what your default setting might be:

```
C:\Program Files\IBM\MQSI\6.1>
mqsicchangeproperties WBRK61_DEFAULT_BROKER
-e default
-o ComIbmWTXManager
-n resourceConfigFile
-v C:\myfile.mrc
```

The Command Console returns the following response:

```
C:\myfile.mrc
BIP8071I: Successful command completion.
```

5. To have the changes take effect, stop and restart the execution group.

For information about these commands, see the specific IBM Integration Bus product documentation.

Samples

Use the samples that are included in the IBM® Transformation Extender for IBM Integration Bus installation to demonstrate how you use the product.

For the latest information about where the samples are installed, and how you use them, see the IBM Transformation Extender for Integration Servers [release notes](#).

The following table lists the Application samples that are available for IBM Transformation Extender for IBM Integration Bus.

Sample name	Description
IVT	A sample that uses a message flow, which runs a IBM Transformation Extender map. You can run this sample as an installation verification test (IVT) of the IBM Transformation Extender product components.
Node	A sample that shows how the TX Map node runs a IBM Transformation Extender map within a message flow.

Containers Overview

Containers are executable units of software that package application code along with its libraries and dependencies. IBM Sterling Transformation Extender (ITX) product offers a runtime container image for running ITX maps and flows through a REST API. The runtime server container image is distributed as, IBM Sterling Transformation Extender for Red Hat OpenShift.

Runtime Server Container

This section describes the IBM Sterling Transformation Extender Runtime Server Container.

- [Introduction](#)
- [Installing ITX Runtime Server](#)
- [ibm-itx-rs Container Image](#)
- [Packs](#)
- [Healthcare](#)
- [Supply Chain EDI](#)
- [SAP Applications](#)
- [Pack for Financial Payments \(FINPAY\)](#)
- [Pack for Financial Payments Plus \(FINPLUS\)](#)
- [IBM MQ Client](#)
- [Helm Chart](#)
- [Bulk copy](#)
- [Deployment Guidelines](#)
- [Auditing](#)
- [Scaling](#)
- [Uninstalling](#)
- [Upgrades and Rollbacks](#)
- [Limitations](#)
- [Migration](#)

Introduction

IBM Sterling Transformation Extender for Red Hat OpenShift, formerly known as IBM Sterling Transformation Extender Runtime Server, is a packaging of the IBM Sterling Transformation Extender (ITX) runtime product in the form of a Linux container image. The container image, ibm-itx-rs, is an IBM distributed container image that exposes REST API endpoints to deploy and invoke ITX compiled maps and packaged flows.

This document describes deploying the container to a Red Hat OpenShift cluster through a Helm Chart. It also describes using the image to run in a container runtime such as Docker or Podman.

This document supplements the information provided in the CASE (Container Application Software for Enterprises) bundle documentation for ITX Runtime Server (ITX-RS).

Information about Industry and Enterprise Packs distributed by the product and setting up those Packs for the ITX Runtime Server has been provided for Packs users.

For additional high-level overview of the ITX Runtime Server product, see [IBM Support page](#).

The following is a list of high-level features provided by the ITX Runtime Server:

- Uploading, listing, downloading, and deleting compiled ITX maps and other files.
- Listing and downloading generated REST API server and map execution engine log files.
- Fenced (REST API server and map execution engine running in separate processes) and unfenced (REST API server and map execution engine running in same process) run modes.
- Synchronous and asynchronous map executions.
- Cataloging support for the maps, with the REST API endpoints automatically generated for the produced map catalog entries.
- Direct map execution and execution via cataloged map REST APIs.
- Ability to override map inputs and outputs with the REST API request and response data.

In addition to the previous features, which are specific to Version 1 (V1) of the ITX Runtime REST API, the following is a list of additional capabilities that are provided by the ITX Runtime Server:

- Integration with Design Server. Using the Design Server graphical interface, both maps and flows can be deployed to a Runtime Server installation and monitored.
- Flow and map executions are supported by using Version 2 of the ITX Runtime REST API.
- The watch functionality of flows allows listeners to be deployed to a Runtime Server.
- Automated map and flow deployments are available via Cloud Object Storage (COS).
- Synchronous and asynchronous map and flow executions.
- Horizontal pod autoscaling with support for custom metrics and behavior policies.

Installing ITX Runtime Server

- [Prerequisites](#)
- [System Requirements](#)
- [Installing ITX Runtime Server from Command Line](#)

- [Configuration](#)
 - [Storage](#)
 - [SecurityContextConstraints Requirements](#)
-

Prerequisites

ITX Runtime Server requires Red Hat OpenShift cluster version 4.14. It is supported on Linux 64 clusters only. ITX maps can be designed and compiled for the Linux 64 platform by using either ITX Design Studio or Design Server. ITX Design Studio and Design Server are available as separate download options and must be installed locally. Design Studio is only supported on Windows. Design Server is supported on Linux and Windows. Refer to the information provided with your entitlement for instructions to download and install ITX Design Studio and Design Server. Refer to the [ITX Documentation](#) for startup instructions and technical guidelines on using both ITX Design Studio and Design Server. For more information about the product release, see [IBM Sterling Transformation Extender for Red Hat OpenShift 11.0.1](#).

When compiling maps in ITX Design Studio, you must choose the option to compile them for Linux 64 platform. Alternatively, you can use multi-platform composite maps, which will increase the size of the map while enabling the same map to run on both Windows and Linux platforms.

A Redis installation is required to use the ITX Runtime Server in fenced mode, to invoke REST APIs for asynchronous map invocations, or to use the V2 REST API. Flow execution is part of the V2 REST API. For more information, see the [Redis Configuration](#) section.

Two filesystem-based persistent volumes must be provisioned for the ITX Runtime Server to be operational. For a Kubernetes installation, if you plan for the ITX Runtime Server pods to be distributed across multiple nodes, the volumes must support the ReadWriteMany access mode option so that they can be bound by all pods in the installation. For more information, see the [Storage](#) section.

In those cases where only maps need to be run, the two file-based persistent volumes are not required as long as S3 or GCP Cloud Object Storage (COS) is accessible. The ITX Runtime Server can leverage COS for both V1-based map executions and V2-based flow executions. When running V1-based map executions, however, the deployment no longer requires these persistent volumes. By decoupling from persistent volumes, the REST service can become highly available and fault tolerant without any dependencies on one single file system.

System Requirements

- When configuring an ITX Runtime Server instance, specify the requested and the maximum Memory and CPU size, as well as the storage capacity.

The following requested values are selected by default:

Memory (Gi)	CPU (milliscores)	Disk (Gi)	Nodes
4	1000	10	1

- Note that if you choose to run an instance of the ITX Runtime Server in **fenced** mode, the REST service process will run separately from the map execution worker process. Therefore, two Java Virtual Machine (JVM) processes will be created, which should be taken into account when planning system resources, especially memory.
- The storage class that you choose to use for persistent volumes must be of filesystem type.
- Following table shows different configuration profiles, the minimum configuration advisable to set up for the ITX Runtime Server:

Profile	Memory (Gi)	CPU (milliscores)	Disk (Gi)	Nodes
Starter	4	1000	10	1 (master and worker node)
Development	4	2000	25	3 (1 master, 2 worker nodes)
Production	16	4000	50	5 (2 master, 3 worker nodes)

Adjust the above suggested configuration based on your transformation needs, by updating the **resources** and **persistence** sections in `values.yaml`.

Installing ITX Runtime Server from Command Line

ITX Runtime Server can be installed in an online (connected) cluster or offline (air-gapped) cluster using command line tools.

- Download and install **cloudctl**, **oc**, and **helm** command line tools. In case of offline installation, download the tools to the bastion server (if using bastion), or to the portable device (if not using bastion). The CASE needs to be saved to the same machine to which the tools are downloaded.
 - **oc**: To interact with the OpenShift Cluster
 - **cloudctl**: To interact with the CASE
 - **helm**: To install and configure the embedded chart
- Open a Linux shell on the machine with the downloaded tools and the CASE.
- Go to the directory containing the **case** directory for the CASE. All **cloudctl** commands must be run from that directory.

Note: **cloudctl case** usage is deprecated. **cloudctl case** usage when discontinued should be replaced with **oc ibm_pak** in the below command lines after **ibm_pak** plugin for **oc** command line tool has been installed.

- [Installing in an Online Cluster through CASE](#)
- [Installing in an Online Cluster through Native helm CLI Commands](#)
- [Installing in Air-Gapped \(Offline\) OpenShift Cluster](#)
- [Verifying Installation](#)

Installing in an Online Cluster through CASE

Run the following command to install the catalog source to the ibm-itx namespace in your cluster. You may choose a namespace different from ibm-itx.

```
cloudctl case launch \
  --case case/ibm-itx-rs \
  --namespace ibm-itx \
  --inventory itxRsProd \
  --args "-accept unknown" \
  --action install
```

Use the --args "" parameter to pass additional arguments to the helm installer. For example, --args "--set persistence.data.capacity=40Gi" expands the default size of 10Gi specified in values.yaml.

Also, the -accept value must be passed in the --args string once you have read and agreed to the product licensing. The product will not operate until this value is set to true.

Installing in an Online Cluster through Native helm CLI Commands

Run the following command, after replacing the <target_namespace> placeholder with the name of the namespace to which you wish to install the Helm chart. The v11 designation prefixes the release name of the ITX-RS instance. You can change this to another value that better describes your installation.

```
helm install v11 \
  --namespace <target_namespace> \
  --set license=false \
  charts/ibm-itx-rs-prod
```

Note: You must read and accept the product licensing terms and then change --set license=false in the above command line to --set license=true. The product will not operate until the value of the license setting is set to true.

Installing in Air-Gapped (Offline) OpenShift Cluster

- [Prepare Bastion Host or Portable Device](#)
- [Prepare CASE](#)
- [Configure Registry Auth](#)
The following `cloudctl case launch` commands all include <target_namespace> argument. The `ConfigureCredsAirgap`, `mirror-images`, and `ConfigureClusterAirgap` actions do not operate on the target cluster and this argument can be omitted.
- [Mirror Images](#)
- [Configure Cluster for Airgap](#)
- [Install ITX definitions to the cluster](#)

Prepare Bastion Host or Portable Device

Log on to the bastion machine, and verify that it has access to:

- the public Internet (to download images from the source image registry)
- the target image registry (where the images will be mirrored)
- the target OpenShift cluster, to install the Helm chart

If a bastion machine is not available, you can use a portable device (such as laptop) instead, which you must be able to carry into the air-gapped environment to connect to the target cluster.

The remaining steps should be run from the bastion machine (portable device).

Open Linux shell and go to the directory that contains the `case` folder for this CASE. All the remaining `cloudctl` commands must be run from this directory.

Prepare CASE

Procedure

1. Prepare the CASE files used by the process. Create a temporary directory used during the process:

```
$ mkdir /tmp/cases
```

2. Run the following command:

```
$ cloudctl case save --case case/ibm-itx-rs --outputdir /tmp/cases Downloading and extracting the CASE ...
Success
Retrieving CASE version ...
Success
Validating the CASE ...
Success
Creating inventory ...
Success
Finding inventory items
Success
Resolving inventory items ...
```

```
Parsing inventory items
Success
```

A set of .tgz and .csv files will be created under the /tmp/cases folder.

Configure Registry Auth

The following **cloudctl case launch** commands all include <target_namespace> argument. The **ConfigureCredsAirgap**, **mirror-images**, and **ConfigureClusterAirgap** actions do not operate on the target cluster and this argument can be omitted.

Procedure

1. Authenticate with the source image registry (Entitled Registry).
2. Run the following command after replacing the <target_namespace> placeholder with the name of the namespace to which you wish the secret to be created, and after replacing <password> placeholder with the password (API key) you have received with your entitlement.

```
$ cloudctl case launch \
  --case case/ibm-itx-rs \
  --namespace <target_namespace> \
  --inventory itxRsProd \
  --action configureCredsAirgap \
  --args "--registry cp.icr.io --user <user> --pass <password>"
```

The credentials will be saved to ~/.airgap/secrets/cp.icr.io.json

3. Authenticate with the target image registry.

Note: This step can be skipped if the target registry does not require authentication.

4. Run the following command, after replacing the <target_namespace> placeholder with the name of the namespace to which you wish the secret to be created, and after replacing <target_registry>, <user> and <password> placeholders with the registry, user and password for the target image registry.

```
$ cloudctl case launch \
  --case case/ibm-itx-rs \
  --namespace <target_namespace> \
  --inventory itxRsProd \
  --action configureCredsAirgap \
  --args "--registry <target_registry> --user <user> --pass <password>"
```

The credentials will be saved to ~/.airgap/secrets/<target_registry>.json

Mirror Images

In this step, the container image for ITX Runtime Server is copied to the target registry in the air-gap environment. This generates calls to the **oc image mirror** command which copies the images over. Run the following command after replacing the <target_namespace> placeholder with the name of the target namespace for the image, and after replacing the <target_registry> placeholder with the name of the target image registry.

```
$ cloudctl case launch \
  --case case/ibm-itx-rs \
  --namespace <target_namespace> \
  --inventory itxRsProd \
  --action mirror-images \
  --args "--registry <target_registry> --inputDir /tmp/cases"
```

Configure Cluster for Airgap

If the portable device was used instead of the bastion server, carry the device to the air-gapped environment with the target OpenShift cluster, and make sure it has connection to the cluster.

This step does the following:

- Creates a global image pull **Secret** for the target registry (skipped if target registry is unauthenticated).
- Creates an **ImageSourceContentPolicy** object with mirror settings.

Warning: Cluster resources must adjust to the new pull secret, which can temporarily limit the usability of the cluster. Authorization credentials are stored under \$HOME/.airgap/secrets and /tmp/airgap* to support this action.

Warning: Applying ImageSourceContentPolicy causes cluster nodes to recycle. Run the following command after replacing the <target_namespace> placeholder with the name of the target namespace for the Helm chart, and after replacing the <target_registry> placeholder with the name of the target image registry.

```
cloudctl case launch \
  --case case/ibm-itx-rs \
  --namespace <target_namespace> \
  --inventory itxRsProd \
  --action ConfigureClusterAirgap \
  --args "--registry <target_registry> --inputDir /tmp/cases"
```

Install ITX definitions to the cluster

Follow the instructions in **Installing in an Online Cluster through CASE** section, or the instructions in **Installing in an Online Cluster through Native helm CLI Commands** section.

With either method, provide the `--set image.repository=<target_registry>` command argument.

Verifying Installation

Once you have installed the ITX Runtime Server, you can test accessing the instance by invoking the version and configuration REST API endpoints, after replacing `<location>` with the location of the **Route** resource deployed with the instance:

`<location>/tx-rest/v1/itx/version`

`<location>/tx-rest/v1/itx/configuration`

The version endpoint returns a JSON document with information about the product name and version. The configuration endpoint returns a JSON document with the configuration options and values that you specified when you deployed the instance, which you can consider as additional confirmation that the instance was configured as expected.

Configuration

This section describes the configuration parameters of the Runtime Server when Helm is used for installation. An alternate configuration method is used for an installation of the Runtime Server as a single container in a container runtime such as Docker or Podman. See the section on the `ibm-itx-rs` container image for information about configuring a single-container installation.

When Helm is used for installation, the Runtime Server supports two installation modes as follows:

- V1 installation mode
- V2 installation mode

The V1 mode only supports the V1 API, which is enabled by the `restV1.deploy` parameter. The V2 mode supports the V2 and V1 APIs, both of which are activated by the `rest.deploy` parameter. The V2 mode requires an external Redis installation because the V2 API requires map and flow execution requests and results to be stored in Redis. The V1 mode does not require an external Redis installation if the server is unfenced. The V1 mode can be used simultaneously with the V2 mode only if the V1 installation is unfenced (`restV1.runMode`).

The V1 mode supports scaling the number of REST servers (`restV1.replicas`). The V2 mode uses an alternate form of scaling where a single REST server provides messages to one or more execution pods (`executor.replicas`). Horizontal pod autoscaling (HPA) is supported for both modes (`restV1.autoscaling.enabled`, `executor.autoscaling.enabled`).

The configuration settings of the Helm chart `values.yaml` file for the Runtime Server can be broken down into the following logical groups:

- REST settings (V1 or V2 API installation, image details, service account, custom environment variables, replica count)
- Probe settings (liveness, readiness)
- Map and Flow settings (map threads, file extension, etc.)
- Logging (log severity, log name uniqueness, log targets, etc.)
- Resource utilization (memory, CPU, storage, replica count)
- Redis configuration (host, port, password, etc.)
- Storage settings (storage class, capacity, dynamic vs. static binding, etc.)
- TLS configuration (certificates, keys, etc.)
- Cloud Object Storage settings

The following configuration settings are a subset of the total settings available to the ITX Runtime Server. For more information, see the installation's `values.yaml` and `README.md` files. The config section of the `values.yaml` file enables the ITX configuration settings in the `config.yaml` file to be overridden. Each `config.yaml` setting is referenced under the config section according to camel case convention, whereby the first letter of a named setting is lowercase while any remaining words in the name start with a capital letter.

Whenever changes are made to the below properties during a Helm upgrade, the pods that are associated with the existing installation will not automatically restart - unless the change affects the actual deployment manifest. For example, an update to the image name or resource metrics will change the deployment manifest of the installation, which in turn will trigger an automatic restart of the pods. However, a basic change to a property that is linked to the ITX `config.yaml` file, such as the logging level, will not trigger an automatic restart. The restart would need to take place manually, as provided by the Kubernetes scale command.

values.yaml property	Description
Licensing	
license	Switch with true/false values for accepting the terms and conditions of the license agreement. The default is false.
REST V1 - Map Executions via Version 1 of the REST API	
restV1.deploy	Deploy the REST V1-based service, which only supports map executions. The default is true.
restV1.replicas	Number of ITX Runtime Server replica pods to deploy and run. The default is 1.
restV1.runMode	Selection of mode (fenced/unfenced). In fenced mode the REST service process and the worker map execution process are separate processes, and in unfenced mode they run in the same process. The default is unfenced.
restV1.serviceAccount.existingName	Name of the ServiceAccount object to use for the installation. If left empty, the default service account is used.
restV1.extraEnvConfigMap	Name of an optional ConfigMap containing key/value pairs to set as environment variables when creating the pods.
restV1.nodeSelector	Name and value of the label of the worker nodes to which the pods must be deployed.
restV1.affinity	The node affinity and anti-affinity preferences when selecting a node for a pod.
REST V2 - Map and Flow Executions via Version 2 of the REST API	
rest.deploy	Deploy the REST V2-based service, which supports map and flow executions. The default is false.

values.yaml property	Description
executor.replicas	Number of ITX Runtime Server replica pods to deploy and run. The default is 1.
rest(existingName	Name of the ServiceAccount object to use for the installation. If left empty, the default service account is used.
rest.extraEnvConfigMap	Name of an optional ConfigMap containing keys/ value to set as environment variable names/values when creating the pods.
rest.nodeSelector	Name and value of the label of the worker nodes to which the pods must be deployed.
rest.affinity	The node affinity and anti-affinity preferences when selecting a node for a pod.
Image Settings	
rest.image.repository	Image registry and repository location.
rest.image.digest	Image digest value, which takes priority over image tag value.
rest.image.tag	Image tag value.
rest.image.pullPolicy	Image pull policy, one of Always, Never and IfNotPresent.
itxImagePullSecret	Name of the pull image Secret object with which to configure the ServiceAccount for pulling images.
Probes Settings	
restV1 rest executor.probes.liveness.enabled	Switch with true/false values to enable/disable liveness probe. The default is true.
restV1 rest executor.probes.liveness.initialDelaySeconds	Number of seconds to wait before making the initial liveness probe call. The default is 35.
restV1 rest executor.probes.liveness.periodSeconds	Number of seconds between liveness probe calls. The default is 20.
restV1 rest executor.probes.readiness.enabled	Switch with true/false values to enable/disable readiness probe. The default is true.
restV1 rest executor.probes.readiness.initialDelaySeconds	Number of seconds to wait before making the initial readiness probe call. The default is 35.
restV1 rest executor.probes.readiness.periodSeconds	Number of seconds between readiness probe calls. The default is 20.
Map and Flow Run Settings	
rest.map.fileExtension	File extension for compiled maps. The default is mmc.
config.rest.map.uploadTime	Time in minutes to keep loaded maps in memory. The default is 1.
config.rest.resources.mapThreads	Maximum number of threads to spawn for running maps concurrently. The default is 10.
config.rest.resources.flowThreads	Maximum number of threads to spawn for running flows concurrently. The default is 10.
restV1 rest.synchronousTimeout	Synchronous map and flow execution timeout in seconds. The default is 300.
config.runtime.runMaps.runMapMaxCacheNum	Controls run map caching in the Runtime Server. The default is -1, which disables run map caching.
config.rest.multipartbody.allAttachment	Switch to include/exclude all attachments in the multi-part body response. The default is false.
Logging	
config.rest.logging.level	Severity level for REST and map/flow execution logs (ALL,TRACE,INFO,ERROR,NONE). The default is ERROR.
config.rest.logging.rotation.fileAge	Number of days to keep logs. The default is 30.
config.rest.logging.rotation.fileCount	Number of log files. The default is 5.
config.rest.logging.rotation.fileSize	Size in bytes of each log file. The default is 20000.
config.runtime.trace	Switch with "stdout" and "" values to enable logging to Kubernetes versus files located in the /data folder. The default is "stdout".
config.runtime.jniLayerTrace.level	Severity level for the Java Native Interface (JNI) component (10=Critical, 20=Error, 30=Warning, 40=Info, 50=Verbose). The default is 20.
config.rest.logging.addWebServerConsoleLogging	A boolean-valued property which, when true, causes web server logs to be sent to the container stderr stream. The default is true.
Resource Registry	
config.rest.resourceRegistryFile	Full path, starting with /data, of the optional ITX resource registry file, if it was previously uploaded or mounted via a Kubernetes ConfigMap.
Redis Settings	
itxRedis.host	Redis host name. Must be a resolvable name on the current network. The default is redis-master.
itxRedis.port	Redis port number. The default is 6379.
itxRedis.password.secret	Name of the Secret object containing the Redis password, when accessing a password-enabled Redis service.
itxRedis.password.key	The key in the Secret object that contains the redis password.
itxRedis.tls	A mapping which contains keys related to client TLS configuration. Used when a Redis server requires TLS.
restV1.redisStem	Key prefix to use for Redis V1 keys produced and consumed by the server. The default is tx-rest-v1.
Resource Constraints	
restV1 rest executor.resources.requests.cpu	Requested number of CPU cores or millicores (using m suffix). The default is 250m.
restV1 rest executor.resources.requests.memory	Requested memory in bytes (with optional base-10 suffix like M or G, or base-2 suffix, like Mi or Gi. The default is 700mi.
restV1 rest executor.resources.limits.cpu	Upper limit number of CPU cores or millicores (using m suffix). The default is 4000m.
restV1 rest executor.resources.limits.memory	Upper limit for memory in bytes (with optional base-10 or base-2 suffix, like Gi for gigabytes). The default is 8Gi, except for the executor setting which is 4Gi.
Persistent Volumes	
rest.persistence.data logs.size	Capacity to request for data persistent volume. Base-10 suffixes like M and G, or base-2 suffixes like Mi and Gi can be used. The default is 20Gi for data and 100Mi for logs.
rest.persistence.data logs.accessMode	Access mode to specify for the data/logs persistent volume. It can be set to ReadWriteMany or ReadWriteOnce. The default is ReadWriteOnce.

values.yaml property	Description
rest.persistence.data logs.useDynamicProvisioning	Switch with true/false values to enable/disable dynamic provisioning for the data/logs persistent volume. The default is false.
rest.persistence.data logs.storageClassName	Name of the storage class supported by the cluster to use for dynamic provisioning the data/logs persistent volumes.
rest.persistence.data logs.enabled	Switch with true/false to enable/disable persistence. Consider setting it to true if COS is used with REST V1 synchronous, “unfenced” deployments. The default is false.
TLS Settings	
restV1 rest.inbound.https.enabled	Switch with true/false values to enable/disable use of HTTPS protocol on the ITX Runtime Server REST pods. The default is false.
restV1 rest.inbound.https.serviceServingCertificates	Switch with true/false values to enable/disable OpenShift service serving certificates for securing service traffic. The default is false.
restV1 rest.inbound.https.clientAuth	Switch with true/false values to enable/disable validation of certificates presented by the REST API callers (mutual authentication). The default is false.
restV1 rest.inbound.https.secret	Name of the Secret object containing CA certificate, Server certificate and Server private key, in PEM format.
Service Settings	
restV1 rest.service.type	Type of the network service to provision, one of ClusterIP, NodePort, LoadBalancer or ExternalName. Defaults to ClusterIP.
restV1 rest.service.port.http	Port to expose for the HTTP traffic, when SSL is not enabled. The default is 8080.
restV1 rest.service.port.https	Port to expose for the HTTPS traffic, when SSL is enabled. The default is 8443.
Cloud Object Storage (COS) Settings	
external.cos.enabled	Enables maps and flows to get downloaded from COS. The default is false.
external.cos.name	Name of ZIP object to download and unzip. The default is maps.zip.
external.cos.bucket	Name of bucket that contains the ZIP object.
external.cos.targetDir	Target directory that ZIP file gets extracted to. The default is /data/maps.
external.cos.platform	Set to s3 for S3-based platform or gcp for Google-based platform. The default is s3.
external.cos.s3.accessKey secretKey region endpoint	Credentials and location information for S3-based platform. The accessKey and secretKey are the keys of the secret, as specified under Secret Settings.
external.cos.gcp.cf	The key of the secret which contains the Credentials file, as specified under Secret Settings.
Secret Settings	
external.secrets[].name data	An array of secret names and keys, which are mounted under the /xdata/sec folder with a filename that uses the value of the data setting.
external.mq.secret	The secret that contains the MQ channel definition table, which is mounted by default under the /xdata/sec folder.
ConfigMap Settings	
external.configMaps[].name key path	An array of ConfigMap objects that get mounted by default under the /xdata/cfg folder.
Auto Scaling Settings	
restV1 executor.autoscaling.enabled	Switch with true/false values to enable/disable auto scaling of ITX Runtime Server pods. The default is false.
restV1 executor.autoscaling.maxReplicas	Maximum number of ITX Runtime Server pods to auto scale.
restV1 executor.autoscaling.minReplicas	Minimum number of ITX Runtime Server pods with which to start.
restV1 executor.autoscaling.cpu.averageUtilization	Threshold CPU utilization percentage to scale a new ITX Runtime Server pod.
restV1 executor.autoscaling.memory.averageUtilization	Threshold memory utilization percentage to scale a new ITX Runtime Server pod.
restV1 executor.autoscaling.custom.enabled	Switch with true/false values to enable/disable custom metrics. The default is false.
restV1 executor.autoscaling.custom.types[]	An array of custom metric type definitions, as defined by Kubernetes v2 API version.
restV1 executor.autoscaling.custom.behavior	Behavior policies for scaling up and scaling down the deployment, as defined by the Kubernetes v2 API version.

- [Redis Configuration](#)
- [SSL/TLS Configuration](#)

Redis Configuration

If you are planning to run ITX Runtime Server instances in fenced mode (which includes the V2 mode) or run maps asynchronously, you must install Redis prior to installing ITX Runtime Server.

A Redis server installation is not included and must be obtained separately. Ensure that the Redis server which you install comes from a reputable source that you trust. Redis server version 7.0.11 was validated with this version of ITX Runtime Server. This version or later is the recommended version to use. Older Redis server versions may not function as expected when combined with this product and may need to be upgraded before they can be used.

The installed Redis server must be accessible to all ITX Runtime Server pods, at the same host URL and port, which means it must be accessible to all worker nodes to which ITX Runtime Server pods are deployed.

If the selected Redis server installation requires a password for authentication, you must create a Kubernetes Secret object with the password to use for connecting to Redis. The name of the secret is given by `itxRedis.password.secret`. The key within the secret which contains the password is given by `itxRedis.password.key`. See the documentation of your Kubernetes platform for information on creating Secret objects and configuring secure cluster secret storage.

Communication with a Redis server which uses TLS is supported. Both mutual and non-mutual TLS are supported. TLS is enabled by setting `itxRedis.tls.enabled` to true. The configuration of the external Redis server determines if mutual authentication is required.

For non-mutual and mutual TLS, a Kubernetes ConfigMap must be used to provide the trust store that is used by the ITX Redis clients to authenticate the Redis server. The name of the ConfigMap is given by `itxRedis.tls.clientCaConfigMap`. The key within the ConfigMap which provides a PEM trust store is given by `itxRedis.tls.certCafilename`. See the documentation of your Kubernetes platform for information on creating ConfigMap objects.

For mutual TLS, a Kubernetes Secret must be used to provide a client certificate and private key. The name of the Secret is given by `itxRedis.tls.clientSecret`. The key which provides the client certificate PEM file is given by `itxRedis.tls.certFilename`. The key which provides the client private key PEM file is given by `itxRedis.tls.certKeyFilename`.

SSL/TLS Configuration

Two options are available for enabling SSL/TLS communication (HTTPS) with the ITX Runtime Server.

With either option, the Runtime Server can be configured to perform client authentication. This is enabled by setting `restV1.inbound.https.clientAuth` or `rest.inbound.https.clientAuth` to true for the V1 and V2 deployment modes, respectively. Client authentication is disabled by default.

- With the first option, a Kubernetes Secret object needs to be provided. This object contains the server certificate and the server certificate key for the Runtime Server. Both are in PEM format. If you wish to enable client authentication, then you also need to provide a CA certificate file, again in PEM format, that can be used to authenticate clients.

See the mappings `restV1.inbound.https` and `rest.inbound.https` for HTTPS TLS configuration. The keys that you use in the Secret object for certificate authority, server certificate and server key must be named `ca.crt`, `tls.crt`, and `tls.key` respectively.

You can define and manage the Secret object directly and deploy it to the namespace prior to installing ITX Runtime Server, or you can use a cert-manager tool in your cluster to generate the secret automatically and to manage the certificates and keys stored in the secret, including their expiry and rotation. The name of the secret needs to be provided when deploying ITX Runtime Server.

- With the second option, the built-in Service Serving Certificates feature of OpenShift is utilized to provide a server certificate, server private key, and a CA trust store. Use of the Service Serving Certificates feature is indicated by setting `restV1.inbound.https.serviceServingCertificates` or `rest.inbound.https.serviceServingCertificates` to true for the V1 and V2 deployment modes, respectively.

When the Runtime Server is deployed, a Secret object is created for a Service whose backend uses the Service Serving Certificates feature. For example, if the Service Serving Certificates feature is used for the V2 deployment option, a Secret is created for the Service which exposes the HTTPS port of the REST server for the V2 deployment option. Naming and use of this Secret is handled automatically.

A ConfigMap object is automatically created which contains a CA certificate for the Service Serving Certificates CA. The CA certificate of this ConfigMap is used when the ITX Runtime Server performs client authentication. In other words, use of the Service Serving Certificates feature together with client authentication implies that only clients who can be authenticated relative to the Service Serving Certificates CA will be able to access the Runtime Server over HTTPS.

For more information about the OpenShift Service Serving Certificates feature, see [OpenShift documentation](#).

In both the V1 and V2 modes, HTTP connections are enabled by default. To disable HTTP connections, set `restV1.inbound.http.enabled` or `rest.inbound.http.enabled` to false, respectively.

- [**Option 1 - Produce Secret with ca.crt, tls.crt and tls.key PEM values**](#)

Option 1 - Produce Secret with ca.crt, tls.crt and tls.key PEM values

If you have CA certificate, server certificate, and server key values in PEM format, you can create **Secret** object manually and deploy it to the namespace where you plan to install ITX Runtime Server. Note that if you choose to manually create **Secret** object, you assume responsibility for its lifecycle, including management of the key and certificate values they store, and restarting the pods to pick up any updates made to the Secret content.

The following is an example of a **Secret** object with the name `itx-rs-ssl-secret`. Before applying it, replace the three indicated base64 value placeholders with the base64 encoded PEM certificates and keys:

```
apiVersion: v1
kind: Secret
metadata:
  name: itx-rs-ssl-secret
  type: Opaque
data:
  ca.crt: "base64_ca_cert"
  tls.crt: "base64_server_cert"
  tls.key: "base64_server_key"
```

To base64 encode a PEM file and display the value in a single line for inclusion in this manifest, you can use the following command:

```
base64 --wrap=0 <file_name>
```

After applying the manifest, if you deployed it from the command line from a yaml file, you may want to delete the file after applying it, since it will contain the server certificate's private key.

In addition to providing a custom **Secret** object, you can also utilize a certificate management tool to automatically create and manage the Secret. For example, IBM Cloud® [Secrets Manager](#) could be used.

Storage

ITX Runtime Server requires two file system based persistent volumes for its default operation:

- data
- logs

The volumes may be provisioned dynamically during the product installation, or created and installed prior to installing the product, as supported by your cluster.

If you wish to use static volume binding and you wish to initialize the data volume with all the maps you want to make available to the ITX Runtime Server pods immediately upon the installation, note that you must organize the file directories in the volume as shown here:

/data/maps - directory for compiled maps

/data/catalog - directory for map catalog

/data/config - directory for configuration files

/data/extras - directory for any additional JARs and shared libraries

/data/tmp - directory for temporary files used by maps

- [Access Modes](#)
- [Cloud Object Storage](#)
- [Extra Content](#)
- [Backup](#)
- [Encryption](#)

Access Modes

Two access modes are supported:

- ReadWriteMany (RWX)
- ReadWriteOnce (RWO)

The default access mode is ReadWriteOnce.

Since all ITX Runtime Server pods are considered equal for handling client requests, they need to be able to process requests that reference the same maps, logs and other data artifacts in the shared data persistent volume. The storage class used for the **data** and **logs** persistent volume must support ReadWriteMany access mode and this mode must be selected in order for the data persistent volume claim to be satisfied and for the data storage to be successfully bound to all running pods.

In cases when all the pods are always deployed to the same worker node, such as the case with development environments, the ReadWriteOnce access mode may be selected, particularly if a storage class in the environment supports this mode and no other storage classes are available that support ReadWriteMany mode.

ITX Runtime Server has been tested with the following storage providers:

- Google Persistent Disk (RWO)
- Rook CephFS (RWO & RWX)
- NFS (RWO & RWX)

ITX Runtime Server supports the following storage provisioning options:

- Dynamic provisioning using a storageClass
- Pre-created PV's (Persistent Volumes)

Cloud Object Storage

ITX Runtime Server can download maps and flows from a Cloud Object Storage (COS) location. The Amazon S3 and GCP APIs are supported (see `external.cos.platform` in `values.yaml`). When deploying a REST V1 or V2-based installation, all the compiled maps and/or packaged flows can be zipped up in a file with a .zip extension and uploaded to COS for subsequent downloading and automatic setup by the REST V1 or REST V2 pods. The automatic download of both maps and flows is done before the REST pod even starts. The REST pod will only become available once all the maps and/or flows are ready to be executed, thereby avoiding any need for manual deployment and streamlining the overall startup process.

In the case of a REST V1-based deployment whereby maps are only invoked in a synchronous and unfenced configuration, the persistent volume requirement is optional. The REST V1 deployment must be configured for unfenced mode, which is the default. Otherwise, a fenced configuration would require incoming REST data to be stored on a shared persistent volume for subsequent processing by one of the executing pods. In the case of unfenced mode, however, each and every pod that is spun up for map execution runs independently of the others - since the maps are loaded in the ephemeral storage location of the running pod. As long as log files are sent to the Kubernetes log and map logs to the REST API invocation when needed, persistent volumes are no longer required and the REST V1-based deployment can be run in a cloud-native capacity that has no single point of failure and a higher degree of availability.

Extra Content

The extra folder is used to host any JAR files, Java classes, or shared libraries that are required for running the maps but that are not included with the product and you will be providing it separately. An example is JDBC and ODBC drivers to be utilized with JDBC and ODBC adapters. Since these components will be loading and executing in the ITX Runtime Server pods, it is critical that you ensure that they come from reputable sources that you trust, and that you validate them before enabling them for use with the ITX Runtime Server.

Backup

If you are using dynamically provisioned persistent volumes for data and logs, be aware that these volumes will be claimed when the ITX Runtime Server instance is installed, but will be recycled after the product has been uninstalled, since at that time the persistent volume claims created by the product when it was installed will be removed as well. For dynamically provisioned persistent volumes, the assumption is that the maps, flows, and associated artifacts will continue to be available in your source systems from which you originally uploaded or deployed them to the ITX Runtime Server environment.

For example, ITX Runtime Server operates on compiled maps (mmc files) which must have all originated from source maps on on-prem systems where they have been produced by compiling the map source definitions (.mms files) with tools like ITX Design Studio. The compiled maps are then uploaded to ITX Runtime Server. In order to be able to upload the maps again if that need arises, they need to continue to remain available at their origin. This is especially critical for the map definitions (.mms files) since they are required to be able to make updates in the future, and can always be compiled again to be uploaded to ITX Runtime Server. Backing up the original storage for those maps, especially the map sources (.mms files) is therefore of utmost importance, and is outside the scope of ITX Runtime Server deployment.

When using persistent volumes, you can choose to configure them with the Retain policy. The content of the persistent volume will then be preserved and can be subsequently retrieved - even after uninstalling ITX Runtime Server. The exact Kubernetes steps for accomplishing this task is documented under the **Storage** section in the README file of the Helm Chart Examples. After configuring the persistent volume with the Retain policy, the **existingClaim** setting in the values.yaml file can be used to identify the persistent volumes, thereby ensuring that deployment of maps and flows need not be repeated after re-installation.

If you need to back up any files in the persistent volumes that were produced by the maps or flows running under ITX Runtime Server, you must download those files individually using the provided data and logs GET APIs. Alternatively, you may choose to backup the persistent volume storage directly, if your cluster environment provides for direct access to the backing storage. Note that the data GET APIs work on subfolders as well, in which case the response will contain the names of all subfolders and files in that folder. This can be utilized to implement scripts which will recursively pull all the files in a given directory.

Encryption

ITX Runtime Server does not itself perform encryption and decryption of the data at rest. To ensure the data in your persistent volumes is encrypted at rest, when defining persistent volume claims for the ITX Runtime Server, you must specify a filesystem-based storage class that is available in your cluster and supports encryption. For some storage classes, the data may be encrypted automatically by their respective storage providers, but in some cases additional manual configuration may be necessary, such as definition and enablement of encryption keys. For more information, see documentation for the storage classes available in your environment.

In those cases where filesystem-based encryption is not available or only needed in select circumstances, the ITX cipher adapter may be used in maps, flows and configuration files to encrypt data that is stored to disk and decrypt the same data that is retrieved from disk. This capability is documented in the ITX Design Studio examples and online help. The data is encrypted with a highly secure symmetric AES-256 cipher. The key that is used for encryption and decryption can be stored as a secret in the cluster. By mounting the secret key via the external.secrets section of the values.yaml file, the data at rest on disk remains securely encrypted.

SecurityContextConstraints Requirements

ITX Runtime Server requires a SecurityContextConstraints resource to be applied to the namespace to which the instance will be deployed. To define and apply it, follow the instructions provided in the [support page](#).

Custom SecurityContextConstraints definition:

```
apiVersion: security.openshift.io/v1
kind: SecurityContextConstraints
metadata:
  annotations:
    kubernetes.io/description: "This policy allows a single, non-root user"
    name: ibm-itx-rs-scc
  allowHostDirVolumePlugin: false
  allowHostIPC: false
  allowHostNetwork: false
  allowHostPID: false
  allowHostPorts: false
  allowPrivilegedContainer: false
  allowPrivilegeEscalation: false
  allowedCapabilities: null
  allowedFlexVolumes: null
  allowedUnsafeSysctls: null
  defaultAddCapabilities: null
  defaultAllowPrivilegeEscalation: false
  forbiddenSysctls:
    - "*"
  fsGroup:
    type: MustRunAs
    ranges:
      - max: 65535
        min: 1
  readOnlyRootFilesystem: false
  requiredDropCapabilities:
    - ALL
  runAsUser:
    type: MustRunAsRange
    uidRangeMin: 1000
    uidRangeMax: 65535
  seccompProfiles:
    - docker/default
  selinuxContext:
    type: RunAsAny
  supplementalGroups:
    type: MustRunAs
```

```

ranges:
- max: 65535
  min: 1
volumes:
- configMap
- downwardAPI
- emptyDir
- persistentVolumeClaim
- projected
- secret
priority: 0

```

From the command line, save the YAML object to a file and run the following command to apply the file content to your namespace.

```
oc apply -f <file_name> -n <namespace_name>
```

ibm-itx-rs Container Image

- [ibm-itx-rs Image and Container](#)
The image is named ibm-itx-rs, where rs stands for Runtime Server. The ibm-itx-rs image can be used to create a container in a container runtime such as Docker or Podman. This is in contrast to the use of the image as the image for a container in a pod in a Kubernetes deployment of the Runtime Server.
- [Configuring a single-container deployment of ibm-itx-rs](#)
- [Example override file](#)
- [ibm-itx-rs REST API](#)
- [ibm-itx-rs Usage Example](#)
The following example demonstrates how to import the ibm-itx-rs image, how to create a running container and how to execute example map OneInOneOut.mmc. This example map takes the input string, converts it to uppercase string and returns that string.
- [ibm-itx-rs V2 Usage Example](#)
The following example demonstrates how to deploy a flow into the ibm-itx-rs container and then invoke it with input overrides. The example Timesheet flow is described further under the Flow section of the Examples Readme, which is located in the examples sub-directory of the ITX Runtime Server Helm chart. The Timesheet flow enables employee work hour information to be extracted from a list of employee timesheets.

ibm-itx-rs Image and Container

The image is named ibm-itx-rs, where rs stands for Runtime Server. The ibm-itx-rs image can be used to create a container in a container runtime such as Docker or Podman. This is in contrast to the use of the image as the image for a container in a pod in a Kubernetes deployment of the Runtime Server.

To design ITX maps to be used by this container, ITX Design Studio or ITX Design Server must be used. To design flows, ITX Design Server must be used.

For the V1 API, the compiled maps must be saved to a volume or a host directory which is bound to the /data/maps location in the container file system.

For the V2 API, packages must be created in Design Server. A package is deployed to an installation of the Runtime Server using the package deployment endpoint of the V2 API.

The image is comprised of the following main components:

- Red Hat Universal Base Image 9 (ubi9 minimal) operating system as the base image
- Tomcat 10 application server
- REST API service running under Tomcat
- ITX runtime for running maps
- IBM MQ Client

The directories which are the values of the rest.persistence mapping in config.yaml are created at runtime if they do not exist.

The following directories have special significance for the Runtime Server:

/data/maps

Contains compiled maps. By default, the maps must have .mmc file extension. If a map references a file relative to the map directory, that file must reside in the directory relative to this directory. Likewise, if a map creates a file without specifying a directory, that file will be created in the map directory. If a map runs other maps, those maps must reside in the same directory, and must be saved under the exact names, including file extension, under which they are referenced in the main map. To ensure that all instances of the container have access to the same maps when those instances are running on different hosts, this location must be accessible on all hosts, for example, it can be an NFS share.

/data/tmp

Contains temporary execution files.

/logs

Contains log files produced by the ITX runtime, Tomcat application server and REST API service deployed to Tomcat.

/data/catalog

Directory used for cataloged maps.

/data/config

Directory used for holding configuration files, for example, key store file for MQ purposes. The configuration of a single-container deployment can be customized with a file at /data/config/config.yaml.

/data/extr

Contains any additional shared libraries and JAR files to make available to the running container, such as for example JDBC driver JARs for use with JDBC adapter.

The above locations must be mounted to volumes, or host machine directories, when creating a container. Alternatively, a single volume or a local host directory that contains maps, tmp and extra sub-directories can be mounted to the /data location in the container and a single volume or host directory for logs that can be mounted to the /logs location in the container.

Configuring a single-container deployment of ibm-itx-rs

A deployment of the ITX Runtime Server as a single container is configured with a YAML file which provides overrides to the default Runtime Server configuration file. This file must be provided at /data/config/config.yaml. It is recommended to separately add this override file to the volume which is used for the /data directory in the container. This file must be readable by the Runtime Server user which is used inside of the container (UID 1001, itxuser). The directory /data/config should be writable by this user. This mechanism is not used with Helm installations of the Runtime Server.

The override file follows the format of the ITX product configuration file config.yaml. Only the properties for which overrides are desired need to be in the override file. Several configuration parameters which are specific to the Runtime Server may be placed in the override file. These are described below.

The environment variables which were used for configuration in previous releases are not supported with one exception. The environment variable **ITX_RS_LICENSE_ACCEPT** must be set with value true to accept the license and use the container image. For example, the Docker command line option **-e ITX_RS_LICENSE_ACCEPT=true** could be used.

The following config.yaml properties are only used with the Runtime Server:

config.yaml property	Description
Logging	
rest.logging.addWebServerConsoleLogging	A boolean property which, when true, causes the REST server to send logging information to the container stderr stream. The default value is false. This property is analogous to the formerly supported environment variables ITX_RS_LOG_SERVICE_LOG_STD_ERR and ITX_RS_LOG_EXEC_LOG_STD_ERR .
rest.logging.disableWebServerAccessLogging	A boolean property which, when true, disables the Runtime Server access log file. The default value is false (access log enabled). This property is analogous to the formerly supported environment variable ITX_RS_LOG_ACCESS_LOG_LEVEL .
rest.logging.useHostname	A boolean property which, when true, causes the hostname to be added to Runtime Server log files. The default value is false for the single-container deployment. This property is analogous to the formerly supported environment variable ITX_RS_LOG_INCLUDE_HOST_IN_LOG_NAME .
SSL/TLS	
rest.inbound.mutualTls	A boolean property which, when true, causes the Runtime Server to authenticate clients. A CA certificate file must be provided in this case. The default value is false. This property is analogous to the formerly supported environment variable ITX_RS_SSL_CLIENT_AUTH .
Redis TLS [valid for single-container deployments only when an external Redis server is used (redis.deploy is false)]	
redis.port	A value of 0 should be used when the external server uses TLS. The value must be non-zero when the external Redis server does not use TLS.
redis.tlsPort	The listening port for TLS connections of the external Redis server. The value must be zero if redis.port is non-zero. By default, TLS is not used.
redis.client.ssl.enabled	A boolean value which, when true, indicates that TLS should be used when connections are made to the Redis server. TLS parameters are ignored if this value is false. The default value is false.
redis.client.ssl.ca	The absolute path in the container of a PEM trust store file which is used to authenticate the Redis server. The default value is empty.
redis.client.ssl.cert	The absolute path in the container of a PEM certificate chain file which will be provided to the Redis server when it performs client authentication. The default value is empty.
redis.client.ssl.key	The absolute path in the container of a PEM private key file which will be used by the client when the Redis server performs client authentication. The default value is empty.
redis.client.ssl.sni	A hostname string which will be used by the client for the TLS SNI extension when the Redis server requires use of this extension. The default value is empty.

Other configuration is provided through the regular ITX configuration options.

For the Runtime Server, HTTP and HTTPS are enabled or disabled with the rest.inbound.ports mapping. A value of 0 disables the scheme. A non-zero value provides the listening port for the scheme. By default, HTTP uses port 8080, and HTTPS is disabled. The tomcat.inbound.ports mapping is not used.

TLS is enabled for the Runtime Server by providing a non-zero value for the rest.inbound.ports.https property as mentioned above. The server certificate is provided as a PEM file at /opt/server/ssl/cert.pem in the container. The server private key is provided as a PEM file at /opt/server/ssl/key.pem in the container. If mutual authentication is used by setting rest.inbound.mutualTls to true, then a CA trust store is provided as a PEM file at /opt/server/ssl/ca.pem in the container. These files must be readable by the Runtime Server user (UID 1001, itxuser).

Authentication is provided in a single-container deployment of the Runtime Server through mutual TLS. The property rest.inbound.authentication.enabled is false for the Runtime Server.

By default, a single-container deployment of the Runtime Server is unfenced (rest.mode). If fenced mode is used, a Redis server will be created within the container by default. An external Redis server is not needed in this case. The default directory in the container for the Redis server database file redis.rdb is /data/catalog.

If an external Redis server is used for a single-container deployment, then communication with the Redis server over TLS is supported. An external Redis server is used if redis.deploy is false. By default, as mentioned above, an internal Redis server is used. Redis TLS parameters are provided in the table above. By default, TLS is not used when an external Redis server is used. The files must be readable by the Runtime Server user. The parameter for the non-TLS port, redis.port, must be 0 when TLS is used. The TLS port of the Redis server is given by redis.tlsPort.

Logging is configured through the properties of the rest.logging mapping such as rest.logging.level and rest.logging.rotation.fileAge. Also, the rest.executor.trace property can be used to send flow executor logging to stdout or stderr instead of being sent to a file.

Example override file

The following YAML content is an example of the content of a YAML file which can be provided at /data/config/config.yaml in a single-container deployment to customize the Runtime Server. These overrides will enable fenced mode, set the logging level to INFO, send web server logs to stderr in addition to file logging, and enable HTTPS on port 8443. In this example, a server certificate and private key would be provided through file mounts in the container as described above.

```
rest:
  mode: "fenced"
  logging:
    level: "INFO"
    addWebServerConsoleLogging: true
  inbound:
    ports:
      https: 8443
```

ibm-itx-rs REST API

The ibm-itx-rs container exposes a Swagger UI web page which documents the supported APIs. This web page can be displayed by pointing your web browser to:

```
http://CONTAINER_HOSTNAME:8080/tx-rest/api-docs?url=/tx-rest/docs
```

Where **CONTAINER_HOSTNAME** is the IP address or hostname assigned to the running container. If the container is running under a cluster environment like Red Hat OpenShift, use ingress or route name of the ITX Runtime Server. Replace the default text v2/docs or v1/itx/docs in the search box with openapi.json and select the Explore button in the Swagger UI web page.

By default, all the deployed map and flow endpoints for V2 REST API are displayed in the Swagger UI web page. If the rest.defaultSwagger property is set to v1, then all cataloged map endpoints for V1 REST API are displayed in the Swagger UI page.

The following REST APIs are supported:

Version

```
GET http://CONTAINER_HOSTNAME:8080/tx-rest/v1/itx/version
```

Where **CONTAINER_HOSTNAME** is the IP address or hostname assigned to the running container.

The return value is JSON in the format:

```
{
  "version": "11.0.1.0",
  "name": "IBM Sterling Transformation Extender for Red Hat OpenShift",
  "up_time": "1 days, 4 hours, 18 minutes, 44 seconds",
  "date": "20240927",
  "revision": "0"
}
```

This API call can be used as a heartbeat mechanism for ensuring the container is up and running.

Configuration

```
GET http://CONTAINER_HOSTNAME:8080/tx-rest/v1/itx/configuration
```

Where **CONTAINER_HOSTNAME** is the IP address or hostname assigned to the running container.

The return value is JSON in the format:

```
{
  "map_dirs": [
    "/data/maps"
  ],
  "work_dir": "/data/tmp",
  "catalog_dir": "/data/catalog",
  "redis_host": "localhost",
  "redis_key_stem": "tx-rest",
  "is_fenced": true,
  "unload_time": 60000,
  "max_map_threads": 10,
  "exec_log_dir": "/logs"
}
```

Direct Map Invocation

```
PUT http://CONTAINER_HOSTNAME:8080/tx-rest/v1/itx/maps/direct/PATH
```

```
POST http://CONTAINER_HOSTNAME:8080/tx-rest/v1/itx/maps/direct/PATH
```

Where **CONTAINER_HOSTNAME** is the IP address or hostname or ingress/route name assigned to the running container and PATH is the path to the compiled map file, relative to the location corresponding to the maps mount point, and with the .mmc file extension omitted.

POST method runs the map asynchronously. Redis is required to be setup to run maps in this mode of execution to find the status of the map execution, fetching outputs, map trace and map audit from the Runtime Server. Environment variables for running map in fenced mode, Redis server host and port options listed in previous page must be passed when launching the Runtime Server container. Fenced mode ensures that an ITX failure does not affect the web server. Fenced mode is required to run a map asynchronously. In unfenced run mode, the servlet runs within the web server.

The following query parameters are supported:

- **input**

The input query parameter is used for overriding source data settings in input cards. Multiple input overrides can be specified by a comma-separated list, or by specifying multiple input parameters. Each value is either an input card number (if the card data is being provided in the request body), or an adapter specification in the form:

```
card-num;adapter-alias;adapter-command
```

For example, to specify that the data for card 2 should be from a file:

```
input=2;FILE;mydata.txt
```

- **output**

The output query parameter is used to specify which output cards should be returned. Multiple output overrides can be specified by a comma-separated list, or by specifying multiple output parameters. Each value is either an output card number (if the card data should be returned in the response body), or an adapter specification in the form:

```
card-num;adapter-alias;adapter-command
```

For example, to specify that the data for card 2 should be written to a file:

```
output=2;FILE;results.txt
```

- **return**

The return query parameter is used to specify the type of information to return about the executed map. Permitted values are **status**, **statusonerror**, **audit**, **auditonerror**, **trace**, and **traceonerror**. Multiple values can be specified, as a comma-separated list in a single return query parameter, or by using multiple return query parameters.

ibm-itx-rs Usage Example

The following example demonstrates how to import the ibm-itx-rs image, how to create a running container and how to execute example map OneInOneOut.mmc. This example map takes the input string, converts it to uppercase string and returns that string.

About this task

To import the ibm-itx-rs image, create a running container and execute example system, complete the following steps:

Procedure

1. Start by logging into the IBM Entitled Registry (cp.icr.io) to pull the ibm-itx-rs image:

```
docker login -u <username> -p <password> cp.icr.io
```

Note: Login credentials, username and password, are provided with the entitlement of the product. Internet access to download the image from the IBM Entitled Registry is a prerequisite for saving the image in the local docker registry or to the local image server.

- a. Pull the docker image by image tag or image digest from the IBM Entitled Registry:

```
docker image pull cp.icr.io/cp/ibm-itx-rs:11.0.1.0.20240927
```

```
docker image pull cp.icr.io/cp/ibm-itx-rs@sha256:9bc80dd8f0cb2db8a35f17f10a11b2eea9b21ec497b3ccae9729fce919475661
```

Note: Image tag and digest change when a refreshed container image is published to the IBM Entitled Registry. Replace proper image tag/digest accordingly in the commands listed in this document.

- b. If you have downloaded the docker image from the IBM Entitled Registry in a docker archive, for example, **ibm-itx-rs_11.0.1.0.20240927.tar.gz**, load the ibm-itx-rs docker image archive to the local docker registry:

```
docker load -i ./ibm-itx-rs_11.0.1.0.20240927.tar.gz
```

- c. List the images to ensure ibm-itx-rs image was imported:

```
docker images
```

The command returns the list of images, which includes image with the name ibm-itx-rs and the tag **11.0.1.0.20240927**.

Note: The **11.0.1.0.20240927** tag value is assigned to the ibm-itx-rs image. It represents the image version and you will use it later to start the container based on this image.

2. Create directories on the host to serve as bind mounts for the container (Docker managed volumes could be used instead of host managed directories as well). For example, as a test, to create them under /tmp on the host system:

```
mkdir -p /tmp/itx-rs/logs
```

```
mkdir -p /tmp/itx-rs/data/maps
```

```
mkdir -p /tmp/itx-rs/data/tmp
```

```
mkdir -p /tmp/itx-rs/data/extra
```

3. Optionally, you can also create a config directory. You can store config.yaml and other runtime configuration files in it if you wish to modify them from the defaults used by the container. To start with the default configuration files deployed with the image, you can perform the following steps to create a temporary container and extract the config directory from it to use it as the starting point:

```
docker create --name itx-rs-tmp ibm-itx-rs:11.0.1.0.20240927
```

```
docker cp itx-rs-tmp:/opt/runtime/config/tmp/
```

```
docker rm itx-rs-tmp
```

- a. You can then edit /tmp/config/config.yaml file and customize the settings in it. Refer to the documentation included in the configuration file for details about the individual settings. No customization is necessary to complete this example.

4. Save the compiled map OneInOneOut.mmc to /tmp/itx-rs/data/maps directory. This map is a simple transformation map that converts the provided string to uppercase. The string is provided in a request and passed to the input card. The result is collected from the output card and returned in the response:

```
cp ./OneInOneOut.lnx /tmp/itx-rs/data/maps
```

5. Run the container by invoking the following command (as one line):

```
docker run --name itx-rs -it -d -p 8080:8080 -v /tmp/config:/opt/runtime/config -v /tmp/itx-rs/data:/data -v /tmp/itx-rs/logs:/logs ibm-itx-rs:11.0.1.0.20240927
```

Note: The **-v /tmp/config:/opt/runtime/config** argument is optional and is needed only if you customized configuration files on the host and have made changes to them that you wish to be used by the runtime in the container.

Note: The **-p 8080:8080** argument is used to bind port 8080 on the local host to port 8080 in the container. This will allow you to submit REST calls via a **curl** command to localhost:8080, which in turn will be served by Tomcat running in the container.

a. Ensure that the itx-rs container is running:

```
docker ps
```

The command should return a list of running containers, which should include a container with name itx-rs.

6. Check that the container is serving traffic by obtaining the version:

```
curl -X GET "http://localhost:8080/tx-rest/v1/itx/version"
```

It returns a JSON document in this format:

```
{"version":"11.0.1.0","name":"IBM Sterling Transformation Extender for Red Hat OpenShift","up_time":"1 days, 4 hours, 34 minutes, 58 seconds","date":"20240927","revision":"0"}
```

7. Run the map synchronously by invoking the following command (as one line):

```
curl -X PUT -d "This is a test" "http://localhost:8080/tx-rest/v1/itx/maps/direct/OneInOneOut?input=1&output=1"
```

It returns the value:
THIS IS A TEST

Note: The map source file OneInOneOut.mms is also provided. If you open it in Design Studio or Design Server, you will observe that it has one input card of File type, reading from file in.txt, and one output card of File type, writing to file out.txt.

Note: In the previous invocation of the map, the input and output cards were overridden to use the request and response data of the REST API call. If you create file in.txt in the same maps directory as the OneInOneOut.mmc compiled map and invoke the map without specifying the overrides, the map will run and produce out.txt file in the same maps directory, with the same text as in.txt file, except it will be changed to uppercase:

```
echo "This is a file test" > /tmp/itx-rs/data/maps/in.txt
curl -X PUT -d "" "http://localhost:8080/tx-rest/v1/itx/maps/direct/OneInOneOut"
cat /tmp/itx-rs/data/maps/out.txt
```

It should display the value:

```
THIS IS A FILE TEST
```

Note: The locations of the files in the provided sample map are specified as relative file names only, which means that they must reside in the same directory as the compiled map. If you specify absolute locations of the files in the maps, note that those locations must be valid locations in the container file system. For example, if you specified **-v /tmp/itx-rs/data:/data** argument when you started the container as in this example, you can store a file somewhere under the /tmp/itx-rs/data directory, and you should refer to it in the maps as if it was under /data directory, since /tmp/itx-rs/data directory in the host file system is mapped to /data directory in the container file system.

8. To run the map asynchronously, the Runtime Server must run in fenced mode and be able to connect to the Redis server to get the execution status of a (long) running map, fetch the map output(s), map trace and audit. The Redis host and port number may have to be configured accordingly and specify the fenced mode when running the Runtime Server instance as following:

```
docker run --name itx-rs -it -d -p 8080:8080 -v /tmp/config:/opt/runtime/config -v /tmp/itx-rs/data:/data -v /tmp/itx-rs/logs:/logs ibm-itx-rs:11.0.1.0.20240927
```

a. Following command returns a JSON response with a link to check the map execution status.

```
curl -X POST -d "This is a test" "http://localhost:8080/tx-rest/v1/itx/maps/direct/OneInOneOut?input=1&output=1"
```

A sample snippet as following:

```
{"id":"5e8fabfc-4da5-4e22-9bff-27db13827f43",
"href":"http://localhost:8080/tx-rest/v1/itx/maps/direct/5e8fabfc-4da5-4e22-9bff-27db13827f43/status"}
```

b. A long running map execution status can then be fetched using the link provided in the JSON response.

```
curl -X GET "http://localhost:8080/tx-rest/v1/itx/maps/direct/5e8fabfc-4da5-4e22-9bff-27db13827f43/status"
```

The map execution status would look something like as shown below with the link(s) for map output(s).

```
{"status":0,"outputs":[{"href":"http://localhost:8080/tx-rest/v1/itx/maps/direct/9690ed3b-0804-4bda-a076-b336bd0cc733/outputs/1","card_number":1,"mime_type":"application/octet-stream"}],"start_timestamp":"2020-12-18T02:41:50.943+0000","elapsed_time":1,"status_message":"Map completed successfully"}
```

c. To fetch the map output from the Runtime Server, the output link in the map execution status JSON response must be utilized as following:

```
curl -X GET "http://localhost:8080/tx-rest/v1/itx/maps/direct/9690ed3b-0804-4bda-a076-b336bd0cc733/outputs/1"
```

It should return the value:

```
THIS IS A TEST
```

9. To perform cleanup, remove the image, container, and local host directories used as bind mounts, run the following commands:

```
docker stop itx-rs
```

```
docker rm itx-rs
docker rmi ibm-itx-rs:11.0.1.0.20240927
rm -rf /tmp/itx-rs
rm -rf /tmp/config
```

- [Cataloged Map Invocation](#)

Cataloging a map simplifies calls to the map and generates Swagger documentation that describes the APIs. The catalog provides a way to expose maps as services and to hide the implementation details of the services.

Cataloged Map Invocation

Cataloging a map simplifies calls to the map and generates Swagger documentation that describes the APIs. The catalog provides a way to expose maps as services and to hide the implementation details of the services.

A map is defined in the catalog as a JSON document, like the 1in1out map example that is defined as following:

```
{
  "name": "1in1out",
  "summary": "Single input & single output",
  "description": "Uppercases the output from a single input",
  "tags": [ "Test" ],
  "inputs": [
    {
      "card_number": 1,
      "mime_type": "text/plain",
      "description": "Plain text"
    }
  ],
  "outputs": [
    {
      "card_number": 1,
      "mime_type": "text/plain",
      "description": "Uppercased input"
    }
  ],
  "url_path": "test/lin1out",
  "map_path": "OneInOneOut"
}
```

The Swagger documentation for the Runtime Server defines the structure of the JSON document and explains the purpose of each element. You register a map in the catalog by posting the JSON document that defines it to the catalog directory. The following commands assume Runtime Server running on localhost (port 8080):

```
curl -X POST "http://localhost:8080/tx-rest/v1/itx/catalog" -H "Content-Type: application/json" -d @1in1out.json
```

Use Swagger to display map details and to test-run maps. To see the Swagger documentation for the catalogued maps, enter the following URL in a browser:

```
http://localhost:8080/tx-rest/api-docs?url=/tx-rest/v1/docs
```

By default, all the deployed map and flow endpoints for V2 REST API are shown in the Swagger web UI page, since the rest.defaultSwagger property defaults to v2. To see the catalogued map endpoints for V1 REST API, replace the text v2/docs in the search box with v1/itx/docs and select the Explore button in the Swagger web UI page.

You can run a catalogued map without specifying which cards to override or the name and path of the calling map. For example, the following command synchronously runs the catalogued version of the OneInOneOut map.

```
curl -X PUT -d "This is a test" "http://localhost:8080/tx-rest/v1/itx/maps/catalog/test/lin1out"
```

It should return the value:

```
THIS IS A TEST
```

ibm-itx-rs V2 Usage Example

The following example demonstrates how to deploy a flow into the ibm-itx-rs container and then invoke it with input overrides. The example Timesheet flow is described further under the Flow section of the Examples Readme, which is located in the examples sub-directory of the ITX Runtime Server Helm chart. The Timesheet flow enables employee work hour information to be extracted from a list of employee timesheets.

About this task

To load the ibm-itx-rs image, create a container, and execute the flow, complete the following steps:

Procedure

1. Load the ibm-itx-rs image into Podman.

```
podman load -i ibm-itx-rs_11.0.1.0.20240927.tar.gz
```

2. Configure the V2 Runtime REST API.

```
echo -e "rest:\n  mode: fenced" >/tmp/config.yaml
```

3. Install the ITX Runtime Server.

```
podman run --name itx-rs-v2 -it -d -p 8080:8080 -e ITX_RS_LICENSE_ACCEPT=true -v /tmp/config.yaml:/data/config/config.yaml  
ibm-itx-rs:11.0.1.0.20240927
```

4. Deploy the flow.

```
curl -X POST "http://localhost:8080/tx-rest/v2/packages" -H "accept: */*" -H "Content-Type: application/octet-stream" --  
data-binary @Timesheet_linux.sqlite
```

5. Run the flow.

```
curl -X PUT -F "1=@allemps1f.txt" -F "2=@employee1.txt" "http://localhost:8080/tx-rest/v2/run/WorkHours"  
curl -X PUT -F "1=@allemps1f.txt" -F "2=@employee2.txt" "http://localhost:8080/tx-rest/v2/run/WorkHours"
```

The first flow invocation returns 50 hours for employee1 while the second returns 90.45 hours for employee2.

Note: The sqlite package and input files are available in the ITX Runtime Server Helm chart under the examples\flow sub-directory. Once the container is stopped, no mapping and logging artifacts are retained because external mount points were not used in the example. If external mount points are used during the **Install**, the **Deploy** need not be run again. The ITX Design Server provides a graphical way to re-deploy changes and monitor flow executions.

Packs

General Notes:

- IBM Sterling Transformation Extender for Red Hat OpenShift must be installed and configured as above, with appropriate permissions provided to any users that will be deploying and executing maps from a Pack.
- The selected Pack (Healthcare, Supply Chain, Financial Payments, SAP) Version for IBM Sterling Transformation Extender for Red Hat OpenShift must be installed on a Windows machine with the existing Transformation Extender Design Studio installed for use of V1 API. For use of V2 API, the specific Pack project to be used should be imported into Design Server.
- As the container is UNIX based, any files transferred as input to a pack map that has Windows-based, line endings (for example, CR/LF) should be converted to UNIX base line endings (LF) before, or during file transfer. Similarly, any paths within maps that are part of the Pack, or that you implement with components of a path, need to use the UNIX expected slash (/) instead of backslash (\) in paths to data files in input or output cards.
- All executable maps must be built for the Linux platform. Design Studio should be used with the container V1-based map executions. The compiled map extension in the container config.yaml is set to .mmc. Using Design Studio to build maps for the Linux target platform gives the map extension .lnx. (see the individual pack tips for details).
- There are several options to synchronize the map extension. Select one of the options below:
 - Modify the container config.yaml and change the setting file Extension mmc to lnx to match the extension with Design Studio compile. The new config.yaml would need to be copied to the <rs_mount> location under /config folder.
 - Modify preferences in Design Studio and select Transformation Extender > Compiled Map Extensions (Linux change to .mmc).
 - Modify preferences in Design Studio and select Transformation Extender > Map > Build Options. Selecting Windows and Linux will build the map for both Windows and Linux (without identifying target platform) and give the file extension .mmc

Docker Tips:

- To restart the container:

```
docker restart itx-rs
```

- Out of space conditions can show unusual errors (for example, java errors and dumps). Try cleaning up the system using the following commands:
 - To clean up unused local volumes, run the command: **docker volume prune**
 - To clean up unused images, run the command: **docker image prune**
 - To clean up stopped containers, run the command: **docker container prune**
 - To clean up all unused containers, networks, images (both dangling and unused), and optionally images, run the command: **docker system prune**

Healthcare

- [HIPAA Compliance Check or HIPAA Data Compliance](#)

To run HIPAA Compliance check or HIPAA Data Compliance, perform the following steps:

- [HL7 Compliance Check](#)
- [Other Healthcare Components](#)

HIPAA Compliance Check or HIPAA Data Compliance

To run HIPAA Compliance check or HIPAA Data Compliance, perform the following steps:

- [Setup for Design Studio V1-based map executions](#)
- [Setup for Design Studio V2-based map executions](#)

Setup for Design Studio V1-based map executions

Procedure

1. Open map compliance_check.mms or hipaa_data_compliance under hipaa/maps.
2. Build all maps for specific platform Linux. All maps will be created with a .lnx extension.
3. With the exception of the compliance_check.lnx or hipaa_data_compliance.lnx, rename all other maps from .lnx to .mmc.
4. Under the container /tmp/itx-rs/data/maps directory, create a directory structure, and copy compiled maps to /maps directory and data files (with UNIX based line endings):


```
/tmp/itx-rs/data/maps/hipaa
/tmp/itx-rs/data/maps/hipaa/maps
/tmp/itx-rs/data/maps/hipaa/data
```
5. Additional steps needed for running Type 5 validation in hipaa data compliance include updates and references to the resource registry files found in hipaa/examples/code_sets.
 - a. Ensure that the host value in the hipaa_code_set.mrn file points to a MongoDB server that is accessible from the container, and that server includes the hipaa_code_sets database with external code set collections/values that are to be validated (these can be loaded with the maps in hipaa/examples/code_sets).
 - b. Copy both files to an accessible directory, such as /tmp/itx-rs/data/maps/code_sets.
 - c. Add the .mrc file to the config.yaml, similar to the following:

```
rest:
  inbound:
    ports:
      http: 8080
      https: 8443
  mode:
    fenced
  resourceRegistryFile: "/data/maps/code_sets/hipaa_code_set.mrc"
```

6. To execute compliance_check through command line:

- a. With the default input files:

```
curl -X PUT -d "" "http://localhost:8080/tx-rest/v1/itx/maps/direct/ hipaa.maps.compliance_check"
```

- b. To override input card 2:

```
curl -X PUT -d "" "http://localhost:8080/tx-rest/v1/itx/maps/direct/ hipaa.maps.compliance_check?
input=2;FILE;../data/inputfile.dat"
```

7. To execute compliance_check using Swagger UI:

- a. In the path field, enter the path: .hipaa.maps.compliance_check
- b. In the input field, specify the input data for card 2 from the file: input=2;FILE;../data/all_hippa_5010_transaction
- c. In the return field, specify audit

Note:

If you need to apply configurable rules to the HIPAA compliance process, be aware that this step can only be done on Windows prior to building the compliance_check maps, and the configurable_rules map does not need to be ported.

Setup for Design Studio V2-based map executions

Procedure

1. Import hipaaCompliance or hipaaDataCompliance.zip
2. Create a package that includes all the run maps, but only need to define an endpoint for the main map: Compliance_check or hipaa_data_compliance
3. In addition to the specific parameters you wish to run with, the compliance_check_5010_parameter.dat or hipaa_data_compliance_parameter.json must include a path for the **Base_Map_Directory**=“..//maps” (dat) or “**BaseMapDirectory**”: “..//maps/” (json)
4. Build and deploy the package to the container.
5. Use the v2/run API to run the map from the endpoint defined through Swagger or Curl command.

HL7 Compliance Check

About this task

To run HL7 Compliance check, perform the following steps:

Procedure

1. Edit file compilemaps.bat under hl7/maps. Replace **.mmc** in the **%MCOMPILER%** lines with .lnx -P LINUX. Make sure the variable “**%DTX_HOME%**” is pointing to the TransformationExtender_11.0.1 directory.
2. Run the compilemaps.bat. Verify .lnx files were created in the hl7/maps directory.
3. Using Design Studio, open maps ncpdp_script_to_hl7.mms and hl7_to_ncpdp_script.mms in the \hl7\examples\hl7_ncpdp_script subdirectory and build all for Linux platform.
4. Under the container /tmp/itx-rs/data/maps directory, create directory structure, and copy compiled maps, schemas and data files to /hl7/maps, /hl7/hl7schemas and /hl7/data subdirectories (with UNIX based line endings).
5. To execute HL7 Compliance, the hl7main map through command line:

```
curl -X PUT -d "" "http://localhost:8080/tx-rest/v1/itx/maps/direct/hl7.maps.hl7main"
```

- a. To override input card 1:

```
curl -X PUT -d "" "http://localhost:8080/tx-rest/v1/itx/maps/direct/hl7.maps.hl7main?
input=1;FILE;../data/sample_hl7_edi_v2_7_adt_a01.dat"
```

Other Healthcare Components

The HIPAA, HL7 and NCPDP passthrough, conversion maps included in the Pack for Healthcare can be run similarly, by building for Linux and making sure any appropriate directory structure is set up under the container maps directory.

If running NCPDP Validation map, the ncpdp_jexit.jar file must be copied from the ncpdp/jars folder to the /data/extra directory.

Supply Chain EDI

- [EDI Compliance check](#)
- [Setup for Design Studio V1-based map executions](#)
Design Studio should be used with the container V1-based map executions. The compiled map extension in the container config.yaml is set to .mmc. Using Design Studio to build maps for the Linux target platform gives the map extension .lnx. The map extension between the config.yaml and the map Design Studio compiled must correspond.
- [Setup for Design Server V2-based flow executions](#)
- [Supply Chain Examples](#)

EDI Compliance check

The main execution maps for the EDI Compliance Checking follow similar naming conventions. For example, component EDIFACT main execution map name is ccdef and with component X12 the main execution map is cx12. By design, Supply Chain compliance checking includes configurations for additional validations, acknowledgments, runtime features, default input and output destinations. The output destinations can be structured for file and card outputs with paths identified in xml or json configuration files requiring the output structure to be defined for the container.

The compiled maps along with input and output folder structures must be saved to the volume, or a host directory, bound to the /data/maps location in the container file system. Configuration files are located (relative to the map location) with the default output file location for the file system defined.

Setup for Design Studio V1-based map executions

Design Studio should be used with the container V1-based map executions. The compiled map extension in the container config.yaml is set to .mmc. Using Design Studio to build maps for the Linux target platform gives the map extension .lnx. The map extension between the config.yaml and the map Design Studio compiled must correspond.

Before you begin

- Build the following maps in Design Studio on Windows, with Linux selected as the target platform, in the <DTXHOME>\packs\supplychain_edi_v10.2.3\x12\compliance\mapsandschemas folder: ccx12, x12segment, x12tval.
- The configuration file is identified on input card three, ConfigInfo, for the main execution map for each pack component. Each pack component uses the default configuration (*defaultconfig.xml or *defaultconfig.json.dat). The x12u and tt-x12 components use the *defaultconfig.json.dat file. With container mount location /tmp/itx-rs/data, the x12defaultconfig.xml file location:

```
/tmp/itx-rs/data/maps/x12/compliance/data/x12defaultconfig.xml
<FileOutput>
    <FileDirectory>.../data/output</FileDirectory>
</FileOutput>
```
- Create the compliance folder structure like the one in Windows. Either copy the entire x12, edifact, or rail folder from the Design Studio pack install (step a.) or create and copy folders and files individually (steps b and c).

Procedure

1. Copy the entire x12 folder to the <rs_mount> location under /maps for example:

```
<rs_mount> = /tmp/itx-rs/data
cp -R x12 /tmp/itx-rs/data/maps
```

Note: Make sure the folders have Write permission. If not, use **chmod +777** to change the permission.
2. Create folders individually <rs_mount> = /tmp/itx-rs/data:

```
mkdir -p /tmp/itx-rs/data/maps/x12/compliance/data/output/x12/reject
mkdir -p /tmp/itx-rs/data/maps/x12/compliance/data/output/fa/inbound
mkdir -p /tmp/itx-rs/data/maps/x12/compliance/data/output/fa/outbound
mkdir -p /tmp/itx-rs/data/maps/x12/compliance/data/output/ta1/inbound
mkdir -p /tmp/itx-rs/data/maps/x12/compliance/data/output/ta1/outbound
mkdir -p /tmp/itx-rs/data/maps/x12/compliance/mapsandschemas
```

Note: Make sure the folders have Write permission. If not, use **chmod +777** to change the permission.
3. Copy files from the Windows mapsandschemas to the Linux mapsandschemas folder:
 - a. Copy all the compiled maps from the Windows mapsandschemas to the Linux mapsandschemas folder.

- b. Copy all the xsd files to the Linux mapsandschemas folder.
 - c. Transfer edicc file to the Linux mapsandschemas folder to ensure CR/LF end of line character is converted to NL character of the target system.
 - d. Copy the files that begin with x12 to the data folder.
 - e. Copy the files that end with xml to the data folder.
-

Setup for Design Server V2-based flow executions

Design Server packages are used to deploy to the V2 based flow executions and are defined using the maps and files in a project. When the package is deployed or compiled, it is saved to the volume, or a host directory, bound to the /data/catalog location in the container file system.

The following information is related to the Design Server project x12Compliance.zip. The main execution map is ccx12. Modify the ccx12 map and input card three, ConfigInfo, to use the x12rsconfig.xml file or the x12fullconfig.xml file. The x12fullconfig.xml file contains additional configuration settings for the compliance checking process. The *config.xml file selected will need to be downloaded, modified, and replaced in the Design Server project. The FileDirectory needs to be updated to follow the container mount location.

```
<FileOutput>
<!-- This section will send output to files. Reference file x12rsconfig.sh -->
  <FileDirectory>/data/data/x12Compliance</FileDirectory>
</FileOutput>
```

The FileDirectory defined in the x12 configuration file should point to the container file system and mount location.

With container mount location /tmp/itx-rs/data and package name x12Compliance.

```
<FileOutput>
  <!-- This section will send output to files. Reference file x12rsconfig.sh -->
    <FileDirectory>/data/x12Compliance</FileDirectory>
</FileOutput>
```

The compliance check output folder structure must also be defined for the container file system and mount location /data.

A script file is located in the x12Compliance project, relative to the map location. The script contains example commands to create the output file structure for the container package deployment to the rest container depending on the server definition in Design Server for deployment.

With container mount location /tmp/itx-rs/data.

```
/tmp/itx-rs/data/maps/x12Compliance/compliance/data/x12rsconfig.sh
```

```
mkdir -p /tmp/itx-rs/data/x12Compliance/x12/reject
```

```
mkdir -p /tmp/itx-rs/data/x12Compliance/fa/inbound
```

```
mkdir -p /tmp/itx-rs/data/x12Compliance/fa/outbound
```

```
mkdir -p /tmp/itx-rs/data/x12Compliance/ta1/inbound
```

```
mkdir -p /tmp/itx-rs/data/x12Compliance/ta1/outbound
```

As an alternative to using the predefined output structure, you can remove the <FileOutput> component from the *config.xml file. This will cause compliance checking to default to card output. Using card output requires that each output card of interest be identified with the map execution. The example below would override output card 5 in the ccx12 map for valid data and card 9 for the x12 acknowledgement.

```
curl -X PUT "http://localhost:8080/tx-rest/v2/run/ccx12?output=5;FILE;/data/x12compliance/x12ValidData.out,
output=9;FILE;/data/x12compliance/x12Ack.out"
```

After deploying the package and modifications, restart the container.

- [Compliance Check Execution](#)

Compliance Check Execution

Procedure

1. Test the map with valid data by issuing this command:

```
curl -X PUT "http://localhost:8080/tx-rest/v2/run/ccx12"
```

If you get a message Map completed successfully, go to /tmp/itx-rs/data/x12Compliance/x12 and locate a file that ends with .in extension. If not, check the map folders to make sure they are created in the correct location.

2. Test the map with invalid data by issuing this command:

```
curl -X PUT "http://localhost:8080/tx-rest/v2/run/ccx12?input=1;FILE;../data/x12_invalid.txt"
```

If you get a message Map completed successfully, go to /tmp/itx-rs/data/x12Compliance/x12/reject folder and locate a file that ends with .log.

3. If you see the following message in the container JSON response, there could be issues with the output file structure.

```
"message": "FAIL function aborted map: Map terminated due to map framework problem"
```

With this situation, the compliance check maps will generate a map framework trace in the mapsandschemas folder. With V2 execution, this is found within the container and will need to be retrieved. One way is to use the docker cp command.

- a. Find the link in the container JSON response. For example, "Settings": "/tmp/6c64-f72a-1350-b233/compliance/mapsandschemas/
- b. Create a folder to copy the files from the container. For example, /home/x12Results
- c. Issue the docker copy command:

```
docker cp itx-rs:/tmp/6c64-f72a-1350-b233 x12Results
```

d. Navigate to the /home/x12Results/6c64-f72a-1350-b233 and the mapsandschemas folder.
e. Open the edilog.xml and search on sendoutput to verify output locations. With the example below the output folder structure /data/x12u/ta1/output is not defined.

```
<SendOutput>
<Send>
    <Command>/data/x12u/ta1/outbound/20240918-192651_ta1.out</Command>
    <Execute>FILE adapter called to process output type TA1OUT - Result: E9000 The TX File adapter was unable to
locate the target output directory structure. Ensure that the configuration input has the correct location and that
directory exists.
    ** SEND FAILURE ** </Execute>
    <FrameworkErrorOccurred>Send output to file failed</FrameworkErrorOccurred>
</Send>
</SendOutput>
```

4. If you see the following message, that means there is something wrong with the edisvu shared library location.
Return code/text: -99999/Adapter not found structure failed. DLL path: ./ Check the edicc.ini file to make sure that it correctly specifies the edisvu shared library
 - a. Make sure the edicc.ini is transferred in ASCII mode. The removal of extra carriage-return when transferring the file is required.
 5. Finally, if you are still experiencing problems, try to restart the container by issuing the command:

```
docker restart itx-rs
```

Supply Chain Examples

Design Server projects are also available for examples. The outputs for most of the examples do not follow any output file structure and can be easily identified with the output override in the curl command.

Exception: edifactExampleBlockChain.zip which contains an output file structure. The project contains two maps that can be executed to retrieve the final output results.

When creating the package for the edifactExampleBlockChain project, select aViewOrderChain, aBuyerNewOrder, and aViewOrderChainInfo and identify the URL for each map. The main execution map is aBuyerNewOrder. The aViewOrderChain map will return the complete order chain and the aViewOrderChainInfo will return a summary of execution.

Example commands:

```
curl -X PUT "http://localhost:8080/tx-rest/v2/run/aBuyerNewOrder"
curl -X PUT "http://localhost:8080/tx-rest/v2/run/aViewOrderChain?output=1;FILE;/data/edifactblockchain/orderchain.out"
```

Flow Example: x12isxTransform.zip which contains 2 Flows, x12ToJsonFlow and xmlToX12Flow. The project contains both flows. The example is designed to execute out of the box using the Flow Designer. With container execution, modifications to the flow output terminals will be needed.

The x12ToJsonFlow contains 2 possible output terminals depending on the Flow variable setting in the decision node of the flow. Terminal 2 generates xml output. The decision node of the flow uses the flow variable convToJson to generate either json output with output terminal 1 or xml output with output terminal 2. Using Design Server and creating the package, select all maps and files and clear URL. Add the URL to maps x12toxml and xmlo tox12. Under the Flow, select x12ToJsonFlow with URL and under default Flow, select mapsandschemas/x12ToJsonFlow. Under files, select all files. Deploy the package to the container and restart the container.

Example command:

```
curl -X PUT "http://172.17.0.1:8090/tx-rest/v2/run/x12ToJsonFlow"
```

Results:

```
{"status": -73, "run_details": {"host": "172.17.0.1", "port": 8090}, "platform": "Linux", "product_version": "11.0.1.0(54)", "name": "x12ToJsonFlow", "id": "66ed7ecb2afbfcc634c7e072a", "timestamp": "2024-09-20T14:08:11.588Z", "elapsed_time": 12, "code": -74, "status": "Flow instance Executed", "message": "Commit output flow terminal failed Terminal=2"}
```

Using Flow Designer, modify the action property setting for output terminal xmloput from create to create unless empty. The same modification is needed for the output terminal jsonOutput. Next, open the flow variables under settings and use the Discover button. Set the flow variable convToJson value to Y. Save the flow, analyze, and run.

Re-deploy the package for the x12ToJsonFlow and re-execute the curl command. Execution should be successful but we need to override the output to be written to the container mount location /tmp/itx-rs/data.

Example command:

```
curl -X PUT "http://localhost:8080/tx-rest/v2/run/x12ToJsonFlow?output=1;FILE;/data/x12ToJson.out"
```

To execute the x12ToJsonFlow to get XML output, you can issue the following command which sets the flow variable to the value N.

```
curl -X PUT "http://localhost:8080/tx-rest/v2/run/x12ToJsonFlow?convToJson=N&output=2;FILE;/data/x12ToXml.out"
```

The xmlToX12Flow contains 1 output terminal. Create a package for this flow, selecting the xmlToX12Flow similar to the X12ToJsonFlow. Deploy the package to the container. Override output 1 similar to the X12ToJsonFlow, using a similar execution command.

SAP Applications

- [Install and run SAP maps](#)
- [Run SAP example maps using endpoint for V1 REST API](#)

- [Run SAP example maps using endpoint for V2 REST API](#)

Install and run SAP maps

About this task

Perform the following steps to install and run SAP maps:

Procedure

1. On Windows machine with Design Studio installed, install IBM Sterling Transformation Extender Pack for SAP Applications V10.2.0.1 for Certified Containers (itxsap_10201_1101_rs.exe).
2. On Linux machine with container installed, ensure you have directories like:
/tmp/itx-rs/logs
/tmp/itx-rs/data/extra
/tmp/itx-rs/data/maps
/tmp/itx-rs/data/tmp
/tmp/itx-rs/data/mqdata
3. Copy file C:\Program Files\IBM\TransformationExtender_11.0.1\sap_pack_libs.tar from Windows to Linux. On Linux, extract contents into /tmp/itx-rs/data/extra directory.
4. Obtain the following user provided libraries from SAP JCo 3.1.10 or later:
libsapjco3.so
sapjco3.jar
5. Copy these files to /tmp/itx-rs/data/extra and ensure they have execute permissions.
/tmp/itx-rs/data/extra should now contain:
.sapjco3.jar
.m4r3ale.jar
.dssap.jar
.m4r3bapi.jar
.libsapjco3.so
.p4saphana.jar

Run SAP example maps using endpoint for V1 REST API

About this task

Perform the following steps to run the SAP example maps:

Procedure

1. For SAP examples installed to C:\ProgramData\IBM\TransformationExtender_11.0.1\examples\sap, edit the maps in Design Studio to use JALE and JBAPI command lines specific to your SAP installation.
2. Build all maps for Linux and copy maps\input files to Linux under /tmp/itx-rs/data/maps/sap.
3. Special notes for executing these example maps on Linux:
 - a. For any text input files (like legacy.txt), you must run **dos2unix** on the file to convert it to Unix line endings.
 - b. Note that the BAPI example's xSingleStyle map calls xBapiInput.mmc by name, so even on Linux, the compiled map needs to have an .mmc extension. Rename xBapiInput.lnx to xBapiInput.mmc.
 - c. The PEPPOL example maps use input and output files including a path name and a backslash as a path separator. These need to be updated with a Linux path separator, and remove any spaces. That is, change input\PEPPOL Invoice example1.xml to input/example1.xml and rename the file itself before copying it to Linux.
 - d. The S4HANA_REST example has an absolute path in the output card's GET call to the configuration file SAP_HANA_REST.json. This path needs to be updated to a Linux path such as /tmp/itx-rs/data/maps/sap/S4HANA_REST/SAP_HANA_REST.json
4. You should then have the following files under /tmp/itx-rs/data/maps:
.JBAPI/xSingleStyle.lnx
.JBAPI/xBapiInput.mmc
.DXOB/LegacyToDXO050.lnx
.DXOB/legacy.txt
.DXOB/leg_dxob.txt
.PEPPOL/from_ubl_invoice.lnx
.PEPPOL/input/example1.xml
.PEPPOL/input/SAPInvoice_mod.dat
.PEPPOL/to_ubl_invoice.lnx
.JALE/JALEOutboundToLegacy.lnx
.JALE/LegacyToInboundJALE.lnx

```

./JALE/legacy.txt
./EDI/invoice02_to_810/InvoicePackets.lnx
./EDI/invoice02_to_810/SAPIInvoice_mod.dat
./EDI/invoice02_to_810/validate_output.lnx
./EDI/850_to_orders05/x12_po.txt
./EDI/850_to_orders05/850_to_Orders.lnx
./S4HANA_REST/GetS4HANA.lnx
./S4HANA_REST/PostToS4HANA.lnx
./S4HANA_REST/ProcOrdConf2_postdata.txt
./S4HANA_REST/SAP_HANA_REST.json

```

5. You can then execute the maps with the following commands:

```

curl -X PUT -d "Testing" http://localhost:8080/tx-rest/v1/itx/maps/direct/sap/DXOB/LegacyToDX0050
curl -X PUT -d "Testing" http://localhost:8080/tx-rest/v1/itx/maps/direct/sap/JALE/LegacyToInboundJALE
curl -X PUT -d "Testing" http://localhost:8080/tx-rest/v1/itx/maps/direct/sap/JALE/JALEOutboundToLegacy
curl -X PUT -d "Testing" http://localhost:8080/tx-rest/v1/itx/maps/direct/sap/JBAPI/xSingleStyle
curl -X PUT -d "Testing" http://localhost:8080/tx-rest/v1/itx/maps/direct/sap/EDI/850_to_orders05/850_to_Orders
curl -X PUT -d "Testing" http://localhost:8080/tx-rest/v1/itx/maps/direct/sap/EDI/invoice02_to_810/InvoicePackets
curl -X PUT -d "Testing" http://localhost:8080/tx-rest/v1/itx/maps/direct/sap/PEPPOL/to_ubl_invoice
curl -X PUT -d "Testing" http://localhost:8080/tx-rest/v1/itx/maps/direct/sap/PEPPOL/from_ubl_invoice
curl -X PUT -d "Testing" http://localhost:8080/tx-rest/v1/itx/maps/direct/sap/S4HANA_REST/GetS4HANA
curl -X PUT -d "Testing" http://localhost:8080/tx-rest/v1/itx/maps/direct/sap/S4HANA_REST/PostToS4HANA

```

Run SAP example maps using endpoint for V2 REST API

Prerequisites

1. Import the example .zip projects on <WINDOWS_pack_install_dir>/examples/sap/SAP_pack_examples_windows.zip into Design Server.
2. Create a package for the project including all maps and files.
3. Deploy package into ITX-RS server instance.
4. Set config.yaml property rest.mode to fenced.

Below are examples of REST calls using `curl` commands to execute Pack for SAP example maps:

- **JALE Inbound example**

```
curl -X 'PUT' 'http://localhost:8080/tx-rest/v2/run/LegacyToInboundJALE?
input=1;FILE;legacy.txt&return=log&audit_file=always' -H 'accept: */*'
```

- **JALE Outbound example**

```
curl -X 'PUT' 'http://localhost:8080/tx-rest/v2/run/JALEOutboundToLegacy?input=1%3BJALE%3B-a%20<program_id>%20-g%20<gateway_host>%20-x%20<gateway_port>%20-tv%20-ar3&return=log&audit_file=always' -H 'accept: */*'
```

- **JBAPI**

Note: You must edit the GET rule with adapter command line prior to exporting.

```
curl -X 'PUT' 'http://localhost:8080/tx-rest/v2/run/xSingleStyle?return=log&audit_file=always' -H 'accept: */*'
```

- **DXOB example**

```
curl -X 'PUT' 'http://localhost:8080/tx-rest/v2/run/LegacyToDX0050?return=log&audit_file=always' -H 'accept: */*'
```

- **EDI examples**

```
curl -X 'PUT' 'http://localhost:8080/tx-rest/v2/run/850_to_Orders?return=log&audit_file=always' -H 'accept: */*'
```

```
curl -X 'PUT' 'http://localhost:8080/tx-rest/v2/run/InvoicePackets?return=log&audit_file=always' -H 'accept: */*'
```

```
curl -X 'PUT' 'http://localhost:8080/tx-rest/v2/run/validate_output?return=log&audit_file=always' -H 'accept: */*'
```

Pack for Financial Payments (FINPAY)

- [Prerequisites](#)
- [Configuring SWIFT MT JVC](#)
- [Building maps for Linux](#)

- [Porting artifacts to UNIX host](#)
 - [Example map execution on ITX-RS using endpoint for V1 REST API](#)
 - [Overrides on input, output cards due to backward slashes](#)
 - [Example map execution on ITX-RS using endpoint for V2 REST API](#)
 - [Not available for execution on ITX-RS](#)
-

Prerequisites

Procedure

1. Install the finpay_n.n.n.n_v1101_4c.zip on Windows.

Example:

```
<bind_mount_dir>/maps/finpay
where <bind_mount_dir> = /tmp/itx-rs/data
```

2. Define the similar directories and subdirectories under <bind_mount_dir>/maps/finpay as defined under the following:

```
<WINDOWS_pack_install_dir>/financial_payments_vn.n.n.n/fix
<WINDOWS_pack_install_dir>/financial_payments_vn.n.n.n/nacha
<WINDOWS_pack_install_dir>/financial_payments_vn.n.n.n/sepa
<WINDOWS_pack_install_dir>/financial_payments_vn.n.n.n/swift
```

Where n represents the current version of the Pack for Financial Payments.

Note:

- No need to define a directory for type trees.
- <bind_mount_dir> refers to directory specified in volume (-v) option on the docker run command, that is,

```
docker run --name itx-rs -it -d -p 8080:8080 -v /tmp/itx-rs/data:/data -v /tmp/itx-rs/ logs:/logs ibm-itx-rs:11.0.1.0.20240927
```

Configuring SWIFT MT JVC

Procedure

1. On <WINDOWS_pack_install_dir>/financial_payments_vn.n.n.n/swift/mt/jvc/maps, update the jvalccyy.prop, replacing the WINDOWS directory references with the corresponding equivalent on ITX-RS docker container mounted data volume: /data/maps/finpay
2. Using Design Studio, update the map rule on the validate_message map, on output card #1 item JvalPropPath. Change the map rule from =NONE to the location of the jvalccyy.prop on the corresponding equivalent on the ITX-RS docker container mounted data volume: ="/data/maps/finpay/swift/mt/jvc/maps".
3. Copy the following JVC jars to the <bind_mount_dir>/extra directory by extracting from <WINDOWS_pack_install_dir>/financial_payments_vn.n.n.n/UIProjectImports/swiftMTjvcConfig.tar.gz
 - jvcwrap.jar
 - jvalccyy.jar
4. Restart the ITX-RS container, by issuing the following command:

```
docker restart itx-rs
```

Building maps for Linux

Procedure

1. Compile all maps for the Linux platform, with a file extension that matches the property rest.map.fileExtension specified in config.yaml. Default file extension for compile maps is set to mmc.
2. Prior to executing any of the following build scripts,
 - a. Using a text editor, open build_deploy_jvc.bat under <WINDOWS_pack_install_dir>/financial_payments_vn.n.n.n/swift/mt/jvc/maps and search for this, **set mcompileOptions= -P %PLATFORM% -A -E -O "%DeployDir%\maps*.mmc"** and change to **set mcompileOptions= -P %PLATFORM% -A -E -O "%DeployDir%\maps*.<rest.map.fileExtension>"**.
 - b. Save file and execute on command prompt, that is, build_deploy_jvc.bat LINUX.
 - c. Using a text editor, open build_deploy_lmf.bat under <WINDOWS_pack_install_dir>/financial_payments_vn.n.n.n/swift/mt/lmf and search for this, **set mcompileOptions= -P %PLATFORM% -A -E -O "%DeployDir%\maps*.mmc"** and change to **set mcompileOptions= -P %PLATFORM% -A -E -O "%DeployDir%\maps*.<rest.map.fileExtension>"**.
 - d. Save file and execute on command prompt, i.e. build_deploy_lmf.bat LINUX.
 - e. Using a text editor, open build_mt_mx_translator.bat under <WINDOWS_pack_install_dir>/financial_payments_vn.n.n.n/swift/mx/examples/mt_mx_translation/maps on a text editor and search for this, **set mcompileOptions= -A -E** and change to **set mcompileOptions= -P LINUX -A -E -O "*.<rest.map.fileExtension>"**.
 - f. Save file and execute on command prompt, i.e. build_mt_mx_translator.bat.
3. When running the mt950_split example, ensure the following run maps are built for Linux but with an extension .mmc:
split_mt950
v2_mt950_runmap_balance

Porting artifacts to UNIX host

- Port compiled maps, schemas, data files to the corresponding <bind_mount_dir>/finpay directories.
- All compiled maps must be ported as BINARY.
- All XML schemas must be ported as BINARY.
- All data files must be ported as BINARY except for the following:
 - In the mt_mx_translation example, mt_mx_errorcodes.txt must be ported as ASCII.

Example map execution on ITX-RS using endpoint for V1 REST API

Below are examples of REST calls using **curl** commands to execute compliance maps:

SWIFT

```
curl -X PUT "http://localhost:8080/tx-rest/v1/itx/maps/ direct/.finpay.swift.mt.jvc.maps.validate_messages?
input=1;FILE;mt101.mt" -H "accept: application/json" -H "Content-Type: application/json" -d "{}"

curl -X PUT "http://localhost:8080/tx-rest/v1/itx/maps/direct/.finpay.swift.mx.maps.swiftval?
input=1;FILE;../data/pacs_008_001_08.xml" -H "accept: application/json" -H "Content-Type: application/json" -d "{}"
```

SEPA

```
curl -X PUT "http://localhost:8080/tx-rest/v1/itx/maps/direct/.finpay.sepa.mapsandschemas.sepvalid" -H "accept:
application/json" -H "Content-Type: application/json" -d "{}"
```

NACHA

```
curl -X PUT "http://localhost:8080/tx-rest/v1/itx/maps/ direct/.finpay.nacha.examples.ach_validation.maps.achval01" -H "accept:
application/json" -H "Content-Type: application/json" -d "{}"
```

Overrides on input, output cards due to backward slashes

Note: Due to some maps reference to directories with Windows specific backward slashes (\), the following maps will require input, output card overrides.
Map isoccd01 on ACH to ISO 20022 Payment Initiation Credit Transfer example

```
curl -X PUT "http://localhost:8080/tx-
rest/v1/itx/maps/direct/.finpay.nacha.examples.ach_iso20022.scenario.credit_transfer.maps.isoccd01?
input=1;FILE;../data/iso_pain001_to_ccd_ct_input.xml&return=audit" -H "accept: application/json" -H "Content-Type:
application/json" -d "{}"
```

Map isoccd08 on ACH to ISO 20022 Payment Initiation Direct Debit example

```
curl -X PUT "http://localhost:8080/tx-
rest/v1/itx/maps/direct/.finpay.nacha.examples.ach_iso20022.scenario.direct_debit.maps.isoccd08?
input=1;FILE;../data/iso_pain008_to_ccd_dd_input.xml&return=audit" -H "accept: application/json" -H "Content-Type:
application/json" -d "{}"
```

Map isorjt01 on ACH Reject to ISO 20022 transformation example

```
curl -X PUT "http://localhost:8080/tx-rest/v1/itx/maps/
direct/.finpay.nacha.examples.ach_iso20022.scenario.reject.maps.isorjt01?input=1;FILE;../data/
iso_pain002_to_reject_ccd_ct_input.xml&return=audit" -H "accept: application/json" -H "Content-Type: application/json" -d "{}"
```

Map isorth01 on ACH Return to ISO 20022 transformation example

```
curl -X PUT "http://localhost:8080/tx-
rest/v1/itx/maps/direct/.finpay.nacha.examples.ach_iso20022.scenario.returns.maps.isorth01?
input=1;FILE;../data/iso_camt053_to_returns_ccd_ct_input.xml&return=audit" -H "accept: application/json" -H "Content-Type:
application/json" -d "{}"
```

Map sepfr3ct_pacs_008_minos on SEPA France example

```
curl -X PUT "http://localhost:8080/tx-rest/v1/itx/maps/direct/.finpay.sepa.examples.france.maps.sepfr3ct_pacs_008_minos?
output=1;FILE;../data/fr_ct_sepa_pacs008_minos.out&return=audit" -H "accept: application/json" -H "Content-Type:
application/json" -d "{}"
```

Map sepde2dd_pain008_dtaus on SEPA Germany example

```
curl -X PUT "http://localhost:8080/tx-rest/v1/itx/maps/direct/.finpay.sepa.examples.germany.maps.sepde2dd_pain008_dtaus?
input=2;FILE;../data/BIC.txt&return=audit" -H "accept: application/json" -H "Content-Type: application/json" -d "{}"
```

Map fxfm0001_email_fix_to_fixml on FIX Email example

```
curl -X PUT "http://localhost:8080/tx-
rest/v1/itx/maps/direct/.finpay.fix.examples.fixFMLEmail.maps.fxfm0001_email_fix_to_fixml?
output=1;FILE;../data/fixxml_email_50_out.xml&return=audit" -H "accept: application/json" -H "Content-Type: application/json" -d
"{}"
```

Map fxfm0009_adv_fix_to_fixml on FIX Advertisement example

```
curl -X PUT "http://localhost:8080/tx-rest/v1/itx/maps/direct/.finpay.fix.examples.fixFMLadv.maps.fxfm0009_adv_fix_to_fixml?output=1;FILE;../data/fixxml_adv_50_out.xml&return=audit" -H "accept: application/json" -H "Content-Type: application/json" -d "{}"
```

Example map execution on ITX-RS using endpoint for V2 REST API

Prerequisites

1. Import the example .zip projects on <WINDOWS_pack_install_dir>/financial_payments_vn.n.n.n/UIProjectImports into Design Server. See pack release notes for complete list of Design Server projects.
 2. Create packages for each project including all maps and files.
 - OPTIONAL: Clear out the URL of run maps, leaving only main maps with URL.
- WORKAROUND: Any issues running any of the endpoints for V2 REST API listed below that is related to XML metadata, manually add native XML schemas into package by importing them under File tab with a logical folder schemas since deployment of package does not include XSDs from project.
3. Deploy package into ITX-RS server instance.
 4. Set config.yaml property rest.mode to fenced.

Below are examples of REST calls using curl commands to execute compliance maps:

SWIFT

```
curl -X PUT "http://localhost:8080/tx-rest/v2/run/validate_messages?output=2;FILE;../data/maps/finpay/swift/mt/jvc/data/results.xml&return=log&audit_file=always" -H "accept: */*"  
curl -X PUT "http://localhost:8080/tx-rest/v2/run/swiftval?input=1;FILE;../data/pacs_008_001_13_invalid.xml&output=3;FILE;../data/maps/finpay/swift/mx/data/results.xml&return=log&audit_file=always" -H "accept: */*" -H "Content-Type: application/xml"
```

SEPA

```
curl -X PUT "http://localhost:8080/tx-rest/v2/run/sepavalid?input=1;FILE;../data/pacs008c_invalid.xml&output=3;FILE;../data/maps/finpay/sepa/mapsandschemas/results.xml&return=log&audit_file=always" -H "accept: */*" -H "Content-Type: application/xml"
```

NACHA

```
curl -X PUT "http://localhost:8080/tx-rest/v2/run/achval01?output=4;FILE;../data/maps/finpay/nacha/examples/ach_validation/data/results.xml&return=log&audit_file=always" -H "accept: */*" -H "Content-Type: application/xml"
```

Map isoccd01 on ACH to ISO 20022 Payment Initiation Credit Transfer example

```
curl -X PUT "http://localhost:8080/tx-rest/v2/run/isoccd01?output=3;FILE;../data/maps/finpay/nacha/examples/ach_iso20022/scenario/credit_transfer/data/results.txt&return=log&audit_file=always" -H "accept: */*" -H "Content-Type: application/xml"
```

Map isoccd08 on ACH to ISO 20022 Payment Initiation Direct Debit example

```
curl -X PUT "http://localhost:8080/tx-rest/v2/run/isoccd08?output=3;FILE;../data/maps/finpay/nacha/examples/ach_iso20022/scenario/direct_debit/results.txt&return=log&audit_file=always" -H "accept: */*" -H "Content-Type: application/xml"
```

Map isorjt01 on ACH Reject to ISO 20022 transformation example

```
curl -X PUT "http://localhost:8080/tx-rest/v2/run/isorjt01?output=3;FILE;../data/maps/finpay/nacha/examples/ach_iso20022/scenario/reject/results.txt&return=log&audit_file=always" -H "accept: */*" -H "Content-Type: application/xml"
```

Map isortn01 on ACH Return to ISO 20022 transformation example

```
curl -X PUT "http://localhost:8080/tx-rest/v2/run/isortn01?output=3;FILE;../data/maps/finpay/nacha/examples/ach_iso20022/scenario/returns/results.txt&return=log&audit_file=always" -H "accept: */*" -H "Content-Type: application/xml"
```

Map sepfr3ct_pacs_008_minos on SEPA France example

```
curl -X PUT "http://localhost:8080/tx-rest/v2/run/sepfr3ct_pacs_008_minos?output=1;FILE;../data/maps/finpay/sepa/examples/france/data/results.xml&return=log&audit_file=always" -H "accept: */*" -H "Content-Type: application/xml"
```

Map sepde2dd_pain008_dtaus on SEPA Germany example

```
curl -X PUT "http://localhost:8080/tx-rest/v2/run/sepde2dd_pain008_dtaus?output=1;FILE;../data/maps/finpay/sepa/examples/germany/data/results.txt&return=log&audit_file=always" -H "accept: */*" -H "Content-Type: application/xml"
```

Map fxfm0001_email_fix_to_fixml on FIX Email example

```
curl -X PUT "http://localhost:8080/tx-rest/v2/run/fxfm0001_email_fix_to_fixml?output=1;FILE;../data/maps/finpay/fix/examples/fixFMLEmail/data/results.xml&return=log&audit_file=always" -H "accept: */*" -H "Content-Type: application/xml"
```

Map fxfm0009_adv_fix_to_fixml on FIX Advertisement example

```
curl -X PUT "http://localhost:8080/tx-rest/v2/run/fxfm0009_adv_fix_to_fixml?output=1;FILE;../data/maps/finpay/fix/examples/fixFMLadv/data/results.xml&return=log&audit_file=always" -H "accept: */*" -H "Content-Type: application/xml"
```

Not available for execution on ITX-RS

The following examples are not supported on ITX-RS:

SWIFT

- Sterling B2B Integrator Envelope, De-envelope examples
- SWIFT MT validation launcher example

SEPA

- z/OS examples
 - Bundle
 - unbundle
 - passthrough
 - sample_jcl

NACHA

- Sterling B2B Integrator Envelope, De-envelope examples.

Pack for Financial Payments Plus (FINPLUS)

- [Prerequisites](#)
- [Configuring CBPR+ Translation](#)
- [Building maps for Linux](#)
- [Porting artifacts to UNIX host](#)
- [Example map execution on ITX-RS using endpoint for V1 REST API](#)
- [Example map execution on ITX-RS using endpoint for V2 REST API](#)
- [Example flow execution on ITX-RS using endpoint for V2 REST API](#)

Prerequisites

Procedure

1. Extract finipayplus_n.n.n.n_v1101_4c.zip on Windows.
2. On the host mounted directory `<bind_mount_dir>`, define a finplus directory, that is, `/tmp/itx-rs/maps/finplus`.
3. Define the similar directories and subdirectories under `<bind_mount_dir>/maps/finplus` as defined under the following:

```
<WINDOWS_pack_install_dir>/financial_payments_plus_vn.n.n.n/cbpr_plus  
<WINDOWS_pack_install_dir>/financial_payments_plus_vn.n.n.n/chaps_enh  
<WINDOWS_pack_install_dir>/financial_payments_plus_vn.n.n.n/chaps_l4l  
<WINDOWS_pack_install_dir>/financial_payments_plus_vn.n.n.n/eba_e1s1  
<WINDOWS_pack_install_dir>/financial_payments_plus_vn.n.n.n/nbor_regis  
<WINDOWS_pack_install_dir>/financial_payments_plus_vn.n.n.n/sic_rtgs  
<WINDOWS_pack_install_dir>/financial_payments_plus_vn.n.n.n/swift_gpi  
<WINDOWS_pack_install_dir>/financial_payments_plus_vn.n.n.n/swift_uc  
<WINDOWS_pack_install_dir>/financial_payments_plus_vn.n.n.n/t2_clm  
<WINDOWS_pack_install_dir>/financial_payments_plus_vn.n.n.n/t2_coco  
<WINDOWS_pack_install_dir>/financial_payments_plus_vn.n.n.n/t2_rtgs  
<WINDOWS_pack_install_dir>/financial_payments_plus_vn.n.n.n/t2_sec
```

Note:

- No need to define a directory for type trees.
- `<bind_mount_dir>` refers to directory specified in volume (-v) option on the docker run command, that is,

```
docker run --name itx-rs -it -d -p 8080:8080 -v /tmp/itx-rs/data:/data -v /tmp/itx-rs/ logs:/logs ibm-itx-rs:11.0.1.0.20240927
```

Configuring CBPR+ Translation

Procedure

1. On `<WINDOWS_pack_install_dir>/financial_payments_plus_vn.n.n.n/UIProjectImports`, extract contents of cbprJnodesConfigIBM.tar.gz.
2. Copy the following jars from `<TX_install_dir>/jars`
 - jackson-core-n.n.n.jar
 - jackson-annotations-n.n.n.jar
 - jackson-databind-n.n.n.jar

Or, download from com/fasterxml/jackson/core.

Note: It is recommended to use the latest version.

3. Copy the following JVC jars to the <bind_mount_dir>/extra directory:

- jnodesO.jar
- jackson-core-n.n.n.jar
- jackson-annotations-n.n.n.jar
- jackson-databind-n.n.n.jar

4. Restart the ITX-RS container, that is,

```
docker restart itx-rs
```

Building maps for Linux

Procedure

1. Compile all maps for the Linux platform, with a file extension that matches the property rest.map.fileExtension specified in config.yaml. Default file extension for compile maps is set to mmc.
2. Execute the scripts under <WINDOWS_pack_install_dir>/financial_payments_plus_vn.n.n.n/tools/build_scripts. For example, build_deploy_cbpr_plus_mt2mx_translation.bat LINUX
3. When running any _framework maps and _val maps, ensure the following run maps are built for Linux but with an extension <rest.map.fileExtension>. For all other run maps, retain the extension .mmc.

Porting artifacts to UNIX host

- Port compiled maps, schemas, data files to the corresponding <bind_mound_dir>/finplus directories.
- All compiled maps must be ported as BINARY.
- All XML schemas must be ported as BINARY.
- All data files must be ported as BINARY.

Example map execution on ITX-RS using endpoint for V1 REST API

Below are examples of REST calls using curl commands to execute translation and compliance maps:

- **CBPR+ Translation**
 - **MT-MX**

```
curl -X PUT "http://localhost:8080/tx-rest/v1/itx/maps/direct/.finplus.cbpr.mt-mx.maps.cbpr2800_mt103_framework?input=1;FILE;../data/MT103.inp&return=audit" -H "accept: application/json" -H "Content-Type: application/json" -d "{}"

curl -X PUT "http://localhost:8080/tx-rest/v1/itx/maps/direct/.finplus.cbpr.mt-mx.maps.cbpr2810_mt202_framework?input=1;FILE;../data/MT202_ADV.inp&return=audit" -H "accept: application/json" -H "Content-Type: application/json" -d "{}"

curl -X PUT "http://localhost:8080/tx-rest/v1/itx/maps/direct/.finplus.cbpr.mt-mx.maps.cbpr2811_mt205_framework?input=1;FILE;../data/MT205_COR.inp&return=audit" -H "accept: application/json" -H "Content-Type: application/json" -d "{}"

curl -X PUT "http://localhost:8080/tx-rest/v1/itx/maps/direct/.finplus.cbpr.mt-mx.maps.cbpr1500_mt9n0_framework?input=1;FILE;../data/MT900.inp&return=audit" -H "accept: application/json" -H "Content-Type: application/json" -d "{}"

curl -X PUT "http://localhost:8080/tx-rest/v1/itx/maps/direct/.finplus.cbpr.mt-mx.maps.cbpr2820_mtn96_framework?input=1;FILE;../data/MT196.inp&return=audit" -H "accept: application/json" -H "Content-Type: application/json" -d "{}"

curl -X PUT "http://localhost:8080/tx-rest/v1/itx/maps/direct/.finplus.cbpr_plus.mt-mx.maps.cbpr2820_mtn96_framework?input=1;FILE;../data/MT296.inp&return=audit" -H "accept: application/json" -H "Content-Type: application/json" -d "{}"
```

- **MX-MT**

```
curl -X PUT "http://localhost:8080/tx-rest/v1/itx/maps/direct/.finplus.cbpr.mx-mt.maps.cbpr2500_pacs002_framework?input=1;FILE;../data/env_pacs_002_199.xml&return=audit" -H "accept: application/json" -H "Content-Type: application/json" -d "{}"

curl -X PUT "http://localhost:8080/tx-rest/v1/itx/maps/direct/.finplus.cbpr.mx-mt.maps.cbpr2700_pacs008_framework?input=1;FILE;../data/env_pacs_008_103.xml&return=audit" -H "accept: application/json" -H "Content-Type: application/json" -d "{}"

curl -X PUT "http://localhost:8080/tx-rest/v1/itx/maps/direct/.finplus.cbpr.mx-mt.maps.cbpr2710_pacs009_mt202_framework?input=1;FILE;../data/env_pacs_009_mt202cor_valid.xml&return=audit" -H "accept: application/json" -H "Content-Type: application/json" -d "{}"

curl -X PUT "http://localhost:8080/tx-rest/v1/itx/maps/direct/.finplus.cbpr.mx-mt.maps.cbpr2724_pacs009_mt205_framework?input=1;FILE;../data/env_pacs_009_mt205cor_valid.xml&return=audit" -H "accept: application/json" -H "Content-Type: application/json" -d "{}"
```

```

curl -X PUT "http://localhost:8080/tx-rest/v1/itx/maps/direct/.finplus.cbpr.mx-mt.maps.cbpr2520_camt029_framework?
input=1;FILE;../data/env_camt_029_valid.xml&return=audit" -H "accept: application/json" -H "Content-Type:
application/json" -d "{}"

curl -X PUT "http://localhost:8080/tx-rest/v1/itx/maps/direct/.finplus.cbpr.mx-mt.maps.cbpr2510_camt054_framework?
input=1;FILE;../data/env_camt_054_crdt.xml&return=audit" -H "accept: application/json" -H "Content-Type:
application/json" -d "{}"

curl -X PUT "http://localhost:8080/tx-rest/v1/itx/maps/direct/.finplus.cbpr.mx-mt.maps.cbpr2600_camt057_framework?
input=1;FILE;../data/env_camt_057_valid.xml&return=audit" -H "accept: application/json" -H "Content-Type:
application/json" -d "{}"

• CBPR+ Validation
  ○ MX Extended

    curl -X PUT "http://localhost:8080/tx-rest/v1/itx/maps/direct/.finplus.cbpr.mx_extended.maps.cbpr6000_val?
input=1;FILE;../data/bah_pacs_008_001_08_valid.xml&return=audit" -H "accept: application/json" -H "Content-Type:
application/json" -d "{}"

  ○ Schema Only

    curl -X PUT "http://localhost:8080/tx-rest/v1/itx/maps/direct/.finplus.cbpr.schema_only.maps.cbpr6100_val?
input=1;FILE;../data/bah_pain_001_001_09_valid.xml&return=audit" -H "accept: application/json" -H "Content-Type:
application/json" -d "{}"

• CHAPS ENH Validation
  ○ MX Extended

    curl -X PUT "http://localhost:8080/tx-rest/v1/itx/maps/direct/.finplus.chaps_enh.mx_extended.maps.chen8000_val?
input=1;FILE;../data/bah_pacs_004_001_09_valid.xml&return=audit" -H "accept: application/json" -H "Content-Type:
application/json" -d "{}"

  ○ Schema Only

    curl -X PUT "http://localhost:8080/itx-rs/v1/maps/direct/.finplus.chaps_enh.schema_only.maps.chen8200_val?
input=1;FILE;../data/bah_camt_052_001_08_valid.xml&return=audit" -H "accept: application/json" -H "Content-Type:
application/json" -d "{}"

• CHAPS L4L Validation
  ○ MX Extended

    curl -X PUT "http://localhost:8080/itx-rs/v1/maps/direct/.finplus.chaps_l4l.mx_extended.maps.chlf8000_val?
input=1;FILE;../data/bah_camt_053_001_08_valid.xml&return=audit" -H "accept: application/json" -H "Content-Type:
application/json" -d "{}"

  ○ Schema Only

    curl -X PUT "http://localhost:8080/tx-rest/v1/itx/maps/direct/.finplus.chaps_l4l.schema_only.maps.chlf8200_val?
input=1;FILE;../data/bah_camt_060_001_05_valid.xml&return=audit" -H "accept: application/json" -H "Content-Type:
application/json" -d "{}"

• EBA E1S1 Validation
  ○ MX Extended

    curl -X PUT "http://localhost:8080/tx-rest/v1/itx/maps/direct/.finplus.eba_e1s1.mx_extended.maps.e1s17000_val?
input=1;FILE;../data/admi_007_001_01_valid.xml&return=audit" -H "accept: application/json" -H "Content-Type:
application/json" -d "{}"

  ○ Schema Only

    curl -X PUT "http://localhost:8080/tx-rest/v1/itx/maps/direct/.finplus.eba_e1s1.schema_only.maps.e1s17200_val?
input=1;FILE;../data/bah_pacs_004_001_09_valid.xml&return=audit" -H "accept: application/json" -H "Content-Type:
application/json" -d "{}"

• SWIFT UC Validation
  ○ MX Extended

    curl -X PUT "http://localhost:8080/tx-rest/v1/itx/maps/direct/.finplus.swift_uc.mx_extended.maps.swuc1000_val?
input=1;FILE;../data/bah_trck_001_001_02_uc_valid.xml&return=audit" -H "accept: application/json" -H "Content-Type:
application/json" -d "{}"

  ○ Schema Only

    curl -X PUT "http://localhost:8080/tx-rest/v1/itx/maps/direct/.finplus.swift_uc.schema_only.maps.swuc1500_val?
input=1;FILE;../data/bah_trck_001_001_02_uc_valid.xml&return=audit" -H "accept: application/json" -H "Content-Type:
application/json" -d "{}"

• SWIFT GPI Validation
  ○ MX Extended

    curl -X PUT "http://localhost:8080/tx-rest/v1/itx/maps/direct/.finplus.swift_gpi.mx_extended.maps.swgp1000_val?
input=1;FILE;../data/bah_trck_005_001_02_valid.xml&return=audit" -H "accept: application/json" -H "Content-Type:
application/json" -d "{}"

  ○ Schema Only

    curl -X PUT "http://localhost:8080/tx-rest/v1/itx/maps/direct/.finplus.swift_gpi.schema_only.maps.swgp1500_val?
input=1;FILE;../data/bah_trck_005_001_02_valid.xml&return=audit" -H "accept: application/json" -H "Content-Type:
application/json" -d "{}"

• T2 CLM Validation
  ○ MX Extended

    curl -X PUT "http://localhost:8080/tx-rest/v1/itx/maps/direct/.finplus.t2_clm.mx_extended.maps.t2cl9000_val?
input=1;FILE;../data/bah_camt_003_001_07_valid.xml&return=audit" -H "accept: application/json" -H "Content-Type:
application/json" -d "{}"

```

```

application/json" -d "{}"

○ Schema Only

curl -X PUT "http://localhost:8080/tx-rest/v1/itx/maps/direct/.finplus.t2_clm.schema_only.maps.t2c19500_val?
input=1;FILE:../data/bah_camt_004_001_08_valid.xml&return=audit" -H "accept: application/json" -H "Content-Type:
application/json" -d "{}"

● T2 CoCo Validation
○ MX Extended

curl -X PUT "http://localhost:8080/tx-rest/v1/itx/maps/direct/.finplus.t2_coco.mx_extended.maps.t2co9000_val?
input=1;FILE:../data/bah_reda_040_001_01_valid.xml&return=audit" -H "accept: application/json" -H "Content-Type:
application/json" -d "{}"

○ Schema Only

curl -X PUT "http://localhost:8080/tx-rest/v1/itx/maps/direct/.finplus.t2_coco.schema_only.maps.t2co9500_val?
input=1;FILE:../data/bah_camt_100_001_01_valid.xml&return=audit" -H "accept: application/json" -H "Content-Type:
application/json" -d "{}"

● T2 RTGS Validation
○ MX Extended

curl -X PUT "http://localhost:8080/tx-rest/v1/itx/maps/direct/.finplus.t2_rtgs.mx_extended.maps.t2rt9000_val?
input=1;FILE:../data/bah_pain_998_001_01_atn_valid.xml&return=audit" -H "accept: application/json" -H "Content-Type:
application/json" -d "{}"

○ Schema Only

curl -X PUT "http://localhost:8080/tx-rest/v1/itx/maps/direct/.finplus.t2_rtgs.schema_only.maps.t2rt9500_val?
input=1;FILE:../data/bah_pain_998_001_01_ais_valid.xml&return=audit" -H "accept: application/json" -H "Content-Type:
application/json" -d "{}"

● NBOR ReGIS Validation
○ Schema Only

curl -X PUT "http://localhost:8080/tx-rest/v1/itx/maps/direct/.finplus.nbor_regis.schema_only.maps.rgis1500_val?
input=1;FILE:../data/bah_camt_029_001_09_valid.xml&return=audit" -H "accept: application/json" -H "Content-Type:
application/json" -d "{}"

● SIC RTGS Validation
○ Schema Only

curl -X PUT "http://localhost:8080/tx-rest/v1/itx/maps/direct/.finplus.sic_rtgs.schema_only.maps.sirt3500_val?
input=1;FILE:../data/acmt_010_001_03_valid.xml&return=audit" -H "accept: application/json" -H "Content-Type:
application/json" -d "{}"

● T2 Sec Validation
○ Schema Only

curl -X PUT "http://localhost:8080/tx-rest/v1/itx/maps/direct/.finplus.t2_sec.schema_only.maps.t2se9500_val?
input=1;FILE:../data/bah_admi_005_001_01_valid.xml&return=audit" -H "accept: application/json" -H "Content-Type:
application/json" -d "{}"

curl -X PUT "http://localhost:8080/tx-rest/v1/itx/maps/direct/.finplus.t2_sec.schema_only.maps.t2se9600_val?
input=1;FILE:../data/head_002_001_01_valid.xml&return=audit" -H "accept: application/json" -H "Content-Type:
application/json" -d "{}"

```

Example map execution on ITX-RS using endpoint for V2 REST API

Prerequisites

1. Import the example .zip projects on <WINDOWS_pack_install_dir>/financial_payments_plus_vn.n.n.n/UIProjectImports into Design Server. See pack release notes for complete list of Design Server projects.
2. Create packages for each project including all maps and files.
 - OPTIONAL: Clear out the URL of run maps, leaving only main maps with URL.
3. Deploy package into ITX-RS server instance.
4. Set config.yaml property rest.mode to fenced.

Below are examples of REST calls using curl commands to execute translation and compliance maps:

```

● CBPR+ Translation
○ MT-MX

curl -X PUT "http://localhost:8080/tx-rest/v2/run/cbpr2800_mt103_framework?
output=3;FILE:/data/maps/finplus/cbpr_plus/mt-mx/data/results.xml&output=4;FILE:/data/maps/finplus/cbpr_plus/mt-
mx/data/audit_msg.json&return=log&audit_file=always" -H "accept: /*/*"

curl -X PUT "http://localhost:8080/tx-rest/v2/run/cbpr2810_mt202_framework?
output=3;FILE:/data/maps/finplus/cbpr_plus/mt-mx/data/results.xml&output=4;FILE:/data/maps/finplus/cbpr_plus/mt-
mx/data/audit_msg.json&return=log&audit_file=always" -H "accept: /*/*"

curl -X PUT "http://localhost:8080/tx-rest/v2/run/cbpr2811_mt205_framework?
output=3;FILE:/data/maps/finplus/cbpr_plus/mt-mx/data/results.xml&output=4;FILE:/data/maps/finplus/cbpr_plus/mt-
mx/data/audit_msg.json&return=log&audit_file=always" -H "accept: /*/*"

curl -X PUT "http://localhost:8080/tx-rest/v2/run/cbpr1500_mt9n0_framework?
output=3;FILE:/data/maps/finplus/cbpr_plus/mt-mx/data/results.xml&output=4;FILE:/data/maps/finplus/cbpr_plus/mt-
mx/data/audit_msg.json&return=log&audit_file=always" -H "accept: /*/*"

```

```

mx/data/audit_msg.json&return=log&audit_file=always" -H "accept: */*
curl -X PUT "http://localhost:8080/tx-rest/v2/run/cbpr2820_mtn96_framework?
output=3;FILE;/data/maps/finplus/cbpr_plus/mt-mx/data/results.xml&output=4;FILE;/data/maps/finplus/cbpr_plus/mt-
mx/data/audit_msg.json&return=log&audit_file=always" -H "accept: */*
curl -X PUT "http://localhost:8080/tx-rest/v2/run/cbpr2820_mtn96_framework?
input=1;FILE;../data/MT296.inp&output=3;FILE;/data/maps/finplus/cbpr_plus/mt-
mx/data/results.xml&output=4;FILE;/data/maps/finplus/cbpr_plus/mt-
mx/data/audit_msg.json&return=log&audit_file=always" -H "accept: */*

```

- MX-MT


```

curl -X PUT "http://localhost:8080/tx-rest/v2/run/cbpr2500_pacs002_framework?
output=2;FILE;/data/maps/finplus/cbpr_plus/mx-mt/data/results.out&output=3;FILE;/data/maps/finplus/cbpr_plus/mx-
mt/data/audit_msg.json&return=log&audit_file=always" -H "accept: */*
curl -X PUT "http://localhost:8080/tx-rest/v2/run/cbpr2700_pacs008_framework?
output=2;FILE;/data/maps/finplus/cbpr_plus/mx-mt/data/results.out&output=3;FILE;/data/maps/finplus/cbpr_plus/mx-
mt/data/audit_msg.json&return=log&audit_file=always" -H "accept: */*
curl -X PUT "http://localhost:8080/tx-rest/v2/run/cbpr2710_pacs009_mt202_framework?
output=2;FILE;/data/maps/finplus/cbpr_plus/mx-mt/data/results.out&output=3;FILE;/data/maps/finplus/cbpr_plus/mx-
mt/data/audit_msg.json&return=log&audit_file=always" -H "accept: */*
curl -X PUT "http://localhost:8080/tx-rest/v2/run/cbpr2724_pacs009_mt205_framework?
output=2;FILE;/data/maps/finplus/cbpr_plus/mx-mt/data/results.out&output=3;FILE;/data/maps/finplus/cbpr_plus/mx-
mt/data/audit_msg.json&return=log&audit_file=always" -H "accept: */*
curl -X PUT "http://localhost:8080/tx-rest/v2/run/cbpr2520_camt029_framework?
output=2;FILE;/data/maps/finplus/cbpr_plus/mx-mt/data/results.out&output=3;FILE;/data/maps/finplus/cbpr_plus/mx-
mt/data/audit_msg.json&return=log&audit_file=always" -H "accept: */*
curl -X PUT "http://localhost:8080/tx-rest/v2/run/cbpr2510_camt054_framework?
output=2;FILE;/data/maps/finplus/cbpr_plus/mx-mt/data/results.out&output=3;FILE;/data/maps/finplus/cbpr_plus/mx-
mt/data/audit_msg.json&return=log&audit_file=always" -H "accept: */*
curl -X PUT "http://localhost:8080/tx-rest/v2/run/cbpr2600_camt057_framework?
output=2;FILE;/data/maps/finplus/cbpr_plus/mx-mt/data/results.out&output=3;FILE;/data/maps/finplus/cbpr_plus/mx-
mt/data/audit_msg.json&return=log&audit_file=always" -H "accept: */*

```
- CBPR+ Validation
 - MX Extended


```

curl -X PUT "http://localhost:8080/tx-rest/v2/run/cbpr6000_val?
output=1;FILE;/data/maps/finplus/cbpr_plus/mx_extended/data/results.xml&return=log&audit_file=always" -H "accept: */*

```
 - Schema Only


```

curl -X PUT "http://localhost:8080/tx-rest/v2/run/cbpr6100_val?
output=1;FILE;/data/maps/finplus/cbpr_plus/schema_only/data/results.xml&return=log&audit_file=always" -H "accept: */*

```
- CHAPS ENH Validation
 - MX Extended


```

curl -X PUT "http://localhost:8080/tx-rest/v2/run/chen8000_val?
output=1;FILE;/data/maps/finplus/chaps_enh/mx_extended/data/results.xml&return=log&audit_file=always" -H "accept: */*

```
 - Schema Only


```

curl -X PUT "http://localhost:8080/tx-rest/v2/run/chen8200_val?
output=1;FILE;/data/maps/finplus/chaps_enh/schema_only/data/results.xml&return=log&audit_file=always" -H "accept: */*

```
- CHAPS L4L Validation
 - MX Extended


```

curl -X PUT "http://localhost:8080/tx-rest/v2/run/chlf8000_val?
output=1;FILE;/data/maps/finplus/chaps_141/mx_extended/data/results.xml&return=log&audit_file=always" -H "accept: */*

```
 - Schema Only


```

curl -X PUT "http://localhost:8080/tx-rest/v2/run/chlf8200_val?
output=1;FILE;/data/maps/finplus/chaps_141/schema_only/data/results.xml&return=log&audit_file=always" -H "accept: */*

```
- EBA E1S1 Validation
 - MX Extended


```

curl -X PUT "http://localhost:8080/tx-rest/v2/run/e1s17000_val?
output=1;FILE;/data/maps/finplus/eba_e1s1/mx_extended/data/results.xml&return=log&audit_file=always" -H "accept: */*

```
 - Schema Only


```

curl -X PUT "http://localhost:8080/tx-rest/v2/run/e1s17200_val?
output=1;FILE;/data/maps/finplus/eba_e1s1/schema_only/data/results.xml&return=log&audit_file=always" -H "accept: */*

```
- SWIFT UC Validation
 - MX Extended


```

curl -X PUT "http://localhost:8080/tx-rest/v2/run/swuc1000_val?
output=1;FILE;/data/maps/finplus/swift_uc/mx_extended/data/results.xml&return=log&audit_file=always" -H "accept: */*

```

- Schema Only


```
curl -X PUT "http://localhost:8080/tx-rest/v2/run/swuc1500_val?
output=1;FILE;/data/maps/finplus/swift_uc/schema_only/data/results.xml&return=log&audit_file=always" -H "accept: */*"
```
- SWIFT GPI Validation
 - MX Extended


```
curl -X PUT "http://localhost:8080/tx-rest/v2/run/swgp1000_val?
output=1;FILE;/data/maps/finplus/swift_gpi/mx_extended/data/results.xml&return=log&audit_file=always" -H "accept: */*"
```
 - Schema Only


```
curl -X PUT "http://localhost:8080/tx-rest/v2/run/swgp1500_val?
output=1;FILE;/data/maps/finplus/swift_gpi/schema_only/data/results.xml&return=log&audit_file=always" -H "accept: */*"
```
- T2 CLM Validation
 - MX Extended


```
curl -X PUT "http://localhost:8080/tx-rest/v2/run/t2cl9000_val?
output=1;FILE;/data/maps/finplus/t2_clm/mx_extended/data/results.xml&return=log&audit_file=always" -H "accept: */*"
```
 - Schema Only


```
curl -X PUT "http://localhost:8080/tx-rest/v2/run/t2cl9500_val?
output=1;FILE;/data/maps/finplus/t2_clm/schema_only/data/results.xml&return=log&audit_file=always" -H "accept: */*"
```
- T2 CoCo Validation
 - MX Extended


```
curl -X PUT "http://localhost:8080/tx-rest/v2/run/t2co9000_val?
output=1;FILE;/data/maps/finplus/t2_coco/mx_extended/data/results.xml&return=log&audit_file=always" -H "accept: */*"
```
 - Schema Only


```
curl -X PUT "http://localhost:8080/tx-rest/v2/run/t2co9500_val?
output=1;FILE;/data/maps/finplus/t2_coco/schema_only/data/results.xml&return=log&audit_file=always" -H "accept: */*"
```
- T2 RTGS Validation
 - MX Extended


```
curl -X PUT "http://localhost:8080/tx-rest/v2/run/t2rt9000_val?
output=1;FILE;/data/maps/finplus/t2_rtgs/mx_extended/data/results.xml&return=log&audit_file=always" -H "accept: */*"
```
 - Schema Only


```
curl -X PUT "http://localhost:8080/tx-rest/v2/run/t2rt9500_val?
output=1;FILE;/data/maps/finplus/t2_rtgs/schema_only/data/results.xml&return=log&audit_file=always" -H "accept: */*"
```
- NBOR ReGIS Validation
 - Schema Only


```
curl -X PUT "http://localhost:8080/tx-rest/v2/run/rgis1500_val?
output=1;FILE;/data/maps/finplus/nbor_regis/schema_only/data/results.xml&return=log&audit_file=always" -H "accept: */*"
```
- SIC RTGS Validation
 - Schema Only


```
curl -X PUT "http://localhost:8080/tx-rest/v2/run/sirt3500_val?
output=1;FILE;/data/maps/finplus/sic_rtgs/schema_only/data/results.xml&return=log&audit_file=always" -H "accept: */*"
```
- T2 Sec Validation
 - Schema Only


```
curl -X PUT "http://localhost:8080/tx-rest/v2/run/t2se9500_val?
output=1;FILE;/data/maps/finplus/t2_sec/schema_only/data/results.xml&return=log&audit_file=always" -H "accept: */*"
```
 - ```
curl -X PUT "http://localhost:8080/tx-rest/v2/run/t2se9600_val?
output=1;FILE;/data/maps/finplus/t2_sec/schema_only/data/pyld_results.xml&return=log&audit_file=always" -H "accept: */*"
```

---

## Example flow execution on ITX-RS using endpoint for V2 REST API

### Prerequisites

1. Import the example .zip projects on <WINDOWS\_pack\_install\_dir>/financial\_payments\_plus\_vn.n.n.n/UIProjectImports into Design Server. See pack release notes for complete list of Design Server projects.
2. Take note of the flow variables used for each flow since their default value can be overridden upon execution.
3. Create packages for each project including all maps and files.
  - OPTIONAL: Clear out the URL of run maps, leaving only main maps with URL.
4. Deploy package into ITX-RS server instance.
5. Set config.yaml property rest.mode to fenced.

Below are examples of REST calls using **curl** commands to execute translation and compliance flows:

- CBPR+ Translation

- MT-MX

```

curl -X GET "http://localhost:8080/tx-rest/v2/run/cbpr_mt103_pacs00n?INPUT_FILE=./cbpr_plus/translation/mt-
mx/data/MT103_RETN.inp&CONFIG_FILE=./cbpr_plus/translation/mt-
mx/data/trx_config.xml&OUTPUT_RESULT_MT=/data/maps/finplus/cbpr_plus/mt-
mx/data/pacs.00n.xml&AUDIT_LOG=/data/maps/finplus/cbpr_plus/mt-mx/data/audit_msg.json&return=log&audit_file=always" -
H "accept: */*"

curl -X GET "http://localhost:8080/tx-rest/v2/run/cbpr_mt9n0_camt054?INPUT_FILE=./cbpr_plus/translation/mt-
mx/data/MT900.inp&CONFIG_FILE=./cbpr_plus/translation/mt-
mx/data/trx_config.xml&OUTPUT_RESULT_MT=/data/maps/finplus/cbpr_plus/mt-
mx/data/env_camt_054_out.xml&AUDIT_LOG=/data/maps/finplus/cbpr_plus/mt-
mx/data/audit_msg.json&return=log&audit_file=always" -H "accept: */*"

curl -X GET "http://localhost:8080/tx-rest/v2/run/cbpr_mt205_pacs00n?INPUT_FILE=./cbpr_plus/translation/mt-
mx/data/MT205_COR.inp&OUTPUT_DOC_RESULT_XML=/data/maps/finplus/cbpr_plus/mt-
mx/data/pacs.00n.xml&CONFIG_FILE=./cbpr_plus/translation/mt-
mx/data/trx_config.xml&AUDIT_LOG=/data/maps/finplus/cbpr_plus/mt-mx/data/audit_msg.json&return=log&audit_file=always" -
H "accept: */*"

curl -X GET "http://localhost:8080/tx-rest/v2/run/cbpr_mtn96_camt029?INPUT_FILE=./cbpr_plus/translation/mt-
mx/data/MT196.inp&OUTPUT_DOC_RESULT_XML=/data/maps/finplus/cbpr_plus/mt-
mx/data/camt_029_out.xml&CONFIG_FILE=./cbpr_plus/translation/mt-
mx/data/trx_config.xml&AUDIT_LOG=/data/maps/finplus/cbpr_plus/mt-mx/data/audit_msg.json&return=log&audit_file=always" -
H "accept: */*"

curl -X GET "http://localhost:8080/tx-rest/v2/run/cbpr_mt202_pacs00n?INPUT_FILE=./cbpr_plus/translation/mt-
mx/data/MT202_COR.inp&OUTPUT_DOC_RESULT_XML=/data/maps/finplus/cbpr_plus/mt-
mx/data/pacs.00n.xml&CONFIG_FILE=./cbpr_plus/translation/mt-
mx/data/trx_config.xml&AUDIT_LOG=/data/maps/finplus/cbpr_plus/mt-mx/data/audit_msg.json&return=log&audit_file=always" -
H "accept: */*"

curl -X GET "http://localhost:8080/tx-rest/v2/run/cbpr_mtn92_camt056?INPUT_FILE=./cbpr_plus/translation/mt-
mx/data/MT192.inp&OUTPUT_DOC_RESULT_XML=/data/maps/finplus/cbpr_plus/mt-
mx/data/camt_056_out.xml&CONFIG_FILE=./cbpr_plus/translation/mt-
mx/data/trx_config.xml&AUDIT_LOG=/data/maps/finplus/cbpr_plus/mt-mx/data/audit_msg.json&return=log&audit_file=always" -
H "accept: */*"

curl -X GET "http://localhost:8080/tx-rest/v2/run/cbpr_mt103sgo_pacs008?INPUT_FILE=./cbpr_plus/translation/mt-
mx/data/MT103_SGO.inp&OUTPUT_DOC_RESULT_XML=/data/maps/finplus/cbpr_plus/mt-
mx/data/pacs_008_sgo.xml&CONFIG_FILE=./cbpr_plus/translation/mt-
mx/data/trx_config.xml&AUDIT_LOG=/data/maps/finplus/cbpr_plus/mt-mx/data/audit_msg.json&return=log&audit_file=always" -
H "accept: */*"

◦ MX-MT

curl -X GET "http://localhost:8080/tx-rest/v2/run/cbpr_camt107_mt110?INPUT_FILE=./cbpr_plus/translation/mx-
mt/data/env_camt_107_valid.xml&OUTPUT_RESULT_MT=/data/maps/finplus/cbpr_plus/mt-
mt/data/mt110.txt&AUDIT_LOG=/data/maps/finplus/cbpr_plus/mx-mt/data/audit_msg.json&return=log&audit_file=always" -H
"accept: */*"

curl -X GET "http://localhost:8080/tx-rest/v2/run/cbpr_camt052_mt942?INPUT_FILE=./cbpr_plus/translation/mx-
mt/data/env_camt_052_mt942_valid.xml&CONFIG_FILE=./cbpr_plus/translation/mx-
mt/data/trx_config.xml&OUTPUT_RESULT_MT=/data/maps/finplus/cbpr_plus/mt-
mt/data/mt942.txt&AUDIT_LOG=/data/maps/finplus/cbpr_plus/mx-mt/data/audit_msg.json&return=log&audit_file=always" -H
"accept: */*"

curl -X GET "http://localhost:8080/tx-rest/v2/run/cbpr_camt108_mt111?INPUT_FILE=./cbpr_plus/translation/mx-
mt/data/env_camt_108_valid.xml&CONFIG_FILE=./cbpr_plus/translation/mx-
mt/data/trx_config.xml&OUTPUT_RESULT_MT=/data/maps/finplus/cbpr_plus/mt-
mt/data/mt111.txt&AUDIT_LOG=/data/maps/finplus/cbpr_plus/mx-mt/data/audit_msg.json&return=log&audit_file=always" -H
"accept: */*"

curl -X GET "http://localhost:8080/tx-rest/v2/run/cbpr_camt053_mt940?INPUT_FILE=./cbpr_plus/translation/mx-
mt/data/env_camt_053_mt940_valid.xml&CONFIG_FILE=./cbpr_plus/translation/mx-
mt/data/trx_config.xml&OUTPUT_RESULT_MT=/data/maps/finplus/cbpr_plus/mt-
mt/data/mt940.txt&AUDIT_LOG=/data/maps/finplus/cbpr_plus/mx-mt/data/audit_msg.json&return=log&audit_file=always" -H
"accept: */*"

curl -X GET "http://localhost:8080/tx-rest/v2/run/cbpr_pacs009_mt205?INPUT_FILE=./cbpr_plus/translation/mx-
mt/data/env_pacs_009_mt205cor_valid.xml&CONFIG_FILE=./cbpr_plus/translation/mx-
mt/data/trx_config.xml&OUTPUT_RESULT_MT=/data/maps/finplus/cbpr_plus/mt-
mt/data/mt205.txt&AUDIT_LOG=/data/maps/finplus/cbpr_plus/mx-mt/data/audit_msg.json&return=log&audit_file=always" -H
"accept: */*"

curl -X GET "http://localhost:8080/tx-rest/v2/run/cbpr_pacs004_mtnnnRTN?INPUT_FILE=./cbpr_plus/translation/mx-
mt/data/env_pacs_004_mt202rtn_valid.xml&CONFIG_FILE=./cbpr_plus/translation/mx-
mt/data/trx_config.xml&OUTPUT_RESULT_MT=/data/maps/finplus/cbpr_plus/mt-
mt/data/mt20nRTN.out&AUDIT_LOG=/data/maps/finplus/cbpr_plus/mx-mt/data/audit_msg.json&return=log&audit_file=always" -
H "accept: */*"

curl -X GET "http://localhost:8080/tx-rest/v2/run/cbpr_camt058_mt292?INPUT_FILE=./cbpr_plus/translation/mx-
mt/data/env_camt_058_valid.xml&OUTPUT_RESULT_MT=/data/maps/finplus/cbpr_plus/mt-
mt/data/mt292.out&CONFIG_FILE=./cbpr_plus/translation/mx-
mt/data/trx_config.xml&AUDIT_LOG=/data/maps/finplus/cbpr_plus/mx-mt/data/audit_msg.json&return=log&audit_file=always" -
H "accept: */*"

curl -X GET "http://localhost:8080/tx-rest/v2/run/cbpr_camt054_mt9n0?INPUT_FILE=./cbpr_plus/translation/mx-
mt/data/env_camt_054_crdr.xml&OUTPUT_RESULT_MT=/data/maps/finplus/cbpr_plus/mt-
mt/data/mt9n0.out&CONFIG_FILE=./cbpr_plus/translation/mx-
mt/data/trx_config.xml&AUDIT_LOG=/data/maps/finplus/cbpr_plus/mx-mt/data/audit_msg.json&return=log&audit_file=always" -
H "accept: */*"

curl -X GET "http://localhost:8080/tx-rest/v2/run/cbpr_camt057_mt210?INPUT_FILE=./cbpr_plus/translation/mx-
mt/data/env_camt_057_valid.xml&OUTPUT_RESULT_MT=/data/maps/finplus/cbpr_plus/mt-
mt/data/mt210.out&CONFIG_FILE=./cbpr_plus/translation/mx-

```

```

mt/data/trx_config.xml&AUDIT_LOG=/data/maps/finplus/cbpr_plus/mx-mt/data/audit_msg.json&return=log&audit_file=always"
-H "accept: */*"

curl -X GET "http://localhost:8080/tx-rest/v2/run/cbpr_camt109_mt112?INPUT_FILE=./cbpr_plus/translation/mx-
mt/data/env_camt_109_valid.xml&OUTPUT_RESULT_MT=/data/maps/finplus/cbpr_plus/mx-
mt/data/mt112.out&CONFIG_FILE=./cbpr_plus/translation/mx-
mt/data/trx_config.xml&AUDIT_LOG=/data/maps/finplus/cbpr_plus/mx-mt/data/audit_msg.json&return=log&audit_file=always"
-H "accept: */*"

curl -X GET "http://localhost:8080/tx-rest/v2/run/cbpr_pacs002_mtn99?INPUT_FILE=./cbpr_plus/translation/mx-
mt/data/env_pacs_002_299.xml&OUTPUT_RESULT_MT=/data/maps/finplus/cbpr_plus/mx-
mt/data/mtn99.out&CONFIG_FILE=./cbpr_plus/translation/mx-
mt/data/trx_config.xml&AUDIT_LOG=/data/maps/finplus/cbpr_plus/mx-mt/data/audit_msg.json&return=log&audit_file=always"
-H "accept: */*"

curl -X GET "http://localhost:8080/tx-rest/v2/run/cbpr_camt029_mtn96?INPUT_FILE=./cbpr_plus/translation/mx-
mt/data/env_camt_029_mt196_valid.xml&OUTPUT_RESULT_MT=/data/maps/finplus/cbpr_plus/mx-
mt/data/mtn96.out&CONFIG_FILE=./cbpr_plus/translation/mx-
mt/data/trx_config.xml&AUDIT_LOG=/data/maps/finplus/cbpr_plus/mx-mt/data/audit_msg.json&return=log&audit_file=always"
-H "accept: */*"

curl -X GET "http://localhost:8080/tx-rest/v2/run/cbpr_pacs009_mt202?INPUT_FILE=./cbpr_plus/translation/mx-
mt/data/env_pacs_009_mt202cor_valid.xml&OUTPUT_RESULT_MT=/data/maps/finplus/cbpr_plus/mx-
mt/data/mt202.out&CONFIG_FILE=./cbpr_plus/translation/mx-
mt/data/trx_config.xml&AUDIT_LOG=/data/maps/finplus/cbpr_plus/mx-mt/data/audit_msg.json&return=log&audit_file=always"
-H "accept: */*"

curl -X GET "http://localhost:8080/tx-rest/v2/run/cbpr_camt056_mtn92?INPUT_FILE=./cbpr_plus/translation/mx-
mt/data/env_camt_056_valid.xml&OUTPUT_RESULT_MT=/data/maps/finplus/cbpr_plus/mx-
mt/data/mtn92.out&CONFIG_FILE=./cbpr_plus/translation/mx-
mt/data/trx_config.xml&AUDIT_LOG=/data/maps/finplus/cbpr_plus/mx-mt/data/audit_msg.json&return=log&audit_file=always"
-H "accept: */*"

curl -X GET "http://localhost:8080/tx-rest/v2/run/cbpr_pacs008_mt103?INPUT_FILE=./cbpr_plus/translation/mx-
mt/data/env_pacs_008_103.xml&OUTPUT_RESULT_MT=/data/maps/finplus/cbpr_plus/mx-
mt/data/mt103.out&CONFIG_FILE=./cbpr_plus/translation/mx-
mt/data/trx_config.xml&AUDIT_LOG=/data/maps/finplus/cbpr_plus/mx-mt/data/audit_msg.json&return=log&audit_file=always"
-H "accept: */*"

```

- CBPR+ Validation

- MX Extended

```

curl -X GET "http://localhost:8080/tx-rest/v2/run/cbpr_validation?
INPUT_FILE=./tools/mx_service/data/cbpr_pacs_009_adv.xml&VALIDATION_TYPE=extended&REPORT_FORMAT=xml&CCY_FILE=../data/
currencycodedecimals.xml&BIC_FILE=../data/bic.xml&MXCONFIG_FILE=../data/mxconfig.xml&OUTPUT_RESULT_XML=/data/maps/fin
plus/cbpr_plus/mx_extended/data/validation_result.xml&OUTPUT_RESULT_JSON=/data/maps/finplus/cbpr_plus/mx_extended/dat
a/validation_result.json&FAILURE_LOG=/data/maps/finplus/cbpr_plus/mx_extended/data/stopMXValidation.json&return=log&
udit_file=always" -H "accept: */*"

```

- Schema Only

```

curl -X GET "http://localhost:8080/tx-rest/v2/run/cbpr_validation?
INPUT_FILE=./tools/mx_service/data/cbpr_pacs_009_adv.xml&VALIDATION_TYPE=schema&REPORT_FORMAT=json&CCY_FILE=../data/c
urrencycodedecimals.xml&BIC_FILE=../data/bic.xml&MXCONFIG_FILE=../data/mxconfig.xml&OUTPUT_RESULT_XML=/data/maps/finp
lus/cbpr_plus/mx_extended/data/validation_result.xml&OUTPUT_RESULT_JSON=/data/maps/finplus/cbpr_plus/mx_extended/data
/validation_result.json&FAILURE_LOG=/data/maps/finplus/cbpr_plus/mx_extended/data/stopMXValidation.json&return=log&a
udit_file=always" -H "accept: */*"

```

- CHAPS ENH Validation

- MX Extended

```

curl -X GET "http://localhost:8080/tx-rest/v2/run/chaps_enh_validation_flow?
INPUT_FILE=./tools/mx_service/data/chen_pacs_009_cov.xml&VALIDATION_TYPE=extended&REPORT_FORMAT=xml&CCY_FILE=../data/c
urrencycodedecimals.xml&BIC_FILE=../data/bic.xml&MXCONFIG_FILE=../data/mxconfig.xml&OUTPUT_RESULT_XML=/data/maps/fin
plus/chaps_enh/mx_extended/data/validation_result.xml&OUTPUT_RESULT_JSON=/data/maps/finplus/chaps_enh/mx_extended/dat
a/validation_result.json&FAILURE_LOG=/data/maps/finplus/chaps_enh/mx_extended/data/stopMXValidation.json&return=log&a
udit_file=always" -H "accept: */*"

```

- Schema Only

```

curl -X GET "http://localhost:8080/tx-rest/v2/run/chaps_enh_validation_flow?
INPUT_FILE=./tools/mx_service/data/chen_pacs_009_cov.xml&VALIDATION_TYPE=schema&REPORT_FORMAT=json&CCY_FILE=../data/c
urrencycodedecimals.xml&BIC_FILE=../data/bic.xml&MXCONFIG_FILE=../data/mxconfig.xml&OUTPUT_RESULT_XML=/data/maps/finp
lus/chaps_enh/mx_extended/data/validation_result.xml&OUTPUT_RESULT_JSON=/data/maps/finplus/chaps_enh/mx_extended/data
/validation_result.json&FAILURE_LOG=/data/maps/finplus/chaps_enh/mx_extended/data/stopMXValidation.json&return=log&a
udit_file=always" -H "accept: */*"

```

- CHAPS L4L Validation

- MX Extended

```

curl -X GET "http://localhost:8080/tx-rest/v2/run/chaps_141_validation_flow?
INPUT_FILE=./tools/mx_service/data/chlf_pacs_009_cov.xml&VALIDATION_TYPE=extended&REPORT_FORMAT=xml&CCY_FILE=../data/c
urrencycodedecimals.xml&BIC_FILE=../data/bic.xml&MXCONFIG_FILE=../data/mxconfig.xml&OUTPUT_RESULT_XML=/data/maps/fin
plus/chaps_141/mx_extended/data/validation_result.xml&OUTPUT_RESULT_JSON=/data/maps/finplus/chaps_141/mx_extended/dat
a/validation_result.json&FAILURE_LOG=/data/maps/finplus/chaps_141/mx_extended/data/stopMXValidation.json&return=log&a
udit_file=always" -H "accept: */*"

```

- Schema Only

```

curl -X GET "http://localhost:8080/tx-rest/v2/run/chaps_141_validation_flow?
INPUT_FILE=./tools/mx_service/data/chlf_pacs_009_cov.xml&VALIDATION_TYPE=schema&REPORT_FORMAT=json&CCY_FILE=../data/c
urrencycodedecimals.xml&BIC_FILE=../data/bic.xml&MXCONFIG_FILE=../data/mxconfig.xml&OUTPUT_RESULT_XML=/data/maps/finp
lus/chaps_141/mx_extended/data/validation_result.xml&OUTPUT_RESULT_JSON=/data/maps/finplus/chaps_141/mx_extended/data
/validation_result.json&FAILURE_LOG=/data/maps/finplus/chaps_141/mx_extended/data/stopMXValidation.json&return=log&a
udit_file=always" -H "accept: */*"

```

- **EBA E1S1 Validation**
  - MX Extended

```
curl -X GET "http://localhost:8080/tx-rest/v2/run/eba_e1s1_validation_flow?
INPUT_FILE=../tools/mx_service/data/e1s1_pacs_009.xml&VALIDATION_TYPE=extended&REPORT_FORMAT=xml&CCY_FILE=../data/currencycodedecimals.xml&BIC_FILE=../data/bic.xml&MXCONFIG_FILE=../data/mxconfig.xml&OUTPUT_RESULT_XML=/data/maps/finplus/eba_e1s1/mx_extended/data/validation_result.xml&OUTPUT_RESULT_JSON=/data/maps/finplus/eba_e1s1/mx_extended/data/validation_result.json&FAILURE_LOG=/data/maps/finplus/eba_e1s1/mx_extended/data/stopMXValidation.json&return=log&audit_file=always" -H "accept: */*"
```
  - Schema Only

```
curl -X GET "http://localhost:8080/tx-rest/v2/run/eba_e1s1_validation_flow?
INPUT_FILE=../tools/mx_service/data/e1s1_pacs_009.xml&VALIDATION_TYPE=schema&REPORT_FORMAT=json&CCY_FILE=../data/currencycodedecimals.xml&BIC_FILE=../data/bic.xml&MXCONFIG_FILE=../data/mxconfig.xml&OUTPUT_RESULT_XML=/data/maps/finplus/eba_e1s1/mx_extended/data/validation_result.xml&OUTPUT_RESULT_JSON=/data/maps/finplus/eba_e1s1/mx_extended/data/validation_result.json&FAILURE_LOG=/data/maps/finplus/eba_e1s1/mx_extended/data/stopMXValidation.json&return=log&audit_file=always" -H "accept: */*"
```
- **SWIFT UC Validation**
  - MX Extended

```
curl -X GET "http://localhost:8080/tx-rest/v2/run/swift_uc_validation_flow?
INPUT_FILE=../tools/mx_service/data/swuc_trck_001_uc.xml&VALIDATION_TYPE=extended&REPORT_FORMAT=xml&CCY_FILE=../data/currencycodedecimals.xml&BIC_FILE=../data/bic.xml&MXCONFIG_FILE=../data/mxconfig.xml&OUTPUT_RESULT_XML=/data/maps/finplus/swift_uc/mx_extended/data/validation_result.xml&OUTPUT_RESULT_JSON=/data/maps/finplus/swift_uc/mx_extended/data/validation_result.json&FAILURE_LOG=/data/maps/finplus/swift_uc/mx_extended/data/stopMXValidation.json&return=log&audit_file=always" -H "accept: */*"
```
  - Schema Only

```
curl -X GET "http://localhost:8080/tx-rest/v2/run/swift_uc_validation_flow?
INPUT_FILE=../tools/mx_service/data/swuc_trck_001_uc.xml&VALIDATION_TYPE=schema&REPORT_FORMAT=json&CCY_FILE=../data/currencycodedecimals.xml&BIC_FILE=../data/bic.xml&MXCONFIG_FILE=../data/mxconfig.xml&OUTPUT_RESULT_XML=/data/maps/finplus/swift_uc/mx_extended/data/validation_result.xml&OUTPUT_RESULT_JSON=/data/maps/finplus/swift_uc/mx_extended/data/validation_result.json&FAILURE_LOG=/data/maps/finplus/swift_uc/mx_extended/data/stopMXValidation.json&return=log&audit_file=always" -H "accept: */*"
```
- **SWIFT GPI Validation**
  - MX Extended

```
curl -X GET "http://localhost:8080/tx-rest/v2/run/swift_gpi_validation_flow?
INPUT_FILE=../tools/mx_service/data/swgp_trck_001_gCCTINST.xml&VALIDATION_TYPE=extended&REPORT_FORMAT=xml&CCY_FILE=../data/currencycodedecimals.xml&BIC_FILE=../data/bic.xml&MXCONFIG_FILE=../data/mxconfig.xml&OUTPUT_RESULT_XML=/data/maps/finplus/swift_gpi/mx_extended/data/validation_result.xml&OUTPUT_RESULT_JSON=/data/maps/finplus/swift_gpi/mx_extended/data/validation_result.json&FAILURE_LOG=/data/maps/finplus/swift_gpi/mx_extended/data/stopMXValidation.json&return=log&audit_file=always" -H "accept: */*"
```
  - Schema Only

```
curl -X GET "http://localhost:8080/tx-rest/v2/run/swift_gpi_validation_flow?
INPUT_FILE=../tools/mx_service/data/swgp_trck_001_gCCTINST.xml&VALIDATION_TYPE=schema&REPORT_FORMAT=json&CCY_FILE=../data/currencycodedecimals.xml&BIC_FILE=../data/bic.xml&MXCONFIG_FILE=../data/mxconfig.xml&OUTPUT_RESULT_XML=/data/maps/finplus/swift_gpi/mx_extended/data/validation_result.xml&OUTPUT_RESULT_JSON=/data/maps/finplus/swift_gpi/mx_extended/data/validation_result.json&FAILURE_LOG=/data/maps/finplus/swift_gpi/mx_extended/data/stopMXValidation.json&return=log&audit_file=always" -H "accept: */*"
```
- **T2 CLM Validation**
  - MX Extended

```
curl -X GET "http://localhost:8080/tx-rest/v2/run/t2_clm_validation_flow?
INPUT_FILE=../tools/mx_service/data/t2clm_pacs_009.xml&VALIDATION_TYPE=extended&REPORT_FORMAT=xml&CCY_FILE=../data/currencycodedecimals.xml&BIC_FILE=../data/bic.xml&MXCONFIG_FILE=../data/mxconfig.xml&OUTPUT_RESULT_XML=/data/maps/finplus/t2_clm/mx_extended/data/validation_result.xml&OUTPUT_RESULT_JSON=/data/maps/finplus/t2_clm/mx_extended/data/validation_result.json&FAILURE_LOG=/data/maps/finplus/t2_clm/mx_extended/data/stopMXValidation.json&return=log&audit_file=always" -H "accept: */*"
```
  - Schema Only

```
curl -X GET "http://localhost:8080/tx-rest/v2/run/t2_clm_validation_flow?
INPUT_FILE=../tools/mx_service/data/t2clm_pacs_009.xml&VALIDATION_TYPE=schema&REPORT_FORMAT=json&CCY_FILE=../data/currencycodedecimals.xml&BIC_FILE=../data/bic.xml&MXCONFIG_FILE=../data/mxconfig.xml&OUTPUT_RESULT_XML=/data/maps/finplus/t2_clm/mx_extended/data/validation_result.xml&OUTPUT_RESULT_JSON=/data/maps/finplus/t2_clm/mx_extended/data/validation_result.json&FAILURE_LOG=/data/maps/finplus/t2_clm/mx_extended/data/stopMXValidation.json&return=log&audit_file=always" -H "accept: */*"
```
- **T2 CoCo Validation**
  - MX Extended

```
curl -X GET "http://localhost:8080/tx-rest/v2/run/t2_coco_validation_flow?
INPUT_FILE=../tools/mx_service/data/t2coco_camt_009.xml&VALIDATION_TYPE=extended&REPORT_FORMAT=xml&CCY_FILE=../data/currencycodedecimals.xml&BIC_FILE=../data/bic.xml&MXCONFIG_FILE=../data/mxconfig.xml&OUTPUT_RESULT_XML=/data/maps/finplus/t2_coco/mx_extended/data/validation_result.xml&OUTPUT_RESULT_JSON=/data/maps/finplus/t2_coco/mx_extended/data/validation_result.json&FAILURE_LOG=/data/maps/finplus/t2_coco/mx_extended/data/stopMXValidation.json&return=log&audit_file=always" -H "accept: */*"
```
  - Schema Only

```
curl -X GET "http://localhost:8080/tx-rest/v2/run/t2_coco_validation_flow?
INPUT_FILE=../tools/mx_service/data/t2coco_camt_009.xml&VALIDATION_TYPE=schema&REPORT_FORMAT=json&CCY_FILE=../data/currencycodedecimals.xml&BIC_FILE=../data/bic.xml&MXCONFIG_FILE=../data/mxconfig.xml&OUTPUT_RESULT_XML=/data/maps/finplus/t2_coco/mx_extended/data/validation_result.xml&OUTPUT_RESULT_JSON=/data/maps/finplus/t2_coco/mx_extended/data/validation_result.json&FAILURE_LOG=/data/maps/finplus/t2_coco/mx_extended/data/stopMXValidation.json&return=log&audit_file=always" -H "accept: */*"
```

- T2 RTGS Validation

- MX Extended

```
curl -X GET "http://localhost:8080/tx-rest/v2/run/t2_rtgs_validation_flow?
INPUT_FILE=../tools/mx_service/data/t2rtgs_pacs_009.xml&VALIDATION_TYPE=extended&REPORT_FORMAT=xml&CCY_FILE=../data/currencycodedecimals.xml&BIC_FILE=../data/bic.xml&MXCONFIG_FILE=../data/mxconfig.xml&OUTPUT_RESULT_XML=/data/maps/finplus/t2_rtgs/mx_extended/data/validation_result.xml&OUTPUT_RESULT_JSON=/data/maps/finplus/t2_rtgs/mx_extended/data/validation_result.json&FAILURE_LOG=/data/maps/finplus/t2_rtgs/mx_extended/data/stopMXValidation.json&return=log&audit_file=always" -H "accept: */*"
```

- Schema Only

```
curl -X GET "http://localhost:8080/tx-rest/v2/run/t2_rtgs_validation_flow?
INPUT_FILE=../tools/mx_service/data/t2rtgs_pacs_009.xml&VALIDATION_TYPE=schema&REPORT_FORMAT=json&CCY_FILE=../data/currencycodedecimals.xml&BIC_FILE=../data/bic.xml&MXCONFIG_FILE=../data/mxconfig.xml&OUTPUT_RESULT_XML=/data/maps/finplus/t2_rtgs/mx_extended/data/validation_result.xml&OUTPUT_RESULT_JSON=/data/maps/finplus/t2_rtgs/mx_extended/data/validation_result.json&FAILURE_LOG=/data/maps/finplus/t2_rtgs/mx_extended/data/stopMXValidation.json&return=log&audit_file=always" -H "accept: */*"
```

- NBOR ReGIS Validation

- Schema Only

```
curl -X GET "http://localhost:8080/tx-rest/v2/run/nbor_regis_validation_flow?
INPUT_FILE=../tools/mx_service/data/rgis_pacs_009_cov.xml&VALIDATION_TYPE=schema&REPORT_FORMAT=xml&CCY_FILE=../data/currencycodedecimals.xml&BIC_FILE=../data/bic.xml&MXCONFIG_FILE=../data/mxconfig.xml&OUTPUT_RESULT_XML=/data/maps/finplus/nbor_regis/schema_only/data/validation_result.xml&OUTPUT_RESULT_JSON=/data/maps/finplus/nbor_regis/schema_only/data/validation_result.json&FAILURE_LOG=/data/maps/finplus/nbor_regis/schema_only/data/stopMXValidation.json&return=log&audit_file=always" -H "accept: */*"
```

```
curl -X GET "http://localhost:8080/tx-rest/v2/run/nbor_regis_validation_flow?
INPUT_FILE=../tools/mx_service/data/rgis_pacs_009_cov.xml&VALIDATION_TYPE=schema&REPORT_FORMAT=json&CCY_FILE=../data/currencycodedecimals.xml&BIC_FILE=../data/bic.xml&MXCONFIG_FILE=../data/mxconfig.xml&OUTPUT_RESULT_XML=/data/maps/finplus/nbor_regis/schema_only/data/validation_result.xml&OUTPUT_RESULT_JSON=/data/maps/finplus/nbor_regis/schema_only/data/validation_result.json&FAILURE_LOG=/data/maps/finplus/nbor_regis/schema_only/data/stopMXValidation.json&return=log&audit_file=always" -H "accept: */*"
```

- SIC RTGS Validation

- Schema Only

```
curl -X GET "http://localhost:8080/tx-rest/v2/run/sic_rtgs_validation_flow?
INPUT_FILE=../tools/mx_service/data/sirt_pacs_009.xml&VALIDATION_TYPE=schema&REPORT_FORMAT=xml&CCY_FILE=../data/currencycodedecimals.xml&BIC_FILE=../data/bic.xml&MXCONFIG_FILE=../data/mxconfig.xml&OUTPUT_RESULT_XML=/data/maps/finplus/sic_rtgs/schema_only/data/validation_result.xml&OUTPUT_RESULT_JSON=/data/maps/finplus/sic_rtgs/schema_only/data/validation_result.json&FAILURE_LOG=/data/maps/finplus/sic_rtgs/schema_only/data/stopMXValidation.json&return=log&audit_file=always" -H "accept: */*"
```

```
curl -X GET "http://localhost:8080/tx-rest/v2/run/sic_rtgs_validation_flow?
INPUT_FILE=../tools/mx_service/data/sirt_pacs_009.xml&VALIDATION_TYPE=schema&REPORT_FORMAT=json&CCY_FILE=../data/currencycodedecimals.xml&BIC_FILE=../data/bic.xml&MXCONFIG_FILE=../data/mxconfig.xml&OUTPUT_RESULT_XML=/data/maps/finplus/sic_rtgs/schema_only/data/validation_result.xml&OUTPUT_RESULT_JSON=/data/maps/finplus/sic_rtgs/schema_only/data/validation_result.json&FAILURE_LOG=/data/maps/finplus/sic_rtgs/schema_only/data/stopMXValidation.json&return=log&audit_file=always" -H "accept: */*"
```

- T2 Sec Validation

- Schema Only

```
curl -X GET "http://localhost:8080/tx-rest/v2/run/t2_sec_validation_flow?
INPUT_FILE=../tools/mx_service/data/t2se_admi_005.xml&VALIDATION_TYPE=schema&REPORT_FORMAT=xml&CCY_FILE=../data/currencycodedecimals.xml&BIC_FILE=../data/bic.xml&MXCONFIG_FILE=../data/mxconfig.xml&OUTPUT_RESULT_XML=/data/maps/finplus/t2_sec/schema_only/data/validation_result.xml&OUTPUT_RESULT_JSON=/data/maps/finplus/t2_sec/schema_only/data/validation_result.json&FAILURE_LOG=/data/maps/finplus/t2_sec/schema_only/data/stopMXValidation.json&return=log&audit_file=always" -H "accept: */*"
```

```
curl -X GET "http://localhost:8080/tx-rest/v2/run/t2_sec_validation_flow?
INPUT_FILE=../tools/mx_service/data/t2se_admi_005.xml&VALIDATION_TYPE=schema&REPORT_FORMAT=json&CCY_FILE=../data/currencycodedecimals.xml&BIC_FILE=../data/bic.xml&MXCONFIG_FILE=../data/mxconfig.xml&OUTPUT_RESULT_XML=/data/maps/finplus/t2_sec/schema_only/data/validation_result.xml&OUTPUT_RESULT_JSON=/data/maps/finplus/t2_sec/schema_only/data/validation_result.json&FAILURE_LOG=/data/maps/finplus/t2_sec/schema_only/data/stopMXValidation.json&return=log&audit_file=always" -H "accept: */*"
```

```
curl -X GET "http://localhost:8080/tx-rest/v2/run/t2_sec_head002_validation_flow?
INPUT_FILE=../tools/mx_service/data/t2se_head_002.xml&VALIDATION_TYPE=schema&REPORT_FORMAT=xml&CCY_FILE=../data/currencycodedecimals.xml&BIC_FILE=../data/bic.xml&MXCONFIG_FILE=../data/mxconfig.xml&OUTPUT_RESULT_XML=/data/maps/finplus/t2_sec/schema_only/data/validation_result.xml&OUTPUT_RESULT_JSON=/data/maps/finplus/t2_sec/schema_only/data/validation_result.json&FAILURE_LOG=/data/maps/finplus/t2_sec/schema_only/data/stopMXValidation.json&return=log&audit_file=always" -H "accept: */*"
```

```
curl -X GET "http://localhost:8080/tx-rest/v2/run/t2_sec_head002_validation_flow?
INPUT_FILE=../tools/mx_service/data/t2se_head_002.xml&VALIDATION_TYPE=schema&REPORT_FORMAT=json&CCY_FILE=../data/currencycodedecimals.xml&BIC_FILE=../data/bic.xml&MXCONFIG_FILE=../data/mxconfig.xml&OUTPUT_RESULT_XML=/data/maps/finplus/t2_sec/schema_only/data/validation_result.xml&OUTPUT_RESULT_JSON=/data/maps/finplus/t2_sec/schema_only/data/validation_result.json&FAILURE_LOG=/data/maps/finplus/t2_sec/schema_only/data/stopMXValidation.json&return=log&audit_file=always" -H "accept: */*"
```

## IBM MQ Client

IBM MQ Client 9.3.0.6 distribution is included with Runtime Server (ibm-itx-rs) image for supporting maps that use IBM MQ (client) adapter as data source to pull messages from IBM MQ, and as a data target to push messages to IBM MQ from the container. The MQ\_OVERRIDE\_DATA\_PATH environment variable has already been set and associated to /data/mqdata folder in the container.

Three other environment variables are defined for IBM MQ purposes to facilitate SSL encrypted communication between the MQ client and MQ server as following:

- MQSSLKEYR environment variable is set to /data/config/mqkeys
  - MQCHLLIB environment variable is set to /data/config
  - MQCHLTAB environment variable is set to AMQCLCHL.TAB
- 

## Helm Chart

Helm Chart for Runtime Server (ibm-itx-rs) is included in the distribution package for deploying the services to Kubernetes platform using Helm 3 package manager. Helm Charts are a set of manifest files that declare and configure the application upon deployment onto a Kubernetes platform like Red Hat OpenShift.

Helm Charts ease defining, installing, upgrading, and managing Kubernetes applications on a Kubernetes platform like Red Hat OpenShift. Helm is a Package manager for Kubernetes (analogous to yum and apt), Charts are the Packages (analogous to debs and rpms). Runtime Server Helm Chart (ibm-itx-rs-prod) defines and installs Runtime Server onto a Kubernetes platform.

- [\*\*ibm-itx-rs-prod Package\*\*](#)

ibm-itx-rs-prod Helm Chart Package includes the Helm Chart for Runtime Server. It defines and installs Runtime Server (ibm-itx-rs) as a Kubernetes application onto a Kubernetes platform like Red Hat OpenShift. The ibm-itx-rs-prod Helm Chart package includes the configuration, deployment, persistent volume claim manifest files defining Kubernetes resources, and the customization file, values.yaml.

---

## ibm-itx-rs-prod Package

ibm-itx-rs-prod Helm Chart Package includes the Helm Chart for Runtime Server. It defines and installs Runtime Server (ibm-itx-rs) as a Kubernetes application onto a Kubernetes platform like Red Hat OpenShift. The ibm-itx-rs-prod Helm Chart package includes the configuration, deployment, persistent volume claim manifest files defining Kubernetes resources, and the customization file, values.yaml.

- [\*\*Installing ibm-itx-rs-prod Package\*\*](#)
- [\*\*Kubectl commands\*\*](#)

Use **kubectl** commands to check on the kubernetes resources deployed in the kubernetes cluster platform like Red Hat OpenShift. On Red Hat OpenShift platform, **oc** command can be used besides **kubectl** command.

---

## Installing ibm-itx-rs-prod Package

### About this task

Installing ibm-itx-rs-prod Helm Chart package onto a Kubernetes platform involves the following steps:

### Procedure

1. Extract the ibm-itx-rs-prod Helm Chart package, ibm-itx-rs-prod-3.0.0.tgz, to a directory on the local machine.
2. Open values.yaml file and customize the configuration for ibm-itx-rs application:
  - a. Some of the configuration settings have the defaults set and some others have been left blank for the user to fill based on their Kubernetes platform requirements and environment.
  - b. Adding an image pull secret information to pull image from an image registry, storage class information to provision the persistent volume claims for persistence needs to save the map, input and output files.
  - c. Setting Runtime Server application specific configuration, for example, resource registry configuration file, log configuration for Server access, service and server logs and their log levels.
3. Deploy ibm-itx-rs-prod Helm Chart to your Kubernetes platform with the following command:

```
helm install itx-rs ibm-itx-rs-prod
```

The *itx-rs* designation in the above command is an example name that helps identify the current deployment of ibm-itx-rs-prod.

4. A successful deployment of ibm-itx-rs-prod to the Kubernetes platform displays as following:  
IBM Sterling Transformation Extender for Red Hat OpenShift deployed successfully
5. After customizations have been saved in the values.yaml, the charts can be saved as a package, ibm-itx-rs-prod-3.0.0.tgz, for future deployments using the following command where the ibm-itx-rs-prod package has been extracted earlier:

```
helm package
```

You can optionally install the future ibm-itx-rs-prod package as follows:

```
helm install itx-rs ibm-itx-rs-prod-3.0.0.tgz
```

6. To uninstall ibm-itx-rs-prod deployment from the cluster, run the following command:

```
helm uninstall itx-rs
```

---

## Kubectl commands

Use **kubectl** commands to check on the kubernetes resources deployed in the kubernetes cluster platform like Red Hat OpenShift. On Red Hat OpenShift platform, **oc** command can be used besides **kubectl** command.

Some of the useful commands are as following:

- List running pods in a specific namespace under Kubernetes cluster:

```
kubectl get pod --namespace=<namespace_name>
```

```
oc get pod --namespace=<namespace_name>
```

- Describe the kubernetes resources in a running Kubernetes cluster:

```
kubectl describe <resource_type> <resource_name>
```

- Display the log file of a running pod:

```
kubectl logs <pod_name> -n <namespace_name>
```

```
oc logs <pod_name> -n <namespace_name>
```

- List of ingress resources created in the Kubernetes cluster:

```
kubectl get ingress --namespace=<namespace_name>
```

```
oc get route --namespace=<namespace_name>
```

## Bulk copy

Runtime Server Data API allows deploying a single file to the data volume. You can deploy a collection of files organized in a directory structure included in a zip file which gets extracted under data volume. Bulk copy requires a special action header, **rs-action: unpack**, be included while invoking Data API.

Examples using **curl** command to deploy a single or a collection of files to the data volume are as following:

- Deploy a single compiled map file to the data volume under maps sub-directory:

```
curl -F "data=@/mydir/mymap.mmc" http://CONTAINER_HOSTNAME:8080/tx-rest/v1/itx/data/maps
```

- Deploy a collection of files in a zip file under data volume under maps sub-directory:

```
curl -F "data=@/mydir/myfiles.zip" -H "rs-action: unpack" "http://CONTAINER_HOSTNAME:8080/tx-rest/v1/itx/data/maps"
```

If your myfiles.zip has the following directory structure:

```
hipaa
|- validate.mmc
|- compliance.mmc
|- inputs
 |- input1.txt
 |- input2.xml
|- config
 |- hipaaconfig.txt
```

All the files in the zip file are extracted under /data/maps volume retaining the same directory structure as it is in the zip file once deployed through the Data API. It is a POST operation to create a single file or multiple files on the data volume. The Data API can be used to transfer a single file or multiple files under any directory level within the data volume.

If secure transport is enabled for the Runtime Server, then `http://` should be replaced with `https://` in the request URL and the port number should be changed from 8080 to 8443. You can map the default 8080 and 8443 default ports for HTTP and HTTPS with **-p** command line option for standalone container execution or through Helm Charts deployed under a Kubernetes cluster environment.

Transferring a single file or multiple files to the data volume is required before running a map under Runtime Server container unless the volume is prepopulated with the required files. This feature of copying a single file or multiple files in a zip file to a data volume is helpful to users running maps under cluster environments.

Runtime Server Helm Charts include provisions to enable either Ingress or Route for the Runtime Server to be accessible from external HTTP clients. Port is not mentioned in the URL when using route in OpenShift.

The example **curl** command using route name with HTTP protocol is as following:

```
curl -F "data=@/mydir/myfiles.zip" -H "rs-action: unpack" "http://ROUTE_NAME/tx-rest/v1/itx/data/maps"
```

Executing the example command above results in unpacking the myfiles.zip file after it is deployed to the ITX Runtime Server. ITX Runtime Server retains the same directory structure as in the zip file under target directory, /data/maps , while unpacking the contents of the zip file.

There are other ways to copy files to data volume, for example, **oc cp** command for OpenShift allows copying the file from local using the container identifier and vice-versa. Using Data API provides a consistent and easy to use mechanism for copying single or multiple files to data volume.

## Deployment Guidelines

- [GDPR](#)
- [CIS Benchmarks](#)

## GDPR

Users of ITX Runtime Server are responsible for ensuring their own compliance with various laws and regulations, including the European Union General Data Protection Regulation (GDPR). Users are solely responsible for obtaining advice of competent legal counsel as to the identification and interpretation of any relevant laws and

regulations that may affect the user's business and any actions the user may need to take to comply with such laws and regulations.

The products, services, and other capabilities described herein are not suitable for all user situations and may have restricted availability. HCL and IBM do not provide legal, accounting, or auditing advice or represent or warrant that its services or products will ensure that users are in compliance with any law or regulation.

For more information about GDPR, see:

- [EU GDPR Information Portal](#)
- [IBM GDPR website](#)

ITX Runtime Server enables an organization to integrate industry-based customer, supplier and business partner transactions across the enterprise. It helps automate complex transformation and validation of data between a range of different formats and standards. Securing and managing the personal data passed through the ITX Runtime Server for processing is a sole responsibility of the user. Some of the security guidelines that follow may help in securing the personal data:

- Harden the master and worker nodes in the cluster based on security benchmarks like CIS Benchmarks.
- Harden the Red Hat Enterprise Linux OS based on security benchmarks like CIS Benchmarks.
- Secure the communication with ITX Runtime Server using HTTPS protocol instead of HTTP.
- Secure the credentials of external systems accessed by ITX Runtime Server using AES-256 encryption in Resource Registry.
- Use secure options provided by the data source/target adapters where applicable to securely process the data.
- Encrypt and decrypt the data, process data through ITX Runtime Server's Cipher and OpenPGP adapters.
- Secure the transport of data from/to maps run by ITX Runtime Server with HTTPS, FTPS adapters.
- Cataloged maps would allow hiding the details of the map location, override settings of the deployed maps.
- Monitor the outputs generated by the ITX Runtime Server, remove input and output files when not required.
- Look out for security bulletins from ITX and Red Hat support teams to resolve security vulnerabilities.
- Apply the security patches when released and keep the cluster nodes up to date to avoid security threats.
- Refer ITX Knowledge Center and Red Hat OpenShift documentation on the security features available in the products.
- Scan map and input files for malware/virus before uploading to the data volume through application endpoint.
- Scan driver files and user-defined exit modules for malware/virus before uploading through the application endpoint.

## CIS Benchmarks

CIS Benchmarks are best practices for the secure configuration of a target system. CIS Benchmarks are developed through a unique consensus-based process of cybersecurity professionals and subject matter experts around the world. CIS benchmark guides are developed and accepted by government, business, industry, and academia.

CIS Benchmarks for Kubernetes and Red Hat OpenShift Container platform environments are available for download from the [CIS Benchmarks](#) website. These benchmark guides provide information on hardening the master and worker nodes in the OpenShift cluster. Guidance on hardening the Red Hat Enterprise Linux OS can be found in the [Red Hat documentation](#).

ITX Runtime Server has been hardened to a level that satisfies the IBM certification requirements for Container Software. Users are responsible for hardening of the nodes in the cluster that meets their security policies and requirements.

## Auditing

IBM License Service provides license consumption reporting and audit readiness for IBM containerized software. IBM License Service is useful to any customer that wants to view and understand the license consumption of IBM Certified Container Software running on a Red Hat OpenShift cluster. For more information about deploying IBM License Service and tracking license usage of standalone IBM Certified Container software, see [license tracking](#) page.

License audit snapshot is a record of license usage in your environment over a period of time. The audit snapshot is a compressed .zip package that includes a complete set of audit documents that certify your cumulative license usage. Audit snapshot is needed for compliance and audit purposes. For core license metrics, user is obliged to use License Service and periodically generate audit snapshots to fulfill container licensing requirements. For more information about core license metrics, see [Reporter metrics](#).

You do not need to complete any manual actions to prepare the license audit snapshot; you only need to generate it. At this point, the license audit snapshot is required to be generated at least once a quarter, and stored for two years in a location from which it could be retrieved and delivered to auditors. For more information on capturing license audit snapshot for compliance and audit purposes, see [retrieving audit](#) page .

Note: The requirements might change over time. You should always make sure to follow the latest requirements that are posted on Passport Advantage.

## Scaling

ITX Runtime Server pods run independently of each other and can serve client requests that can be load balanced across them in any pattern. This includes running ITX maps asynchronously, including scenarios when one pod instance processes the REST API POST call request to run the map and another pod subsequently processes the REST API GET call request to return the status of the map execution, even if the two pods are running on different worker nodes. To access the compiled maps, the pods are accessing shared persistent volume where the compiled maps are saved, and to access statuses of the asynchronous map executions the pods are utilizing Redis queues, which is the reason Redis is required when asynchronous map calls are made.

The number of pods running at any time can be controlled through the **replicaCount** property exposed by the product. When sufficient system resources are available and have been provisioned for the newly added pods to the deployment, it will increase the ability of the deployment to handle additional workload as more pods will be available to process the incoming REST API calls simultaneously.

In addition to supporting multiple pods in a single deployment (ITX Runtime Server instance), multiple deployments can be installed in the cluster, but they need to be installed in separate projects (namespaces), one instance per project. This provides for example the ability to run separate QA and Dev ITX Runtime Server deployments in the same cluster simultaneously.

ITX Runtime Server pods can be auto scaled based on threshold target CPU or Memory utilization on the worker nodes of the cluster. Auto scaling of ITX Runtime Server pods is disabled by default in the Helm Chart for IBM Sterling Transformation Extender for Red Hat OpenShift. Enabling autoscaling option disables **replicaCount** specified in the configuration. IBM Sterling Transformation Extender for Red Hat OpenShift supports horizontal pod autoscaling. However, it does not support vertical pod autoscaling at this time.

## Uninstalling

To uninstall ITX Runtime Server, run the following command, after replacing the <target\_namespace> placeholder with the name of the namespace from which you wish to uninstall the ITX Runtime Server deployment.

```
helm uninstall itx-rs -namespace <target_namespace>
```

## Upgrades and Rollbacks

Upgrade or Rollback activity of ITX Runtime Server instances from Helm Chart version 2.1.x or earlier has to be performed by uninstalling the current deployment and reinstalling the ITX Runtime Server Helm Chart version 3.0.0, which uses the ITX container image 11.0.1. Helm uninstall and install commands are to be used to deploy a new major version of ITX Runtime Server Helm Chart.

Whenever a Helm upgrade is made to an existing deployment with a Helm chart version of 3.x, the pods that are associated with the installation will not automatically restart - unless the change affects the actual deployment manifest. For example, an update to the image name or resource metrics will change the deployment manifest of the installation, which in turn will trigger an automatic restart of the pods. However, a basic change to a values.yaml property that is linked to the ITX config.yaml file, such as the logging level, will not trigger an automatic restart. The restart would need to take place manually, as provided by the Kubernetes scale command.

## Limitations

- ITX Runtime Server is supported on OpenShift 4.14, and on Linux 64 only. This means that any maps designed to run on other platforms, such as those using adapters supported on Windows operating systems only, will not be able run in ITX Runtime Server.
- Because of the unique characteristics of running workloads in Kubernetes environments compared to running them in on-prem environments, existing ITX maps may require adjustments before they can be run in ITX Runtime Server, especially if they rely on invoking custom tools or applications, or are making assumptions about the underlying host's operating system, network, local storage or other system-level resources.
- Any files referenced by the maps must assume /data/maps path as the map directory. If a map writes to a file without specifying its path, the file will be created in the /data/maps directory.
- Deploying multiple instances of ITX Runtime Server to the same projects (namespaces) is not supported. Deploying ITX Runtime Server to multiple projects (namespaces), with one instance per project, is supported.
- The V2 configuration API is not supported in Helm installations.

## Migration

- The migration guidelines given below are applicable for the following releases:
  - IBM Sterling Transformation Extender Runtime Server 10.0.3
  - IBM Sterling Transformation Extender for Red Hat OpenShift 10.1.1
  - IBM Sterling Transformation Extender for Red Hat OpenShift 10.1.2
- Both container and non-container variants of the Runtime REST API endpoints have been synchronized. The /itx-rs/v1 substring in the endpoints has to be replaced with /tx-rest/v1/itx when invoking V1 REST API endpoints of the Runtime Server. API synchronization allows consistent invocation of REST API endpoints across environments, thereby easing the design, development, testing, deployment, and maintenance of user applications.
- Swagger 2.0 documentation is no longer provided for REST API endpoints. Only OpenAPI 3.0 documentation is provided for REST API endpoints. By default, deployed map and flow endpoints of V2 REST API are shown to users in the Swagger web UI page if rest.defaultSwagger property is not set to v1.
- Environment variables that were supported to configure the container deployment are no longer supported. All configuration settings have to be specified through an overridden config.yaml file if the default value is not suitable for your deployed environment. It is mandatory to accept the license for running the product successfully. The environment variable, ITX\_RS\_LICENSE\_ACCEPT, is indirectly set during a Helm install based on the license parameter, which should only be set to true after reading through the license terms and conditions.
- The default map extension expected by the Runtime Server has been changed from lnx to mmc. Compiled maps with .lnx extension should be renamed to .mmc extension before deploying them to the Runtime Server. If the .lnx extension is still desired, override the rest.mapFileExtension setting from mmc to lnx.

Tip:

Design Server generates multi-platform composite maps. Users may opt to generate multi-platform composite maps in Design Studio, by selecting Windows and Linux platforms under Window > Preferences > Transformation Extender > Map > Build Options panel. Multi-platform composite maps get .mmc extension. You may also change the default extension of the compiled map for a specific platform under Window > Preferences > Transformation Extender > Map > Compiled Map Extensions panel.

## Introduction

This introduction to the Pack for SAP Applications includes the following topics:

- [Overview Pack for SAP Applications](#)

- [Integrating information](#)
  - [Overview SAP R/3 interfaces](#)
  - [Pack for SAP Applications overview](#)
  - [Integrating information](#)
  - [Overview of SAP R/3 interfaces](#)
- 

## Pack for SAP Applications overview

The Pack for SAP Applications is a unique data transformation technology for interfacing SAP R/3 software applications with third-party applications and legacy systems. Connectivity to the SAP R/3 environment is provided for both inbound and outbound data, and is supported on the major SAP platforms. Communications are similar to SAP R/3 to SAP R/3 communication, except the components used in IBM, and the Pack for SAP Applications appear as if they are another SAP R/3 system.

The Pack for SAP Applications makes use of SAP's synchronous and asynchronous (transactional) Remote Function Calls (RFCs). This approach ensures effective and secure communication between external systems and SAP R/3. Detailed knowledge of RFCs is not required to create interfaces for SAP R/3 that are reliable, efficient, and easy to maintain. Some knowledge of Remote Communication Basis is helpful but not required.

In addition, the pack can be used with several file-based interfaces, such as SAP DXOB/DMI and EDI subsystems.

The Pack for SAP Applications is used for:

- Initial data load
  - Ongoing real-time or near-real time interfaces
- 

## Integrating information

Success with SAP R/3 depends on an ability to integrate and share information with other systems. In a typical corporate setting, a single SAP R/3 system needs to integrate with a wide range of applications. These include legacy systems, applications provided by a third-party, and other SAP systems within the enterprise.

SAP R/3 data synchronization between applications is critical, requiring the transfer of transactions and master data. Events occurring in one location, such as the receipt of a customer order, may trigger actions at other locations, such as the generation of a production order or the updating of inventory. To achieve a distributed, fully integrated SAP R/3 environment, the hardware and software infrastructure must be in place to support the flow of this vital information seamlessly between systems.

---

## Overview of SAP R/3 interfaces

The Pack for SAP Applications supports the following interfaces and technologies that include the automatic generation of metadata-specific schemas:

- Intermediate Documents
    - Application Link Enabling (ALE)
    - ALE Message Handler (AMS)
    - Electronic Data Interchange (EDI)
  - DXOB/Data Migration Interface (DMI)
  - Business Application Programming Interface (BAPI)
  - For the adapter-specific commands, see the [Adapter Commands List](#).
  - [Intermediate Documents \(IDocs\)](#)
  - [DXOB/Data Migration Interface \(DMI\)](#)
  - [Business Application Programming Interface \(BAPI\)](#)
- 

## Intermediate Documents (IDocs)

The IDocs interface consists of the definition of a data structure and the processing logic for this data structure. This interface is used to exchange business data between two different systems. SAP R/3 support for the IDoc interface includes both Electronic Data Interchange (EDI) and Application Link Enabling (ALE) technologies.

- [Application Link Enabling \(ALE\)](#)
  - [ALE Message Handler \(AMS\)](#)
  - [Electronic Data Interchange \(EDI\)](#)
- 

## Application Link Enabling (ALE)

ALE links multiple SAP R/3 systems in distributed environments, and non-SAP systems to SAP R/3 without file-to-file transfers. The SAP R/3 architecture supports the transfer of data between applications using messaging as opposed to files. SAP R/3 enables the creation of a distributed, fully integrated suite of loosely coupled applications.

The SAP component supports the integration of distributed SAP R/3 systems under ALE and the flow of information among multiple SAP R/3 systems to accomplish specific business objectives. It also embraces integration among non-SAP R/3 applications, such as third-party applications or legacy systems, by providing the conversion technology required to accomplish this level of integration among diverse systems.

The ALE adapter (JALE) adds Unicode character set data support inbound to and outbound from SAP.

## ALE Message Handler (AMS)

The AMS interface provides connectivity between SAP R/3 and one or more disparate applications sending and receiving IDocs for one or more SAP R/3 instances. AMS does not provide any conversion function and delivers IDocs without changing them. Transactional RFC must be the communication method used between the SAP R/3 system and the ALE Message Handler.

## Electronic Data Interchange (EDI)

EDI enables companies with the ability of communicating business transactions and documents electronically with their partners, vendors, and clients.

## DXOB/Data Migration Interface (DMI)

DXOBs are SAP R/3 business objects that can be transferred automatically into the SAP R/3 system. Examples of business objects are organizational units or master data. The Pack for SAP Applications support for the DXOB interface includes data transfer object structures in both beta and released formats. During data transfer, data is transferred from an external system into the SAP R/3 system.

SAP applications support the data transfer of numerous SAP business objects. The data transfer program specifies the data format definition that is necessary to import the data into the SAP R/3 system. The pack provides the data transformation server to convert the external system data format to the format of the DX object.

## Business Application Programming Interface (BAPI)

The BAPI adapter (JBAPI) enables full Unicode support when installed with the latest version of ITX. The JBAPI adapter provides unified access to both Unicode and non-Unicode systems. BAPI provides a business-oriented interface for accessing SAP R/3 business processes and data from external systems. BAPIs are part of the SAP business framework, an open component-based architecture enabling software components from SAP and other providers to be integrated.

The Pack for SAP Applications supports the ability to call a synchronous BAPI in the mapping rule of an output card. Data returned from the BAPI can be mapped to other output data objects, or can be used for conditional logic. The SAP R/3 adapters support BAPI as well as any remote enabled function module (RFC).

## Setting up your SAP R/3 environment

The system requirements for installing and running SAP R/3 on Windows and UNIX platforms are detailed in the following sections.

Installation and use of the SAP Stand-Alone Gateway and RFC destination activation is discussed. There is also a description of how to set up your SAP R/3 environment to successfully send and receive data.

- [System requirements and installation](#)
- [Pack for SAP R/3 Server](#)
- [SAP Gateway](#)
- [Configuring the SAP R/3 system](#)

## System requirements and installation

For details about the minimum system requirements and instructions for installing or removing your Windows SAP R/3 product refer the [Release notes for IBM Transformation Extender Pack for SAP Applications V10.2.0.1](#).

IBM is installed on a Microsoft Windows platform and contains the product components necessary for establishing a build-time environment for the development of maps. In addition to the Design Server, there are several Pack for SAP Applications product components that are installed that facilitate the development of SAP R/3 interfaces:

- Examples for each of the SAP R/3 interfaces supported by the Pack for SAP Applications
- Type tree importers. For information, see the platform-specific documentation.

## Pack for SAP R/3 Server

The server on which the IBM and the Pack for SAP Applications is installed is referred to as the IBM Pack for SAP R/3 Server.

For supported platforms, see the system requirements on the Support Portal.

Note: SAP JCo library 3.1.10 or later is used on all platforms. For information about downloading and installing SAP JCo, see SAP Note 1672418: Downloading the SAP JCo. Refer to the Release Notes for installation information.

## SAP Gateway

The SAP Gateway is a SAP software product available for Windows and UNIX platforms to provide network connectivity. The R/3 adapters support the Registration method for RFC destination activation. To achieve the necessary RFC destination activation, the SAP Gateway, used as a stand-alone product, must be installed. The SAP Stand-Alone Gateway provides a secure method for communications between SAP and external systems. This communication method is the communication method of choice by SAP. SAP requires the use of a stand-alone gateway for all certified ALE interfaces.

For information on how to install the SAP Gateway, refer to your SAP documentation.

## Configuring the SAP R/3 system

Your SAP R/3 system must be properly configured to send and receive data. This configuration requires:

["1. Creating a Logical System \(BD54\)"](#)

["2. Creating RFC Destination for Outbound Data \(SM59\)"](#)

["3. Creating a Distribution Model \(BD64\)"](#)

["4. Generating a Partner Profile \(BD82\)"](#)

- ["1. Creating a logical system \(BD54\)"](#)
- ["2. Creating RFC destination for outbound data \(SM59\)"](#)
- ["3. Creating a Distribution Model \(BD64\)"](#)
- ["4. Generating a Partner Profile \(BD82\)"](#)
- ["5. Manually creating partner profiles \(WE20\)"](#)

### 1. Creating a logical system (BD54)

Every SAP R/3 client used for ALE/RFC must have a base logical system (LS) associated with the SAP client.

Note: This association is typically created at installation time by the SAP Applications installation team.

Every base LS becomes the sender for outbound messages and a receiver for inbound messages.

In addition to the base LS, additional logical system(s) must be created within that SAP R/3 system for every SAP R/3 and external system used for ALE interfaces. In an inbound ALE/RFC interface, this second LS represents the sender with respect to the base LS. In an outbound ALE/RFC interface, this second LS is a receiver.

A logical system may be created for each external system that communicates with SAP R/3. The design of your maps and systems dictate what is an external system. Generally, you create an external system to correspond to each distinct non-SAP system.

Transaction **BD54** creates a logical system to represent the server system for the distribution of data to and from the SAP R/3 system.

To create a logical system

1. Enter **/nBD54** in the command field and click **Enter**.
  2. Click **Enter** in the information window, which alerts you that you are in the process of maintaining a client-independent table and that any changes that you make will have an effect on all other clients in the system.  
The Change View "Logical systems": Overview window opens, listing the logical systems that are currently defined.
  3. Click **New entries** to create the logical system (LS).
- The New Entries: Overview of Added Entries window opens.
4. Enter a name for the LS and a meaningful description. (Throughout this example, the LS name will be **CUSTOMERLS**. The name of the base LS is **E47CLNT800**.)  
Note: The naming convention you follow for the LS should be informative because it will be used in naming the RFC destination and the partner profiles.
  5. Select **CTRL + F4**.  
The Change ==...> Display dialog box opens to confirm the change.
  6. Click **Yes** to save.  
The Prompt for Workbench request dialog box opens.
7. To complete the creation of the LS, you must obtain a request number. Create a request number for each new LS added. In the **Request** field, select a value from the drop-down list and click **Enter**.
  8. Return to the Display View "Logical systems": Overview window.
  9. Choose **Save** from the Table View menu and click **Enter**.  
The LS is created.

### 2. Creating RFC destination for outbound data (SM59)

The RFC destination is used during the Communication Layer of ALE processing. When Receiver Determination identifies a tRFC communication, the RFC destination defines the physical communication to the remote destination. RFC destinations are used only for communications outbound from SAP R/3.

To create RFC destination for outbound data:

- Set up a RFC destination.
- Set gateway options to reflect the values used when installing the stand-alone gateway.
- Optionally, set tRFC options to set the specifications for connection retries in the event of communication timeouts or connection errors. This is done by specifying the number of retry attempts, as well as the interval between two consecutive attempts. Alternatively, you can suppress the background job for connection retries.

SAP R/3 has a collective error-processing feature that you can enable to do error handling through a scheduled job that runs in the background.

- [Setting up an RFC destination](#)
- [Gateway options](#)
- [tRFC options](#)
- [Collective error processing](#)

## Setting up an RFC destination

The RFC destination may be created to define the physical communication to the remote destination. The following steps refer to the design of the SAP R/3 Enterprise application.

To set up a RFC destination:

1. Enter transaction /nsm59 in the command field.  
The Display and maintain RFC destinations window opens.
2. Right-click on TCP/IP connections and click Create.  
The RFC Destination window opens.
3. Enter the options as defined in the table. The RFC Destination window propagates with your information.

Table 1. Details for the RFC Destination window

| Field           | Enter                                                                            |
|-----------------|----------------------------------------------------------------------------------|
| RFC destination | Enter the name for the RFC destination. Use the same name as the Logical System. |
| Connection type | Enter T (TCP/IP) for the Connection type.                                        |
| Description     | Enter a meaningful description in the Description field.                         |

4. Click the Special Options tab. The Trace feature is located under the Special Options tab.  
You can enable Trace for testing purposes. However, do not enable Trace in a production environment.

5. Click the Technical Settings tab. Enter the following information:

Table 2. Technical Settings tab

| Field           | Enter                                                                                                                                        |
|-----------------|----------------------------------------------------------------------------------------------------------------------------------------------|
| Activation type | Enable Registered Server Program.                                                                                                            |
| Program ID      | Enter the Program ID. This is the program ID that is specified when using the Program ID (-A) adapter command for any R/3 source (outbound). |

6. Click the Logon/Security tab. Activate or Inactivate the Security Option. Click the Save button.

## Gateway options

Gateway options are set to correspond exactly to the values used when installing the SAP Stand-Alone Gateway.

To set gateway options:

1. Enter transaction /nsm59 in the command field.  
The Display and maintain RFC destinations window opens.
2. Expand TCP/IP connections and double-click the applicable TCP/IP connection.  
The RFC Destination window opens.
3. Click the Technical settings tab. Enter the information as defined in the table.  
Note:  
The values for Gateway host and Gateway service can be symbolic names. However, specifying the IP address and service number can improve performance and reduce errors.

Table 1. Gateway options

| Field           | Enter                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |
|-----------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Gateway host    | Enter the name of your gateway host or the IP address of your gateway host. (In this example, it is <b>192.168.1.229</b> .)<br>The value that you enter in Gateway host must match the value that you specify when using the Gateway Host (-G) adapter command for any SAP R/3 source.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
| Gateway service | Enter the name of your gateway service. Gateway service is the service name created during the installation of the SAP Stand-Alone Gateway software. The default is sapgw00. Alternatively, you can use the port address instead of the gateway service name (In this example, the value is 3300.)<br>The value that you enter in Gateway service (-x) must match the value that you specify when using the Gateway Service adapter command for any SAP R/3 source.<br><br>When migrating to SAP JCo 3, in order to use the JALE, you must perform the following when using SAP Gateway Service:<br><br>If using SAP Gateway Service name sapgwXX, or sapmsXXX, a lookup in the etc/services is required: -x sapgw00. As an example, if you are using a Windows platform, you must modify the services file located in C:\WINDOWS\system32\drivers\etc. In this case, sapgwXX must be added to the services file, along with port number and protocol type:sapgw00 3300/tcp #sap gateway for sapserver1.company.com<br><br>Note: The use of port numbers instead of symbolic service names prevents the lookup. |

## tRFC options

Depending on your configuration, you may find it useful to specify automatic retries for outbound data. tRFC options (retry options) dictate the SAP R/3 behavior during communication errors or failures. If you are experiencing frequent communication time outs or connection errors (visible in SM58 tRFC Monitoring), implement these retry options.

Note: Setting tRFC options is a one-time step. You can maintain tRFC Options by clicking the tRFC button in the RFC Destination window. Set tRFC Options to specify automatic retries for outbound data. Also, SAP R/3 has a collective error-processing feature that can be enabled, which does error handling through a scheduled job that runs in the background.

To set the tRFC Options:

1. Enter transaction `/nsm59` in the command field.  
The Display and maintain RFC destinations window opens.
2. Expand TCP/IP connections and double-click the applicable TCP/IP connection. (For this example, double-click DATASTGTX1.)  
The RFC Destination window appears.
3. From the Destination menu, choose tRFC Options.  
The Transactional RFC: System Setting for Connection Error dialog box opens.
4. In the Suppress background job if conn.error field, enter the value `x` and click Continue.  
The RFC Destination window reappears.
5. From the Destination menu, choose Save.

---

## Collective error processing

SAP R/3 handles RFC errors by starting a background process that restarts the RFC until it is processed successfully. If the connection with the receiving system is broken, this process can run indefinitely. This scheme can result in a large number of background processes running in the sending system that will degrade overall performance. To minimize these effects in a production environment, use collective error processing. With collective error processing, the failed RFC will not be resubmitted immediately. Instead, a periodically scheduled background job will collect failed RFCS and restart them as a packet. This technique is applicable for SAP R/3 and TCP/IP connections.

---

### Setting collective error processing for RSARFCEX

Periodically schedule program **RSARFCEX** to process failed transmissions.

---

### Setting collective error processing using SM58

1. After the above settings are in place, you can manually select transaction **SM58**.
2. Select the appropriate criteria and click Execute.
3. Select a failed function module and select Execute LUW from the Edit menu.

---

## 3. Creating a Distribution Model (BD64)

A Distribution Model contains the specifications identifying which messages (Message Types) flow to which logical system. A Distribution Model designates the type of data to be exchanged.

You will be modeling the exchange of data types from both the Sender view and the External system view for your SAP R/3 system and for IBM.

- [Creating message types for the Distribution Model](#)

---

### Creating message types for the Distribution Model

Message types are used in a Distribution Model to represent the type of data being exchanged between your SAP R/3 system and the Pack for SAP Applications.

To create message types:

1. Enter `/nbd64` in the command field.  
The Display Distribution Model window appears.
2. Expand SUBSYSTEMS to view the list. (Typically, the default SUBSYSTEMS Distribution Model is used.)
3. Select your SAP R/3 system. (For example `E47CLNT800`.)
4. Click the Add message type button.  
The Add Message Type dialog box opens.
5. Enter the message type data for the communication from the SAP R3 system to an external system.
  - Sender: Enter the current SAP R3 logical system. (In this example, it is `E47CLNT800`.)
  - Receiver: Enter the name of the logical system that was created. (In this example, it is `CUSTOMERLS`.)
  - Message type: Enter the message type. (This example uses `DEBMAS`.)

Note: A communication model from the SAP R3 system to an external system (`CUSTOMERLS`) is established.
6. Click Enter.
7. Return to the Change Distribution Model window and continue with the process of adding another message type to complete the distribution model.
8. Select SUBSYSTEMS and then click Add message type.  
The Add message type dialog box opens.

9. Enter the message type data for communication from the external system to the SAP R/3 system:
  - Sender: Enter the Logical system for the external system. (In this example, it is CUSTOMERLS).
  - Receiver: Enter the name of the current SAP R3 logical system. (In this example, it is E47CLNT800).
  - Message type: Enter the message type. (This example uses DEBMAS).

Note: A communication model from an external system to the R3 system is established.
10. Click Enter to return to the Change Distribution Model window.
11. In the Model menu, choose Save and click Enter.

## 4. Generating a Partner Profile (BD82)

The port definition can be generated based on the RFC destination and the partner profile. The partner profile is generated based on the Distribution Model and the port definition.

To generate a partner profile:

1. Enter /nbd82 in the command field.  
The Generating partner profile window opens.
2. In the Model view field, select a distribution model view from the drop-down list. (In this example, it is SUBSYSTEMS.)
3. In the Partner system field, select a Logical System from the drop-down list. (In this example, it is CUSTOMERLS.)
4. Select Collect IDocs and transfer. (This is the default, which can be changed using transaction code **WE20**.)
5. Select Trigger by background processing so there is no overriding by express flags. (This parameter is the default, which can be changed using **WE20** transaction code.)
6. Click Execute.  
The Generating partner profile window opens listing the messages and confirming the generation of a port and partner profile.

## 5. Manually creating partner profiles (WE20)

The Distribution Model is only applicable to ALE interfaces. When using the Distribution Model, the BD82 transaction will automatically update/create partner profile definitions and create RFC port definitions as needed. However, these definitions must be created or modified manually in situations such as the following:

- When you are not able to use the Distribution Model, you must manually create your partner profile definitions (WE20) and port definitions (WE21). Reference the *SAP Online Library* for additional information.
- When using the EDI file-based method of IDoc creation, you will need to manually create partner profiles.
- When modifying the Partner Profile parameters prior to production or to create partner profiles manually.

There are situations where partner profiles must be created or modified manually.

- **Inbound parameters**

### To manually create partner profiles

1. Enter /nWE20 transaction in the command field.  
The Partner Profiles window opens.
2. From the Partners menu, select Create or Display Change to create a new profile or to modify an existing profile.
3. Enter Outbound parameters or Inbound parameters as appropriate.

In the Outbound Parameters window, you can add or change the information as required.

From transaction codes the various reports can be run.

- RSEOUT00: Dispatch collected IDocs in a batch job.
- RSEOIND: Check the successful completion of the transmission of a tRFC to the communication layer. If successfully completed, the status of the IDoc is changed.

#### Schedule report RSEOUT00

As SAP recommends, the program RSEOUT00 should be scheduled to send IDocs by using transactional RFCs. This sends IDoc packages in one transmission, using one logon, and may provide better performance than sending IDocs individually. Avoid sending individual IDocs because each transmission involves significant overhead such as the loading of programs, establishing the connection, and logging on.

When using RSEOUT00, note the size of the IDoc packages, which can be defined in the outbound parameters of the corresponding partner agreement. For optimization, the size of the data objects involved and the number of available processes at the transmitting and receiving ends must be considered. SAP R/3 generally recommends packing 2 to 20 IDocs per package for IDocs that have numerous segments (such as ALEUD and GLDCMT) and 20 to 2,000 IDocs per package for IDocs with a small number of segments.

The use of RSEOUT00 requires one or more dialog work processes to enable parallel transmissions of IDoc packages. For SAP R/3 versions 3.1i and higher, tRFC resource management support is available. SAP R/3 advises that to avoid timeouts, the number of sender-side dialog work processes should be less than or equal to the number of available processes on the receiver. See *OSS note 74141* in the *SAP R/3 Documentation* for details.

When sending IDoc packages in parallel using this method, the packages cannot be received in serial order. If serialization of IDoc package transmission is required, use a periodic process on the received packages. To do this, configure your partner profiles for background processing and select a processing interval sufficiently large enough to ensure all interdependent packages can be received. Based on their time stamp, the packages are then reordered before processing.

#### Schedule Report RBDMOIND

When outbound IDocs are passed to the communication layer successfully, they are assigned the status **data passed to port OK**. This does not mean that it was a successful tRFC transmission. The RBDMOIND report from **SE38** should be regularly started to check whether the communication was successfully completed. If

successfully completed, the status of the IDoc is changed.

The RBDMOIND report indicates whether the outbound IDocs were transmitted to the communication layer successfully and should be scheduled to run regularly.

## To schedule the RBDMOIND report

---

1. Define variants for the job.
  2. Schedule the job with the report and one variant in one step.
- 

## Inbound parameters

From the Partner profiles: Inbound parameters window, you can add or change the a partner profile information as required.

To change inbound parameters for the Partner Profile, choose the appropriate Message type and the corresponding Process code, which is linked to the Function Module.

Run report RBDAPP01 to process the background IDoc.

## Schedule report RBDAPP01

---

Processing of the background IDoc is done by report RBDAPP01.

Avoid single IDoc transmissions because significant overhead is involved (such as program loading, connection, and logon).

Inbound processing time is affected at the receiving end when IDoc packages are initially separated into separate IDocs and individual inbound IDocs are stored in the database. The application transmission control defined in the corresponding partner agreement determines if the IDoc should be processed immediately or scheduled for processing by program RBDAPP01. SAP recommends using RBDAPP01 to improve performance if immediate processing is not required.

The information provided is minimal and may not address all of your communication requirements such as information about global company code maintenance, change pointer activation, maintaining number ranges, and so on. Refer to the *SAP Online Library* for additional information.

Also, the default parameter values in the instance profile are not sufficient for high-volume ALE interfaces. Consult with your SAP Applications installation team and OSS notes for recommended changes to these values.

---

## Creating SAP connections, actions, and schemas

Use the Importer Wizard to import BAPI, IDoc, and DXOB files to automatically generate a schema.

For more information about the Importer Wizard, review the following:

- [Overview of the Importer Wizard](#)
  - [Running the Importer Wizard](#)
  - [Overview of the Importer Wizard](#)
  - [Running the Importer Wizard](#)
- 

## Overview of the Importer Wizard

The Importer Wizard uses SAP R/3 structure-specific metadata input to generate schemas containing the corresponding type definitions. The Importer Wizard uses a series of maps to convert metadata into a schemascript file. The Type Tree Maker then processes this schema script and generates a schema containing all the supported types defined in the metadata that is imported.

For example, the following illustration shows several RSEIDOC3 IDoc definition files being imported into the Importer Wizard, which automatically generates the corresponding schema.

Schemas generated by the Importer Wizard can be used immediately for map development. Depending on the contents of the interface-specific metadata file, it may be necessary for the generated schema to be modified using the Type Designer. See each interface-specific section for information about the generated schemas.

---

## Running the Importer Wizard

When SAP metadata input contains one of the supported structures describing the data for which you want to create an interface, you are ready to run the Importer Wizard.

To run the Importer Wizard:

1. Start the Design Studio.
2. Select **File > Import**.  
The Importer Wizard opens.
3. Expand the folder.
4. Select the SAP interface (IDOC, DXOB, BAPI) for which you want to generate a schema and click **Next**.
5. The next dialog box opens, prompting you for information about the SAP system that originated the metadata file you are importing. Enter the appropriate information and click **Next**.

- The language dialog box opens.
6. Specify the national language and data character set that describe the data for map execution then click Next.
  7. The next dialog boxes that appears depends on the interface for which you are generating a schema. The Wizard presents the applicable dialog boxes for you to specify interface-specific information. For example, when generating a schema for BAPI, the SAP Connection Settings dialog box appears to enable you to specify adapter connect parameters. See each interface-specific chapter for this particular information.
  8. After entering the interface-specific information presented by the Wizard, a dialog box opens requesting the full path and file name of the metadata for which you are generating a schema.
  - Specify the metadata object for which you want to generate a schema and provide any additional information that is required by the Importer Wizard for your specific interface. For example, when generating schema from an IDoc, you must designate whether the IDoc file represents data that is for an ALE or EDI interface. See each interface-specific section for additional information.
  9. Enter the name of the schema to be generated and any additional information required by the Wizard for your specific interface, and click Next. See the interface-specific chapter for additional information.
  10. The next Importer Wizard dialog box opens showing the status of the schema being created.  
You can scroll through the schema. The Importer Wizard dialog box also shows the number of errors and warnings that have occurred when generating the schema.

---

## The R/3 adapters

The SAP specific components for the SAP R/3 runtime environment are the R/3 adapters for Windows based and UNIX based platforms.

This section introduces the R/3 adapters and provides instructions for adapter use, adapter settings, and adapter commands, including the following:

- [Overview of the R/3 adapters](#)
- [Using R/3 adapter commands](#)
- [Connection commands](#)
- [Syntax summaries for R/3 adapters](#)
- [R/3 adapter aliases](#)
- [Using R/3 system commands](#)
  
- [Overview of the R/3 adapters](#)
- [Using R/3 adapter commands](#)
- [Connection commands](#)
- [Adapter commands list](#)
- [Syntax summaries for R/3 adapters](#)
- [R/3 adapter aliases](#)
- [Using R/3 system commands](#)

---

## Overview of the R/3 adapters

Both client and server functionality is supported within an R/3 adapter. An R/3 adapter contains the Remote Function Calls (RFCs) necessary to pass inbound and outbound data to and from SAP R/3. The adapter includes transparent integration of the RFCs so that it is not necessary to possess detailed knowledge about the RFCs. The adapters also manage transactional information such as message transaction IDs (TIDs) as needed. This feature ensures seamless integration with SAP R/3 and also provides a robust auditing mechanism.

When a map has an input card that defines an R/3 adapter as the source, the adapter's server function is invoked to perform the RFCs required to ensure successful collection of the input data from the SAP R/3 system. For example, an SAP R/3 system has data that has been defined as input for a specific map executing on a Command Server. The map contains an input card that has the R/3 adapter defined as the adapter source. The R/3 adapter (server) receives the necessary RFCs to retrieve the SAP R/3 data. The adapter then passes the data through memory buffers from SAP R/3 directly into the running map for transformation into the target format.

For an output card, after a IBM map transforms the source data, the client function of the R/3 adapter is invoked, which performs the RFCs required to ensure successful delivery of the inbound data to the target on an SAP R/3 system.

The R/3 adapters used to interface with the SAP R/3 application are:

- JALE (for Unicode and non-Unicode)
- JBAPI (for Unicode and non-Unicode)
  
- [Unicode support](#)

---

## Unicode support

The JBAPI adapter is based on JCo, the SAP Java API that enables communication with SAP systems. The JBAPI adapter is fully backward-compatible with the native BAPI adapter; however the JBAPI adapter provides full Unicode support.

The JBAPI adapter supports Unicode directly and provides unified access to both Unicode and non-Unicode systems.

The JALE adapter includes Unicode character set data support inbound to, and outbound from, SAP. To enable Unicode support maps that use the JALE adapter you must use the IDoc schemas generated by the IDoc importer with the data character set option defined as Unicode.

---

## Sending Unicode data inbound to SAP from the JALE adapter

The JALE adapter is able to communicate Unicode data to an SAP R/3 system. The Unicode data format supported by the JALE adapter is UTF-16BE, Unicode big-endian. For mapping purposes it is strongly recommended you use the schemas generated by the IDoc schema importer. Using IDoc Unicode schema guarantees that data passed

to the JALE adapter has the correct syntax, recognizable by the JALE adapter. To send Unicode data to the JALE adapter use the ALE schema for the Unicode character set. The SAP R/3 system must be Unicode compliant to be able to process the Unicode data sent to it from the JALE adapter.

## Sending Unicode data outbound from SAP to the JALE adapter

---

The JALE adapter is able to receive and process Unicode data from an SAP R/3 system. The Unicode data format supported by the adapter is UTF-16BE, Unicode big-endian. Before sending Unicode data from an SAP R/3 system to an external system, the SAP RFC destination assigned to the external system must be configured correctly.

To enable the Unicode options:

1. Enter SAP transaction /nsm59.
2. Click the Special Options tab.
3. Select the Unicode option under the Character Width in Target System section.
4. Save the changes.

After saving, you will be able to send Unicode data to the selected RFC Destination. The IDoc schema that will be used to parse the JALE data source must also support Unicode data.

Use the IDoc schema importer to generate the IDoc schema. Using the IDoc Unicode schema guarantees that data passed to the JALE adapter has the correct syntax, recognizable by the JALE adapter. To send Unicode data to the JALE adapter, use the ALE schema for Unicode character set.

---

## Using R/3 adapter commands

R/3 adapter commands are used to customize the adapter's operation and can be specified for the data source of an input map card, the data target for an output map card, or both.

For more information, see Resource Adapters in the online documentation.

- [Execution command overrides](#)
- [Card settings](#)
- [R/3 data retrieval behavior for bursts](#)
- [OnFailure behavior with R/3 adapters](#)
- [Adapter commands syntax and usage](#)
- [RUN, GET, and PUT functions](#)
- [From the Map Designer](#)
- [From the Integration Flow Manager](#)

---

## Execution command overrides

Execution commands can be used, through command lines, to override settings for an existing SAP R/3 source. You can also override existing files, application, message source, or message target.

---

## Card settings

For details on card settings, see the Map Designer information in the online documentation.

The following values can be used for the source input card and target output card to enable SAP adapters:

- JALE
- JBAPI

---

## R/3 data retrieval behavior for bursts

All SAP R/3 objects (IDoc, BAPI, and DXOB) are retrieved from the adapter prior to the first burst. The server then consumes them at the rate specified by the FetchUnit setting.

---

## OnFailure behavior with R/3 adapters

When the Transaction>OnFailure input card setting equals Commit and the map does not complete successfully, the IDoc data is lost.

---

## Adapter commands syntax and usage

The following example is a command Target setting on an output card for an ALE adapter.

```
-c 800 -u ALE-USER -p IDES -h M699 -s 00 -ar3
```

Adapter commands are used in connecting to the SAP R/3 system to which the transformed data is being targeted.

Table 1. Adapter Commands Syntax and Usage

| Adapter command/value    | Interpretation                                                                                                                                                                                           |
|--------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>-c 800</code>      | Identifies the client number. Required for logging onto the SAP R/3 system.                                                                                                                              |
| <code>-u ALE-USER</code> | Identifies the user name. Required for logging onto the SAP R/3 system.                                                                                                                                  |
| <code>-p IDES</code>     | Identifies the password. Required for logging onto the SAP R/3 system.                                                                                                                                   |
| <code>-h M699</code>     | Identifies the host name or SAP route string of the SAP R/3 server to connect.                                                                                                                           |
| <code>-s 00</code>       | Identifies the system number.                                                                                                                                                                            |
| <code>-ar3</code>        | Specifies the creation of an adapter log file containing information about the transactions that occurred for this target during map execution. (In the example, <code>-ar3</code> signifies Audit R/3.) |

See the [Adapter Commands List](#) for information about the syntax and the use of each adapter command.

## RUN, GET, and PUT functions

You can use the R/3 adapter commands when using `RUN`, `GET`, and `PUT` functions when defining map rules in the Map Designer. See the *Functions and Expressions* information in the online documentation for information about the `RUN`, `GET`, and `PUT` functions.

- [RUN example](#)
- [GET example](#)
- [PUT example](#)

### RUN example

The following example shows map rules using a `RUN()` function for JALE using R/3 adapter commands to override execution commands.

```
RUN ("somemap.mmc" ,
 "-OMJALE1 ` -c remo -u " +
 userid:profile + " -p " + password:profile +
 " -h SAPh03 -s 03'")
```

In this example, a map named somemap is being executed overriding output card #1.

### GET example

A `GET()` function uses the R/3 adapter to retrieve data.

The following examples show map rules using a `GET()` function for JALE and JBAPI using R/3 adapter commands to specify the destination key for the Saprfc.ini file.

```
GET ("JALE", "-D MY_R3 -AR3")
GET ("JBAPI", "-D MY_R3 -c 800 -u IDES_USER -p initpass",
 CPACKAGE (JBAPI, "NATIVE"))
EITHER (GET ("JBAPI", "-D MY_R3-c 800 -u IDES_USER -p initpass",
 CPACKAGE (JBAPI, "NATIVE")),
 IF (0 < LASTERRORMSG(), FAIL(LASTERRORMSG())))
```

Note: The JBAPI adapter can only be used in the GET function. It cannot be used in a map input card, map output card, or PUT function.

### PUT example

A `PUT()` function sends data to the R/3 adapter.

The following example shows the `PUT()` function being used to send data to the R/3 adapter as well as enabling an adapter trace.

```
PUT ("JALE", "-c 800 -u IDES_USER -p initpass -h 127.0.0.1 -s 00 -t",
 CPACKAGE (CREMAS IDoc Input, "NATIVE"))
```

## From the Map Designer

From the Map Designer, you can use the R/3 adapter for a source or target. For example, in the output card, you can select R/3 ALE or JALE as the value for the target settings.

This example output card named **Legacy\_Data** is for a map named ALE Outbound DEBMAS.mms. The Target Command setting is composed of several R/3 adapter commands, including the required adapter commands `-c`, `-u`, `-p`, `-h`, and `-s`, which specify the necessary client, user ID, password connection information, host identifier, and system identifier. The `-ar3` adapter command specifies the creation of an adapter log file containing information about the transactions that occurred for this target during map execution.

## From the Integration Flow Manager

From the Integration Flow Manager, when you access the execution settings (either Launcher or Command Server) for a map, you can select one of the SAP R/3 adapters as a source or target.

For example, to override adapter settings of an output card or specify the SAP R/3 adapter for a target, access the execution settings for a map. Select R/3 ALE or JALE from the Target drop-down list and specify the adapter commands in the field.

This example of the Launcher settings for a map component named OrdersByDepartment shows that for **Output #1Target Command Accounting**, the R/3 adapter is specified as the target. The Target Command setting is composed of several R/3 adapter commands, including the required adapter commands for connection: **-c**, **-u**, **-p**, **-h**, **-s** that identify the client, user ID and password respectively, and **-d** which specifies the destination key for saprfc.ini that designates the additional connection information). The trace (**-t**) adapter command indicates to enable adapter trace, which creates a trace file that contains information detailing adapter activity and **-ar3**, which indicates to create the audit log.

## Connection commands

When you use an R/3 adapter, it is dependent on the SAP R/3 interface and the action that you want to specify. Many of the SAP R/3 adapter commands are required in particular situations and with specific SAP R/3 interfaces. The following topics explain situations where particular SAP R/3 adapter commands apply:

- [Required connection commands for JALE sources](#)
- [Required connection commands for JALE targets and calling a BAPI](#)
- [Optional connection commands for JALE sources and targets](#)
- [Optional connection commands for all sources and targets](#)
- [Required connection commands for JALE sources](#)
- [Required connection commands for JALE targets and for calling a BAPI](#)

## Required connection commands for JALE sources

The following adapter commands are required for a JALE source when connecting to an SAP R/3 system and the Saprfc.ini file is not used. Each adapter command is used to provide the necessary connection information required by the SAP system.

```
-U usr_id -P pwd -H host_name -S sys_num -C clnt_num
-A pgm_id -G gtwy_name -X gtwy_conn
```

The Destination adapter command is required when using the Saprfc.ini file, which contains the default connection information. The value specified with this adapter command is case sensitive.

```
-D dest_key
```

## Required connection commands for JALE targets and for calling a BAPI

The following adapter commands are required when calling a BAPI and for JALE targets for connection to a SAP system.

```
-C clnt_num -U usr_id -P pwd
```

After specifying **-c**, **-u**, and **-p** arguments, you must use one of the following three adapter command groupings, which are required to complete the connection information:

- Use these adapter commands to specify the SAP host ID and the SAP system number:  
  

```
-H host_name -S sys_num
```
- The Destination (**-d**) adapter command is required when using the Saprfc.ini file, which contains the default connection information. The value specified with this adapter command is case sensitive.  
  

```
-D dest_key
```
- This group of adapter commands is required when using the SAP R/3 load balancing principle:  
  

```
-BAL -H host_name -G grp -S sys_name
```
- The Language (**-l**) adapter command is an optional adapter command that can be used when needed for your particular connection requirements. Depending on the SAP system, this adapter command may be required for connection.  
  

```
-L lang_cd
```

## Adapter commands list

The following table lists each R/3 adapter command, the command syntax, and the interfaces for which it can be used for a source (outbound data), target (inbound data), or both.

Table 1. Adapter command list

| Name                   | Command Syntax                                     | JALE          | JBAPI  |
|------------------------|----------------------------------------------------|---------------|--------|
| Program ID             | <b>-A <i>pgm_id</i></b>                            | source        |        |
| Audit                  | <b>-AR3 [+ U] [%tid%   <i>full_path</i> ]</b>      | source/target |        |
| Backup                 | <b>-B [I] [X] [%tid%   <i>full_path</i> ]</b>      | source/target |        |
| Load Balancing         | <b>-BAL</b>                                        | target        | target |
| Bytes Per Character    | <b>-BPC</b>                                        | source/target |        |
| Client Number          | <b>-C <i>cint_num</i></b>                          | source/target | target |
| Character set encoding | <b>-enc</b>                                        | source        |        |
| Destination            | <b>-D <i>dest_key</i></b>                          | source/target | target |
| GatewayHost            | <b>-G <i>gtwy_name</i></b>                         | source        |        |
| IDoc Field Generation  | <b>-GEN [0 1] [ <i>flds</i> ]</b>                  | target        |        |
| Group                  | <b>-GROUP <i>NAME</i></b>                          | target        | target |
| Host ID                | <b>-H <i>host_name</i></b>                         | source/target | target |
| Logon Language         | <b>-L <i>lang_cd</i></b>                           | target        | target |
| Listen                 | <b>-LSN {0 1} <i>dur</i> [: <i>int</i> ]</b>       | source        |        |
| Listener Threads       | <b>-N</b>                                          | source        |        |
| Password               | <b>-P <i>pwd</i></b>                               | target        | target |
| Packet Size            | <b>-PKT <i>IDoc_qty</i></b>                        | target        |        |
| Reprocess backup files | <b>-R</b>                                          | source        |        |
| System ID              | <b>-S <i>sys_num</i></b>                           | source/target | target |
| Timeout                | <b>-timeout <i>secs</i></b>                        |               | target |
| Trace                  | <b>-T[V E N] [+ U] [%tid%   <i>full_path</i> ]</b> | source/target | target |
| Transaction ID         | <b>-TID <i>trans_ID</i></b>                        | source/target |        |
| IDoc Type              | <b>-TY OTHER\$ <i>doc_type*</i></b>                | source        |        |
| User ID                | <b>-U <i>usr_id</i></b>                            | target        | target |
| Gateway Service        | <b>V</b>                                           | source        |        |

- [Program ID \(-A\)](#)
  - [Audit \(-AR3\)](#)
  - [Backup \(-B\)](#)
  - [Load Balancing \(-BAL\)](#)
  - [Bytes Per Character \(-BPC\)](#)
- The following are optional client connection parameters. A command parameter, -BPC is used for Bytes Per Character.
- [Client Number \(-C\)](#)
  - [Character Set Encoding \(-enc\)](#)
  - [Destination \(-D\)](#)
  - [Gateway Host \(-G\)](#)
  - [IDoc Field Generation \(-GEN\)](#)
  - [Group \(-GROUP\)](#)
  - [Host ID \(-H\)](#)
  - [Logon Language \(-L\)](#)
  - [Listen \(-LSN\)](#)
  - [Listener Threads \(-N\)](#)
  - [Password \(-P\)](#)
  - [Packet Size \(-PKT\)](#)
  - [Reprocess backup files \(-R\)](#)
  - [System ID \(-S\)](#)
  - [Timeout \(-timeout\)](#)
  - [Trace \(-T\)](#)
  - [Transaction ID \(-TID\)](#)
  - [IDoc Type \(-TY\)](#)
  - [User ID \(-U\)](#)
  - [Gateway Service \(-X\)](#)

## Program ID (-A)

Use the Program ID (-A) adapter command to specify the program ID to be used for RFC activation when using the Registration method. The program ID is a unique identifier that is used by the SAP Gateway to identify the listener (the program). SAP recommends a format of machine.program for the program identifier. This identifier must match the Registration Program ID defined in SM59. SAP R/3 uses this identifier to distinguish multiple programs registered at the same SAP Gateway. For example, the program IDs **xyz.CREMAS** and **xyz.MATMAS** could both be registered at the same SAP Gateway (**-G 186.114.3.126**) and gateway number (**-X 3300**).

**-A *pgm\_id***

### Option

#### Description

*pgm\_id*

Unique program identifier.

SAP recommended format machine.program.

Has no relationship to the logical system, although a common practice is to name the key similarly to the logical system.

## Audit (-AR3)

Use the Audit (-AR3) adapter command to create a file that records the adapter activity for each specified input and output card. The default is to produce a file named m4r3adapter.log in the directory where the map is located. Optionally, you can append the audit information to an existing file, specify to use the transaction\_id for the file name, specify your own file name, incorporate the transaction\_id into your own file name, or specify a full path for the file.

**-AR3 [+][U] [%tid%|full\_path]**

### Option

| Option           | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
|------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>+</b>         | Append audit information to an existing file.                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| <b>U</b>         | For Command Server: Produce a file named <i>transaction_id.log</i> where <i>transaction_id</i> is the SAP R/3 assigned Transaction Identifier (TID) that is always in the map directory, regardless of whether u is in the input or output card of the map.<br>For Launcher: Produce a file named <i>transaction_id.log</i> where <i>transaction_id</i> is the SAP R/3 assigned Transaction Identifier (TID). If u is in the input or output card, then <i>transaction_id.log</i> is in the map directory. |
|                  | If a Transaction Identifier (TID) cannot be assigned, <i>transaction_id</i> is m4r3serial_number. When this option is used and the Backup (-B) adapter command is also specified, the name of the backup file and the audit file always match.                                                                                                                                                                                                                                                             |
| <b>%tid%</b>     | When specifying the name for the audit file, you can incorporate this literal as part of the file name and it will be replaced by the TID_number.                                                                                                                                                                                                                                                                                                                                                          |
| <b>full_path</b> | Specify the name for the audit file, which can include the directory path.                                                                                                                                                                                                                                                                                                                                                                                                                                 |

## Backup (-B)

Use the Backup (-B) adapter command to create a backup file of processed IDocs. The default is to produce a file named *transaction\_id.tid* where *transaction\_id* is the SAP R/3 assigned Transaction Identifier (TID) in the map directory.

If a Transaction Identifier (TID) cannot be assigned, *transaction\_id* is m4r3serial\_number. Optionally, you can specify that the file be input for a map, delete the file upon successful map execution, specify your own file name, incorporate the *transaction\_id* into your file name, or specify a full path for the file.

Note: When a failure occurs, a *transaction\_id.tid* backup file is created with the connection information and the IDOC (when normal usage of -B occurs only the IDOC is created).

Note: When the Audit adapter command and its U option (-AR3U) is used in conjunction with this adapter command, the name of the audit file and the backup file always match.

**Source -B[I][X] [%tid%|full\_path]  
Target -B[X] [%tid%|full\_path]**

### Option

| Option           | Description                                                                                                                                         |
|------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>I</b>         | Create backup file in temporary directory instead of using memory buffers. This option is only applicable when the ALE adapter is used as a Source. |
| <b>X</b>         | Delete the backup file from the temporary directory upon successful completion of map execution.                                                    |
| <b>%tid%</b>     | When specifying the name for the backup file, you can incorporate this literal as part of the file name and it is replaced by the TID_number.       |
| <b>full_path</b> | Specify the name for the backup file, which can include the directory path.                                                                         |

The following are examples using the Backup (-B) adapter command:

### Command

#### Result

|                         |                                                                                                                                                                                                                                                                                                  |
|-------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>-B %tid%</b>         | <i>dstx_directory\TID_number.tid</i> where <i>dstx_directory</i> is the installation directory and <i>TID_number</i> represents the TID number generated. Before using the -B adapter option with %tid%, you must first create a resource registry entry for tid and assign it a value of %tid%. |
| <b>-B d:\bak\%tid%</b>  | <i>d:\bak\TID_number.tid</i> where <i>TID_number</i> represents the TID number generated.                                                                                                                                                                                                        |
| <b>-B Archive\%tid%</b> | <i>dstx_directory\ArchiveTID_number.tid</i> where <i>dstx_directory</i> is the installation directory and <i>TID_number</i> represents the TID number that is generated.                                                                                                                         |

## Load Balancing (-BAL)

Specify the Load Balancing (-BAL) adapter command to logon with SAP R/3 load balancing. The load balancing principle allows a login to a server that will dynamically route processes based on the availability of the application servers. Using load balancing typically results in improved performance and addresses the potential for failure from connecting to one specific application server. Additional information about load balancing can be found in the SAP RFC SDK.

Note: Using the SAP R/3 load balancing principle requires the existence of a logon group in the SAP system. Consult your Basis team for assistance and information about this logon group.

**-BAL -H HOST -S R3NAME -G GRP**

**Option****Description**

|                         |                                                                              |
|-------------------------|------------------------------------------------------------------------------|
| <b>-H <i>HOST</i></b>   | Specify the server host name.                                                |
| <b>-S <i>R3NAME</i></b> | Specify the R/3 name of the system.                                          |
| <b>-G <i>GRP</i></b>    | Specify the logon group.<br>Group typically has the default value of PUBLIC. |

Note: When using **-BAL**, the values passed using **-H**, **-S**, and **-G** are always the values for **-BAL**. Therefore, never use the connection adapter commands for Host ID, System Number, or Gateway Host, which also use **-H**, **-S**, and **-G** for their values.

In addition, an entry must be added to the Services file, which is located in c:\winnt\system32\drivers\etc\services using the following syntax:

```
sapms R3NAME 36SYSNR/tcp
```

For example:

```
s apmsTSI 3600/tcp
```

The following is an example of the R/3 adapter command string for load balancing:

```
-c 800 -u JSharp -p S836GJ -bal -h MSG40 -s TSI -g PUBLIC
```

If it is necessary to connect to a gateway, you can specify **GHOST=gwhost** directly within the adapter commands.

The Saprfc.ini file also supports the use of load balancing. The following is the example distributed by SAP as a sample entry in the Saprfc.ini file.

```
/*=====
/* Type B: R/3 system - load balancing feature */
/*=====

DEST=BIN
TYPE=B
R3NAME=TSI
MSHOST=MSG40
GROUP=PUBLIC
RFC_TRACE=0
ABAP_DEBUG=0
USE_SAPGUI=0
```

The following is an example of the R/3 adapter command string to accommodate using the Saprfc.ini file with load balancing:

```
-d BIN -c 800 -u JSharp -p S836GJ
```

## Bytes Per Character (-BPC)

The following are optional client connection parameters. A command parameter, **-BPC** is used for Bytes Per Character.

The **-BPC** parameter allows the client connection to be bypassed. For example, a connection string of: "-BPC 2 -a pgm\_id -g gtwy\_name -x gtwy\_conn -t+ ar3" will use a BPC of 2 to indicate Unicode. BPC is used only if none of the parameters for Client, User, Password, System Host and Destination are present. If neither BPC, nor the other parameters are set, a default BPC value of 1 is assumed.

In the command line, where CHUPSD represents the parameters, **-C**, **-H**, **-U**, **-P**, **-S** and **-D**

- If (-BPC and any of CHUPSD): Log "BPC cannot be used with connection parameters"
- If (-BPC and none of CHUPSD): the BPC is used, no connection attempted
- If (no -BPC and any of CHUPSD): Connection is attempted to retrieve BPC
- If (no -BPC and none of CHUPSD): default BPC of 1 is used

The parameters **-C**, **-H**, **-U**, **-P**, **-S** and **-D** are now optional.

## Client Number (-C)

Use the Client Number (**-C**) adapter command to specify the client number in the SAP R/3 system to connect to. This adapter command is required for SAP connection when calling a BAPI and for all targets. Typically, this client number is specified in the SAP R/3 login window whenever you log into your R/3 system.

```
-C clnt_num
```

**Option****Description**

|                 |                           |
|-----------------|---------------------------|
| <i>clnt_num</i> | R/3 system client number. |
|-----------------|---------------------------|

## Character Set Encoding (-enc)

To receive IDoc messages from a Unicode SAP R/3 system, specify the character set encoding for SAP ALE data sources, using this command option. By default the IDoc data received from a Unicode SAP system will be encoded in UTF-16BE big-endian format. Use this command option, also to decode IDoc data into another format.

```
-enc ch_set_command
```

| <b>Option</b>               | <b>Description</b>                                                                                                                                                                                                                                                                                     |
|-----------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>ch_set_command</code> | Possible values for the character set command option are listed in the IANA Charset Registry (available from iana.org). Not all character sets listed are supported.<br>Valid character sets are those that are supported by the J2SE Java Runtime Environment (JRE).<br>Value must be a string value. |

## Destination (-D)

The Destination (-D) adapter command is required in order to specify the destination key when using the Saprfc.ini file for connection to an SAP R/3 system. SAP support of the Saprfc.ini file, which replaces the previous SIDEINFO technology, allows for RFC specific parameters to be contained in the .ini file.

`-D dest_key`

Note: The destination key has no relationship to the logical system, although a common practice is to name the key similarly to the logical system.  
The SAP provided example Saprfc.ini file contains the documentation explaining the file format.

By default, the RFC libraries look for the Saprfc.ini file in the directory specified by the RFC\_INI system environment variable. Therefore, make sure to set the environment variable RFC\_INI to specify the path and file name where the Saprfc.ini file is located. For example, if the file is located in the C:\IBM directory (RFC\_INI=c:\IBM\saprfc.ini), perform the following steps to set the system environment variable:

1. From the Start menu, select Settings > Control Panel > System.
2. Select the Environment tab.
3. Select any variable in the System Variable section. (Add your new system variable in the System Variable section, not in the User Variable section.)
4. Add RFC\_INI in the Variable field at the bottom of the window and add the location of your saprfc.ini file in the Value field.
5. Click Set > Apply > OK.

Note: Saprfc.ini entries take precedence over values that are entered in any adapter command string. ASHOST is the equivalent of the Host ID (-H) connection adapter command. SYSNR is the equivalent of the System Number (-S) connection adapter command. The value of the DEST= (-D adapter command) is case-sensitive.

6. Reboot your system.

## Example inbound -D R/3 adapter command string

The following example shows an inbound -D R/3 adapter command string with the Saprfc.ini file.

`-d R3EXAMPLE -c 800 -u userid -p pswd`

The following example is the entry in the Saprfc.ini file:

```
/*=====
/* Type A: R/3 system - specific application server */
/*=====*/
DEST=R3EXAMPLE
TYPE=A
ASHOST=sp2
SYSNR=00
RFC_TRACE=0
ABAP_DEBUG=0
USE_SAPGUI=0
```

Note: Entries are case-sensitive. The string must match the entry in the Saprfc.ini file.

Saprfc.ini entries take precedence over anything entered in any adapter command string. ASHOST is the equivalent of the Host ID (-H) connection adapter command. SYSNR is the equivalent of the System Number (-S) connection adapter command.

## Example outbound -D R/3 adapter command string

The following example shows the outbound -D R/3 adapter command string for use with the Saprfc.ini file.

`-d R3EXAMPLE -t`

The following example shows the entry in the Saprfc.ini file:

```
/*=====
/* Type R: Register a RFC server program at a SAP Gateway */
/* or connect to an already registered RFC server program */
/*=====*/
DEST=R3EXAMPLE
TYPE=R
PROGID=4handler
GWHOST=192.168.1.127
GWSERV=3300
RFC_TRACE=1
```

Note: Entries are case-sensitive. The string must match the entry in the Saprfc.ini file.

Saprfc.ini entries take precedence over anything entered in any adapter command string. PROGID is the equivalent of the name of the program registered. GWHOST is the equivalent of (-G) and GWSERV is the equivalent of (-X) in the adapter commands.

## Gateway Host (-G)

Specify the host name or SAP route string of the gateway host using the Gateway Host (-G) adapter command. When not using the Saprfc.ini file, this adapter command is required for connection purposes for ALE sources (outbound from SAP R/3).

**-G *gtwy\_name***

**Option**

**Description**  
**gtwy\_name**

Gateway host name or gateway host SAP route string.

---

## IDoc Field Generation (-GEN)

Specify automatic field generation using the IDoc Field Generation

(-GEN) adapter command. Use this adapter command to control the way IDoc fields are generated. The adapter automatically generates certain fields for an ALE client call. These fields will not be mapped because the adapter overrides them (except as noted in the options table). Separate fields in the list with a comma and omit the field list to change the behavior of all fields.

**-GEN [0|!] [*field1*[,*field2*...]]**

| Option | Option                             | Description                                                                |
|--------|------------------------------------|----------------------------------------------------------------------------|
| 0      |                                    | Specify IDoc fields to generate if blank.                                  |
| !      |                                    | Specify to suppress IDoc generation.                                       |
|        | [ <i>fld1</i> [, <i>fld2</i> ...]] | Specify the following IDoc fields as desired:                              |
|        | <b>MANDT</b>                       | as specified by -c                                                         |
|        | <b>DOCNUM</b>                      | sequentially assigned for each IDoc                                        |
|        | <b>DIRECT</b>                      | '2'                                                                        |
|        | <b>RCVPOR</b>                      | 'SAP' + remote system id                                                   |
|        | <b>RCVPRT</b>                      | 'LS'                                                                       |
|        | <b>RCVPRN</b>                      | copied from RCVPOR (possibly also generated) if not mapped by the user     |
|        | <b>SNDPOR</b>                      | 'CUSTOMERLS' if not mapped by the user                                     |
|        | <b>SNDPRT</b>                      | 'LS'                                                                       |
|        | <b>SNDPRN</b>                      | copied from SNDPOR (possibly also generated) if not mapped by the user     |
|        | <b>CREDAT</b>                      | system date                                                                |
|        | <b>CRETIM</b>                      | system time                                                                |
|        | <b>SEGNUM</b>                      | sequentially assigned for each segment                                     |
|        | <b>PSGNUM</b>                      | copied from the SEGNUM assigned to the first IDoc in the preceding HLEVEL. |
|        | <b>HLEVEL</b>                      | copied from the most recently assigned HLEVEL if not mapped by the user    |

All fields except RCVPOR and RCVPRN will be recorded in the backup file if a backup file is used. RCVPOR and RCVPRN are determined after an RFC connection is established. The control records are updated at that time.

PSGNUM generation requires that HLEVEL be specified. PSGNUM is generated by applying control-break logic to the HLEVEL field. The HLEVEL need only be entered for the first segment at each level, subsequent segments at that level can be left blank (for PSGNUM generation purposes, it is assumed to be the same). The SEGNUM for the first segment at an HLEVEL is used as the PSGNUM for segments at the next higher HLEVEL. PSGNUM for HLEVEL 01 is 000000.

The following are examples of using the IDoc field generation adapter command.

**Example**

**Description**  
**-GEN**

Automatic generation of all IDoc fields listed in the IDoc Field Generation (-GEN) topic.  
This is the default. User does not have to specify this adapter command.

Adapter automatically generates all IDoc fields regardless of the value of the fields in the input.

**-GEN0 *fld1*, *fld2***

Adapter generates IDoc fields for input field1, if it is blank and input field2, if it is blank.

**-GEN! *fld1*, *fld5***

Adapter suppresses automatic generation of those IDoc fields listed next to -GEN!

---

## Group (-GROUP)

The Group (-GROUP) adapter command is required to specify a group name to associate with the Batch Input Session. This adapter command is used for BDC targets only to pass the value directly to a Batch Input Session.

**-GROUP *NAME***

**Option**

**Description**  
**NAME**

Group name.  
Value must be specified in uppercase.

---

## Host ID (-H)

Specify the host name or SAP route string of the SAP R/3 server to which to connect using the Host ID (-H) adapter command. This adapter command may be required for connection. Use SAPLogon or the SAPGUI parameter.

**-H host\_name**

**Option**

**Description**

**host\_name**

R/3 server host name or R/3 server SAP route string.

The SAP router serves as a proxy in a secure network connection between SAP R/3 systems and external RFC systems. It is a useful extension to an existing firewall system because it allows connections through the firewall. When a SAP router is in use, a valid route string is used as the host name value. For example:

**-H /H/saprouter1/S/3300/H/saprouter2/S/3300/H/SAPappServer/S/SAPservice**

For more information about the SAP router, refer to SAP Online Library BC- Basis Components > BC-SAP Router.

---

## Logon Language (-L)

Specify the code for the logon language using the Logon Language (-L) adapter command.

**-L lang\_cd**

**Option**

**Description**

**lang\_cd**

Logon language.

---

## Listen (-LSN)

The Listen (-LSN) adapter command can be used for ALE sources. Use this adapter command to specify how long (in seconds) to wait for a message. If duration is omitted, then listen indefinitely. In addition, duration can optionally be specified as *count:interval*. For the Launcher only, when the **Listen** command is not specified, the ALE adapter will assume the **Listen** option values are set to 15:5. To implement registration method with a Command Server, the 0 option is required.

**-LSN {0 | dur[:int]}**

Table 1. Listen (-LSN) adapter command

| Option     | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
|------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>dur</b> | Specifies how long in seconds to listen for data. If the LSN command is omitted, the values default to 15:5. To listen for an indefinite period, specify 0 as the duration. Required value for registration method with a Command Server.<br>To fine-tune the listener, because the listener does not respond to service control requests while listening, specify an interval for the duration in seconds. A colon is used to separate the duration from the interval.<br><br><b>Example:</b><br>-LSN 0<br>-LSN 30:10 |

---

## Listener Threads (-N)

The Listener Threads (-N) adapter command can be used for ALE sources only when using the Launcher. Use this adapter command to specify to the listener to register multiple times at the gateway and to listen for incoming IDocs. To process large IDocs sent from the SAP ALE adapter, use this command. This adapter command may increase performance at the gateway by allocating more work processes (worker threads) to service requests. The recommended count for listener threads is a number equal to the total work processes of the SAP Application Server.

**-N cnt**

**Option**

**Description**

**cnt**

Listener thread counts.

---

## Password (-P)

Specify the password assigned to the user name for authorization for the SAP R/3 system using the Password (-P) adapter command. This adapter command is required for SAP connection for BAPI sources and all targets.

**-P pwd|@full\_path**

**Option**

**Description**

**pwd**

Password associated with the user name.

**@full\_path**

Accesses a security file, which is used to store passwords, with the specified name in the specified directory. (By default, the directory is where the map is located.)

Note: If password expiration is used in the SAP R/3 system, the maps must be updated or overridden, as appropriate, to accommodate the valid password.

## Packet Size (-PKT)

Use the Packet Size (-PKT) for output (inbound to SAP R/3) only. Specify the number of IDocs to include in a packet. A separate call is made for each packet. This option is useful for tuning the performance of the ALE interface and helps to reduce communication errors. This also eliminates the need for pre-adapters that use the **RUN()** function to break an IDoc transmission into smaller chunks.

**-PKT *IDoc\_qty***

**Option**

**Description**

*IDoc\_qty*

Number of IDocs to be included in a packet.

## Reprocess backup files (-R)

Use the Reprocess backup files (-R) command to process backup files left from the previous Launcher session. This command option is valid for data sources, and can be used only in combination with the (-BIX) command option. When this command option is specified on Launcher startup, the ALE listener will look for all backup files created in the previous Launcher session that were not processed. For each backup file, the ALE listener triggers one map execution to process the backup file IDoc data.

**-R**

## System ID (-S)

Specify the system number of the SAP R/3 system to which to connect using the System ID (-S) adapter command. This adapter command may be required for connection. Use the value as indicated for system number in SAPLogon or SAPGUI (for example, 00).

**-S *sys\_num***

**Option**

**Description**

*sys\_num*

SAP R/3 system number specified in SAPLogon or SAPGUI.

## Timeout (-timeout)

Specify the time in seconds for the adapter to wait for a response from the SAP server. If there is no response from the SAP server in the given time, a timeout error will be returned.

If no value is defined, the default value is 30 seconds (-timeout 30).

**-timeout *secs***

**Option**

**Description**

*secs*

Number in seconds that the adapter waits for a response from the SAP server.

## Trace (-T)

Use the Trace (-T) adapter command to enable the R/3 adapter trace file. By default, m4r3adapter.mtr is located in the same directory as the map, where adapter is the adapter type such as JALE or JBAPI. Optionally, you can specify to record detail information about all adapter activity or only errors. Also, you can append the trace information to an existing file or specify a name or the full path for the file.

**-T[V|E|N] [+ ] [full\_path]**

The **N** suffix disables creation of the RFC trace files.

If **N** is not set (as -T), both, the adapter and the RFC trace file are created.

If it is set (as -TN), only the adapter trace file will be created, while the RFC trace file will not.

**Option**

**Description**

**V**

Designates verbose. Detailed trace information is recorded.

**E**

Produces a trace file that contains only the errors that occurred during the map execution. If there are no errors, the trace file will not be created. When the **-TE** command is used with SAP and the map fails, the resulting trace file has an extension of .mtr unless a specific file name is designated.

- N** Disables creation of the RFC trace files.
- +** Append trace information to an existing file.

**full\_path** Creates a trace file with the specified name in the specified directory. (By default, the directory is where the map is located and the file name is *m4r3adapter.mtr*.)

---

## Transaction ID (-TID)

Use the Transaction ID (**-TID**) adapter command for ALE sources or targets. When used for a target, specify the Transaction ID (TID) for which a previously failed transmission is being resubmitted. When resubmitting a transaction, this adapter command is required so that SAP R/3 can recover properly from the previous failed attempt and avoid duplicate IDoc processing in the R/3 system.

**-TID trans\_ID**

### Option

#### Description

**trans\_ID**

Transaction ID of the previously failed transmission that is being resubmitted.

Transaction ID is a Globally Unique Identifier (GUID) that is calculated by the SAP R/3 system. It is a unique value that does not contain values to mask. The following is an example of a transaction ID:

**COA8012A004A38AC3BC60585**

Use this adapter command for a source to enable Matching Source Names to Target Names using the SAP R/3 TID number as the source wildcard value. This is useful when using the Launcher for map execution. If a source has a wildcard, any target that contains an asterisk is assigned the source wildcard value. For example, you might use **-TID \*** on the source, and a file target of **\myData\IDoc.\*** to name the destination files.

Note: For additional information, refer to the Launcher information in the online documentation.

---

## IDoc Type (-TY)

The IDoc Type (**-TY**) adapter command can be used for ALE sources and to designate the document types that can be received.

Note: When using this adapter command to designate document types, there is a potential for misconfiguration if SAP R/3 sends an IDoc type for which there is no map. If no handler exists for the IDoc, the R/3 adapter raises an RFC exception, causing the offending transaction to be held in the tRFC queue (SM58).

When using the Integration Flow Designer, a map component of a ITX system that is executed using the Launcher can use the special document-type **OTHER\$** to catch the IDocs that do not match the types designated by another map component contained in the same system definition file (.msl). In this scenario, if IDocs other than the type specified in a map component are received, rather than causing an RFC exception, a second map component with **-TY OTHER\$** defined is automatically triggered and handles the unmatched IDocs as specified.

**-TY OTHER\$ | doc\_type\***

### Option

#### Description

**OTHER\$**

When using the Launcher, use this option in a map component to handle all of the types not matched by any other trigger. When unmatched IDocs are detected, they become triggers for the OTHER\$ type instead of causing an RFC exception.

**doc\_type\***

You specify the document type that can be received. If SAP R/3 sends an IDoc that does not match what is specified, the adapter will raise an RFC exception.

Document types are case-sensitive.

Example:

**-TY DEBMAS\***

---

## User ID (-U)

Specify the SAP R/3 user name for login to the SAP R/3 system using the User ID (**-U**) adapter command. This adapter command is required for SAP R/3 system connection when calling a BAPI, and for ALE targets. The user is typically a CPIC user, but must be enabled for ALE/EDI permissions. See your Basis team, or SAP User Authorizations documentation for additional information.

**-U usr\_id|@full\_path**

### Option

#### Description

**usr\_id**

SAP R/3 user name.

**@full\_path**

Accesses a security file, which is used to store user IDs, with the specified name in the specified directory. (By default, the directory is where the map is located.)

---

## Gateway Service (-X)

Specify the service name or port number of the gateway process at the gateway host using the Gateway Service (-x) adapter command. When not using the Saprfc.ini file, this is a required adapter command for SAP connection for ALE sources (outbound from R/3).

-x *gateway\_conn*

**Option**

**Description**  
*gtwy\_conn*

Gateway process service name or port number.

---

## Syntax summaries for R/3 adapters

Syntax summaries provide a detailed list of required and optional adapter commands as they are used with the R/3 adapters. Syntax summaries use the command syntax notation.

Syntax summaries for the R/3 adapters are organized by ALE and BAPI.

- [JALE adapter commands syntax summary](#)
  - [JBAPI adapter commands syntax summaries](#)
- 

## JALE adapter commands syntax summary

The syntax of the JALE adapter commands used with data sources is as follows:

```
-A pgm_id -G gtwy_name -x gtwy_conn
 -C clnt_num -U usr_id -P pwd
 [-H host_name -S sys_num]
-D dest_key
[-AR3[+][U] [%tid%|full_path] [-B[I][X] [%tid%|full_path] [-R]]
[-TID trans_ID]
[-LSN {0|dur[:int]} [-TY OTHER$|doc_type*]]
[-T[V|E][+] [full_path]]
[-N num_of_threads] [-enc encoding]
```

The syntax of the JALE adapter commands used with data targets is as follows:

```
-C clnt_num -U usr_id -P pwd
 {-H host_name -S sys_num} -D dest_key
 -BAL -H HOST -S R3NAME -GROUP GRP [-AR3[+][U] [%tid%|full_path]
-B[I][X] [%tid%|full_path] trans_ID
[-GEN[0|1]] [fld1[,,fld2...]] [-PRT IDoc_qty]
[-L lang_cd]
[-T[V|E][+] [full_path]]
```

---

## JBAPI adapter commands syntax summaries

The syntax of the JBAPI adapter commands used with data sources is as follows:

```
-C clnt_num -U usr_id -P pwd {-H host_name -S sys_num} -D dest_key
-BAL -H HOST -S R3NAME -GROUP GRP
[-L lang_cd] [-T[V|E][+] [full_path]]
```

---

## R/3 adapter aliases

You can specify the adapter commands using an execution command string on the command line or you can create a command file that contains adapter commands that dictate the desired execution settings.

Use the execution commands **-IM** and **-OM** with the appropriate adapter alias that is specific for the utility adapter as follows:

| Adapter | Alias | As input        | As output       |
|---------|-------|-----------------|-----------------|
| JALE    | JALE  | -IMJALEcard_num | -OMJALEcard_num |
| JBAPI   | JBAPI | N/A             | N/A             |

When using an adapter alias in conjunction with the execution command, the adapter commands can be issued on the command line or in a command file. You can use the adapter commands to specify adapter functions such as specifying a particular message identifier, allowing output data to be broken up into multiple messages, or to retrieve a logical message from a source queue with a correlation identifier.

For example, to override the adapter commands defined in output card 1, the command string for the R/3 adapter might be

```
dstx testclnt -AE -WD -R0 -B -OMALE1 '-c 800 -u userid -p password -h sp2 -s 00 -t -ar3'
```

For example, an existing data source that is a file can be overridden and specified to be the R/3 adapter using the Input Source Override (-IM) execution command and the **JALE** alias. Alternatively, you can override an existing target with a different target using the Output Source Override (-OM) execution command with the correct R/3 adapter alias. For information about the options that you can use within these commands, see *Execution Commands* in the online documentation.

## Using R/3 system commands

The R/3 adapter passes any other SAP supported commands and connection parameters to the SAP R/3 system as defined in the RFC DK. These are defined in [RfcOpenEx](#). See your SAP documentation for details.

An example of using an SAP R/3 system command as an optional R/3 adapter command is specifying SAPLOGON\_ID=SAP\_logon\_key to read connect parameters from Saplogon.ini if SAPLogon is installed. After specifying the required connection R/3 adapter commands (for example, -c, -u, -p), you can use SAPLOGON\_ID= to specify the SAP\_logon\_key, which is the name defined in SAPLogon (the one that appears in the menu). Enclose the entire argument in single or double quotes so that the adapter correctly interprets it for your platform (especially if the name contains spaces).

Using this feature, the commands needed to open a connection to a SAP system such as DEST, GHOST, GSERV, ASHOST, SYSNR, MSHOST, R3NAME, GROUP, and SNC are not necessary because the RFC library obtains this information from the SAPLOGON data files. This feature provides a potential maintenance benefit for SAPLogon users and is recommended because the connection information exists in only one location (saplogon.ini), as opposed to two (saplogon.ini and Saprfc.ini).

## Intermediate Documents (IDocs)

IDocs are SAP standard data containers or formats that provide the basis for both ALE and EDI interfaces to SAP R/3. The topics below discuss using IDocs with SAP R/3 and provides the specific information required for ALE and EDI interfaces:

- [Overview of IDocs](#)
- [Generating the IDoc Parser report](#)
- [Using the Importer Wizard for IDocs](#)
- [Implementing an ALE interface](#)
- [Mapping](#)
- [Inbound and outbound processes](#)
- [Control records for IDoc mapping](#)
- [Sending EDI IDocs using ALE](#)
  
- [Overview of IDocs](#)
- [Generating the IDoc parser report](#)
- [Using the Importer Wizard for IDocs](#)
- [Implementing an ALE interface](#)
- [Mapping](#)
- [Inbound and outbound processes](#)
- [Control records for IDoc mapping](#)
- [Sending EDI IDocs using ALE](#)

The following are 3.x and 4.x control record examples to assist you in mapping to and from an IDoc.

## Overview of IDocs

Several hundred IDocs are shipped with each SAP R/3 system version 3.0. ALE and EDI interfaces use functionally equivalent IDocs; however, they differ in the manner in which the IDocs are communicated to or from SAP R/3.

- ALE IDocs are communicated through memory buffers, without intermediate files, directly to or from an RFC port using transactional remote function calls (RFCs).
- EDI IDocs are passed using an intermediate file.

These interfaces (ALE and EDI) that use the IDoc structure represent SAP's strategic approach to interfacing SAP R/3 with legacy and third-party applications where loose coupling is appropriate. Therefore, the SAP R/3 IDoc approach should be considered the first choice for developing interfaces where asynchronous, near-real time, or batch links are required. This pack supports SAP-supplied IDocs as well as user-defined IDocs.

One or more IDocs are held in a container called a packet. A packet is the largest data container in an IDoc schema. The restart attribute is automatically added to reject invalid IDocs contained within a packet.

Note: When using the Importer Wizard to create a schema in the structure of your IDocs, you have the option of selecting whether you want to include restrictions. You select with a flag that lets your map know at runtime to create restriction lists.

- [IDoc structural format](#)

## IDoc structural format

The IDoc format consists of hierarchically structured segments with multiple levels of nesting. IDocs exist for master data such as customers and materials, for transactional data such as sales orders, and for control data such as company codes. In ALE terms, these are called IDoc types, which are described as follows:

### Control record

There is one control record, and its format is identical for all IDoc types. It consists of the IDoc ID, Sender ID, Receiver ID, IDoc type, and EDI attribute.

### Data records

There can be one or more data records. A data record consists of a fixed administration section (IDoc ID, Sequence/Hierarchy) and a data part (Segment). A

Segment is the formal definition for header data and item data. The number and format of the segments can be different for each IDoc type.

### Status records

The status records describe the processing stages through which an IDoc can pass, and contain the IDoc ID and status information. There will be several status records.

The IDoc types are assigned to release-independent message types, which themselves are assigned to object types in the Business Object Repository (BOR). Customer enhancements are generally subject to naming conventions. IDoc data segments are a maximum of 1000 bytes in length and generally only contain CHAR fields. The data segments of an IDoc type are described by release-independent segment types and have release-specific segment definitions that are stored as internal structures in the Dictionary. This means IDocs can be sent with different data contents to different recipients because the recipient's release is defined in the sending system.

## Generating the IDoc parser report

The SAP:IDoc Importer is a facility for automatically generating schemas that describe the format of IDocs used in your SAP R/3 system. The Importer Wizard uses the definition of an IDoc contained in an IDoc parser report (RSEIDOC3) to automatically generate schemas from IDocs for use in mapping external data to SAP R/3 data. See [Running the Importer Wizard](#) for additional information.

After you have identified the IDoc type you want to use, use transaction WE63 or the program RSEIDOC3 to generate the IDoc parser report. RSEIDOC3 reports can be produced from a SAP R/3 system for SAP-supplied IDocs, modified SAP-supplied IDocs, or customized IDocs. The Importer Wizard works with any of these IDocs. It also works with IDocs produced from any version of a SAP R/3 system.

The IDoc definition that is input to the Importer Wizard must be in the format produced by running the RSEIDOC3 report from your R/3 system. When you run the RSEIDOC3 program, you may want to save the report to a file with the extension .ido because that is the default for IDocs when using the Importer Wizard.

Note: The definition file produced by the RSEIDOC3 report can contain definitions for one or more IDocs. If the definition file contains more than one IDoc, all of the valid IDoc definitions are created in a single schema.

Note: You must also enable the Control record (in RSEIDOC3) and Data record check boxes.

After you have produced the appropriate RSEIDOC3 report(s) from SAP R/3, download it to your PC on which the Importer Wizard is installed using the Download File option. Be sure to select Unconverted.

Note: SAP supplied IDocs may change from one SAP R/3 version to another. When you update from one version to another or when you have reduced or modified a SAP-provided IDoc, use the Importer Wizard to regenerate your schema.

## Using the Importer Wizard for IDocs

After you have produced an RSEIDOC3 report from SAP R/3 that describes the format of the IDoc or IDocs and you have moved that file to your local PC, you are ready to run the Importer Wizard. See [Running the Importer Wizard](#).

An IDoc report file produced in SAP R/3 is used to test how the Importer Wizard creates a type tree for an IDoc structure.

To run the SAP:IDoc Importer:

1. In the Type Designer, select Import from the Tree menu.
2. After the Importer Wizard starts, select SAP:IDOC as the structure for which you want to generate a type tree. Click Next.
3. Click Next until the File Name field appears in the window. In the **File Name** field, enter the file you want to import, specify the RSEIDOC3 report for which you want to generate a type tree, and select either ALE or EDI to specify the interface for which you want to generate the type tree.
  - [Understanding the IDoc schema](#)

## Understanding the IDoc schema

The RSEIDOC3 report is composed of a packet container that can hold one or more IDocs. Therefore, the schema generated represents the contents. Each IDoc contains a control record, many data records, and some status records. Each IDoc consists of, and its sequence and structure are dictated by, the sequence and structure of segments in a particular kind of IDoc.

The IDoc structure is either ALE or EDI and schemas are generated for each format appropriately.

- The ALE format is designed specifically for use with SAP's Application Linking and Enabling architecture. Each schema generated is compatible with ALE/RFC communications. It requires that each data segment in an IDoc be of the same length with no terminators and the control segment be of fixed length. Each segment is padded to a fixed length.
- The EDI format is loosely modeled after Electronic Data Interchange standards. Each schema generated is compatible with EDI/FILE-based communications. Each segment is variable-length and line-character-terminated.

The attributes of a schema correlate directly to the IDoc metadata report. The IDoc fields correspond to items in a type. For example, the field name generates the item name and the text generates the item description. IDoc data is character-based and only the text or item description is language-dependent.

The schemas generated for either ALE or EDI contain types that are identically defined named Packet, IDoc, Group\_number, Field, and Control Record. However, there are differences in how the segments are defined for the two different formats. A schema generated for ALE does not contain a Status Record type.

## Implementing an ALE interface

The SAP Application Link Enabling (ALE) technology enables data communication between two or more SAP R/3 systems and/or SAP R/3 and external (for example, legacy) systems. The SAP R/3 system must be configured to send and receive IDocs.

To implement an ALE Interface:

1. Enter transaction code /nWE63 in the SAP application to select IDoc(s) and download the metadata.
2. Use the Importer Wizard to generate the schema.

3. Use the Map Designer to create the map.

4. Configure the SAP R/3 system for inbound or outbound processing. See [Configuring the SAP R/3 system](#) in and [Setting up an RFC destination](#).

- [IDoc selection and metadata download \(WE63\)](#)
  - [Creating a Unicode metadata file](#)
- 

## IDoc selection and metadata download (WE63)

The IDoc type indicates the SAP format that is to be used to interpret the data of a business transaction.

To select a Basic IDoc:

1. Enter **/nwe63** in the command field and click **Enter**.  
The **Documentation** window opens.
  2. Enable Basic type and select from the drop-down list the IDoc type. (For this example, choose **DEBMAS03**.)
  3. Click **Parser** to execute.  
The Documentation for Basic type window opens.
  4. In the System menu, choose **List Save > Local File**.  
The Save list in file dialog box opens.
  5. Select unconverted as the format and click **Enter**.  
The Save As dialog box opens.
  6. Specify the full path in which to save the file, then click **Save** to transmit the list and save the file.
- 

## Creating a Unicode metadata file

The IDoc importer is able to generate Unicode metadata (schema with Unicode Type Names). SAP currently does not support Unicode characters for metadata names, but it does support Unicode characters for metadata descriptions. To generate schemas that can contain Unicode Type Names, generate the IDoc metadata file and save it as a Unicode file:

Note: SAP currently does not support Unicode character for metadata names, but it does support Unicode characters for metadata descriptions. To generate schemas that can contain Unicode Type Names, generate the IDoc metadata file and save it as a Unicode file:

1. Log onto the SAP R3 system using the SAP Front end/SAP logon.
  2. Go to transaction **/nwe63** and type in the name of the Basic Type (or enhancement/extension), select Control Record, Data Record and Status Record. Set the Segment release. Then click Parser (F9)
  3. Select **System > List > Save > Local File**
  4. Select Unconverted.
  5. Select Encoding as **UTF8 (4110)** and click Generate.
- 

## Mapping

After you have configured the RFC destination and created and transferred the metadata file, use the Map Designer to create the map.

- [Using the Importer Wizard for ALE](#)
  - [Creating an outbound map](#)
- 

## Using the Importer Wizard for ALE

In the Type Designer, select Import from the Tree menu to start the Importer Wizard. The Importer Wizard guides you through schema generation process. Enter the appropriate information, which includes:

- Specifying the ALE format
  - Generating the schema from the transferred metadata file
- 

## Creating an outbound map

Use the Map Designer to create a map to send the source data from SAP R/3 to the target external system.

- [Creating an input card and configuring outbound processing](#)
  - [Creating an output card and configuring inbound processing](#)
- 

## Creating an input card and configuring outbound processing

Create an input card that specifies JALE for the Source setting.

When the R/3 adapter is specified as the input card source, it performs the RFCS required to ensure successful communication of server IDoc outbound data from the R/3 system. The data is retrieved and passed through memory buffers into the executing map for transformation into the destination target format.

Note: The R/3 system configuration must have already been completed.

## To obtain an IDoc (DEBMAS) from the SAP system and pass it to the map

---

1. Enter /nbd12 in the command field and click Enter.  
The Send Customers window opens.
2. Enter Customer information in the appropriate field, then click Execute.
3. An Information window opens indicating that the master data sent was read from the database and formatted into an IDoc format. This IDoc is called a master IDoc. The master IDoc is stored in a memory buffer until the communication IDoc is generated.  
Click Enter.
4. An Information message opens indicating that the ALE service layer generated a separate IDoc from the master IDoc for each recipient who is interested in the data. These recipient-specific IDocs are called communication IDocs and are stored in the database. The recipients are determined from the Distribution Model.  
Click Enter.

## To obtain the information on the status and the data of the IDoc

---

1. Enter /nwe02 in the command field and click Enter.  
The IDoc Lists window opens.
2. Execute by clicking Enter.
3. The SAP window opens, displaying status information about the data passed to the port.  
Select an IDoc by double-clicking it.
4. In this window, you can view the Customer Master Data sent by the SAP system.  
To read the Status Records, click the individual status message.

If the partner profile is configured for Dispatch immediate, a map immediately executes for each transmitted IDoc. For example, if five IDocs are generated, five maps will execute. However, if the partner profile is configured for Collect IDocs, it is necessary to dispatch the IDocs. Do this by using transaction BD88 or by scheduling the job RSEOUT00. Maps will execute after this job completes. The number of map instances depends on the number of IDocs collected in a packet as defined in the partner profile. If the defined number of IDocs per packet is 20 and five IDocs are generated, one map will execute.

During a map execution, the status information displays: map name, status, elapsed time, input, output, and number of objects.

## Creating an output card and configuring inbound processing

---

Create a map using Map Designer to transform data from an external system to IDoc format providing inbound data to SAP R/3.

- In the output card of the map, select R/3 JALE for the Target setting and specify the adapter settings, including entering the adapter commands in the Target Command setting.
- Build and run the executable map.  
During the map execution, status information displays: map name, status, elapsed time, input, output, and number of objects.

To obtain the information on the status of the IDoc you sent into SAP:

1. Enter /nwe02 in the command field and click Enter.  
The IDoc list window opens.
2. Execute by clicking Enter.
3. Select the desired IDoc by double-clicking it.  
When inbound IDocs have been successfully posted through the communication layer, they are assigned the status **IDoc added**. When inbound IDocs have been successfully posted through the application layer, they are assigned the status **Application document posted**. If the IDoc did not go through the application layer, an error message displays.

## Inbound and outbound processes

---

This section provides a summary of possible inbound and outbound ALE scenarios describing a high-level technical overview of the communications processes between SAP R/3 and an external non-SAP R/3 system using the Pack for SAP Applications.

- [Inbound ALE to SAP](#)
- [Outbound ALE from SAP](#)

## Inbound ALE to SAP

---

The R/3 adapter can retrieve input from many different non-R/3 systems (including files, databases, MCM, web, or ERP applications) as defined by the maps executing on either the Launcher or the Command Server.

To retrieve input from a non-SAP R/3 system into an SAP R/3 system:

- As the transformation server continues to execute the map(s) as defined, output is produced and passed to the R/3 adapter.
- The R/3 adapter connects to the R/3 system using the provided connection information and user ID. SAP R/3 confirms the authorizations of the user.
- If a connection failure occurs, the adapter returns a failure code to the transformation server.
- The R/3 adapter makes the RFC calls to load the output data into the Internal Table used by the RFC libraries.
- If automatic field generation or automatic EDI to ALE conversion is enabled, the adapter produces these conversions as data is loaded into the Internal Table.
- If OnFailure setting is set to Rollback, the R/3 adapter waits for final acknowledgement from the transformation server that all output resources are ready to commit. Use of this option requires the adapter to use memory equal to two times the IDoc data size.
- Data is passed (committed) to the SAP R/3 system by packet. One packet at a time is assigned a unique TID number and processed by SAP R/3. If a communication failure occurs at this time, only packets that are not committed are in the rollback file.
- If communication failure occurs at this point, the R/3 adapter returns a failure code to the transformation server.

## Outbound ALE from SAP

The R/3 adapter can retrieve output from many different SAP R/3 systems as defined by the maps executing on either the Launcher or the Command Server.

To retrieve output from an SAP R/3 system into a non-SAP R/3 system

- One of the application modules of SAP generates a request for IDoc data. This request can be on demand by using ALE, or based on events such as those defined with Output Determination. This generates the IDoc Data.
- This request is directed to the partner profile. The configuration of the partner profile determines the port designation.
- The defined port determines whether this is a file-based communication or an ALE memory-based communication. For memory-based communications, the port designates the RFC Destination.
- The RFC Destination determines that the communication method is Registration (or Start) and also the physical parameters of the external system.
- The IDoc data is passed to the gateway. Inquiry into the IDoc logs will indicate a message Passed to Port OK.
- The communication layer connects to the external system as configured in the RFC Destination. If the communication is successful, the outbound communications start. If the communication fails, the RFC Transaction monitor (SM58) will indicate a communication error.
- For successful connections, the communication layer will connect to the SAP R/3 server functionality of the R/3 adapter. The communication layer makes the RFC calls to establish the communication, perform the TID management functions, and pass the IDoc data.
- The IDoc data is passed to the R/3 adapter by means of memory. This implies that the external system and user have adequate memory resources on the external platform. The RFC layer requires up to twice the IDoc data size so that there is sufficient memory to decompress the IDoc data. (The data is transmitted on the network in compressed form to increase network performance.) In addition, it is required to have memory equal to the data size to hold the data in memory. The data is held from the time it is received until it can be processed by the appropriate map(s). This requires careful planning if using the Launcher, especially if the maps have low throughput (or are single-threaded) because maps in the init pending state are consuming memory for staged IDocs.
- When the R/3 adapter has received the IDoc data, a backup file of IDoc data is created if the **-B** adapter option is used. The transformation server begins the validation of all input files, building temporary work files as necessary.
- The transformation server builds all output data and connects to any non-file systems via resource adapters.
- If the map completes successfully, the R/3 adapter closes the connection from SAP. If the map does not complete successfully, a rollback file of the IDoc data is created if the **OnFailure** setting is set to **Rollback** for the ALE input.

## Control records for IDoc mapping

The following are 3.x and 4.x control record examples to assist you in mapping to and from an IDoc.

- [Control record example](#)

## Control record example

| Pos | Field   | Description                                  | Value      |
|-----|---------|----------------------------------------------|------------|
| 1   | TABNAM  | Name of table structure                      | "EDI_DC40" |
| 2   | MANDT   | Client                                       |            |
| 3   | DOCNUM  | IDoc number                                  |            |
| 4   | DOCREL  | SAP Release for IDoc                         |            |
| 5   | STATUS  | Status of IDoc                               |            |
| 6   | DIRECT  | Direction                                    |            |
| 7   | OUTMOD  | Output mode                                  |            |
| 8   | EXPRSS  | Overriding in inbound processing             |            |
| 9   | TEST    | Test flag                                    |            |
| 10  | IDocTYP | Name of basic type                           | "CREMAS01" |
| 11  | CIMTYP  | Name of extension type                       |            |
| 12  | MESTYP  | Logical message type                         | "CREMAS"   |
| 13  | MESCOD  | Logical message code                         |            |
| 14  | MESFCT  | Logical message function                     |            |
| 15  | STD     | EDI standard, flag                           |            |
| 16  | STDVRS  | EDI standard, version and release            |            |
| 17  | STDMES  | EDI message type                             |            |
| 18  | SNDPOR  | Sender port (SAP system, external subsystem) |            |
| 19  | SNDPRT  | Partner type of sender                       |            |
| 20  | SNDPFC  | Partner function of sender                   |            |

| <b>Pos</b> | <b>Field</b> | <b>Description</b>                             | <b>Value</b>    |
|------------|--------------|------------------------------------------------|-----------------|
| 21         | SNDPRN       | Partner number of sender                       | BD64<br>"LSEXT" |
| 22         | SNDSAD       | Sender address (SADR)                          |                 |
| 23         | SNDLAD       | Logical address of sender                      |                 |
| 24         | RCVPOR       | Receiver port (SAP system, external subsystem) |                 |
| 25         | RCVPRT       | Partner type of recipient                      |                 |
| 26         | RCVPFC       | Partner function of recipient                  |                 |
| 27         | RCVPRN       | Partner number of recipient                    | "T90CLNT090"    |
| 28         | RCVSAD       | Recipient address (SADR)                       |                 |
| 29         | RCVLAD       | Logical address of recipient                   |                 |
| 30         | CREDAT       | Created on                                     |                 |
| 31         | CRETIM       | Time created                                   |                 |
| 32         | REFINT       | Reference to transfer (EDI interchange)        |                 |
| 33         | REFGRP       | Reference to message group (EDI message group) |                 |
| 34         | REFMES       | Reference to message (EDI message)             |                 |
| 35         | ARCKEY       | Key for (external) message archive             |                 |
| 36         | SERIAL       | Serialization field                            |                 |

## Sending EDI IDocs using ALE

To maximize performance during development and testing, always use ALE format IDocs when sending data using a JALE target. The SAP R/3 adapter will translate EDI format IDocs to ALE when necessary. This simplifies the development effort when converting file-based interfaces such as EDI to ALE.

Note: This automatic conversion feature should be used only for development and testing. It should not be used in a production environment because it severely degrades performance (by a factor of about 1,000).

## Data Transfer Objects (DXOB)

Data transfer objects (DXOB) are SAP R/3 business objects that can be transferred into an SAP R/3 system. To use SAP R/3 to create DXOB interfaces including the creation of initial data transfer files (DXOB reports) and using the Importer Wizard to generate the corresponding schemas read the sections below:

- [Overview of Data Transfer Objects](#)
- [Generating DXOB metadata](#)
- [Mapping](#)
- [Transferring the mapped data](#)
  
- [Overview of Data Transfer Objects](#)
- [Generating DXOB metadata](#)
- [Mapping](#)
- [Transferring the Mapped Data](#)

## Overview of Data Transfer Objects

DXOB (Data Transfer Object) reports are produced from SAP-supplied DXOBs or modified SAP-supplied DXOBs.

The Importer Wizard works with either of these DXOB types.

Note: SAP supplied DXOBs may change from one SAP R/3 version to another. When you upgrade SAP R/3 versions, use the Importer Wizard to regenerate your schemas accordingly.

The DXOB definition that serves as metadata input to the Importer Wizard must be in the format produced by running the DXOB report from your SAP R/3 system. When you run the DXOB program, you may save the report to a file with the extension .dx, which is the default for the Importer Wizard.

After you have produced the appropriate DXOB report(s) from SAP R/3, transfer the report file to the PC on which the Type Designer is installed.

Note: The metadata file produced by the DXOB report will contain definitions for a single DXOB.

To process DXOB interfaces, follow the steps in these topics:

1. [Generating DXOB metadata \(SXDA\)](#).
2. [Using the SAP-DXOB Importer](#) to generate the schema.
3. [Creating a map for DXOB formatted data](#)
4. [Transferring the mapped data to the application layer](#).
5. [Processing the Batch Input Session \(SM35\)](#)

## Generating DXOB metadata

SAP R/3 provides a Data Transfer Workbench (transaction code SXDA) to facilitate initial data loads into SAP R/3. The Data Transfer Workbench provides a central point of control for creating DXOB reports and starting the DX programs.

- [Data Transfer Workbench \(SXDA\)](#)

## Data Transfer Workbench (SXDA)

To generate DXOB metadata:

1. Enter /nsxda in the command field and click Enter.  
The Data Transfer Workbench window opens.
2. From the Goto menu, select DX Tools.  
The Data Transfer - Tools window opens.
3. In the Object Type field, select the DXOB from the drop-down list for which you want to generate a report. (For example, use DX object KNA1 for Customer.)
4. In the Program type field, select a program type from the drop-down list. (For example, BINP.)
5. In the Program field, select a method from the drop-down list. (For example, RFBIDE00.)
6. In the Extras menu, select Display Import Structure.  
The Documentation Dataset Record Types for Initial Data Transfer (Parser) window opens. This example shows the interface structure for Customer master data required of the DX object **0050**.
7. In the List menu, select Download to transfer this structure to a path that exists on the PC on which the Design Studio is installed.  
The Save list in file dialog opens.
8. Select unconverted and click **Enter**.  
The Save As dialog opens.
9. Specify the full path for the file that you are downloading and click Save. (In this example, the file is DXOB0050.dx.)  
The file transfer is complete.

## Mapping

After you have finished creating the DXOB report and transferred this metadata file, use Map Designer to create the map.

- [Using the SAP:DXOB Importer](#)
- [Creating a map for DXOB formatted data](#)

## Using the SAP:DXOB Importer

After you have produced a DXOB report describing the format of the desired DXOB(s) and you have transferred that file to your Design Studio PC, you are ready to run the Importer Wizard. The SAP:DXOB (Initial Data Transfer [DX]) Importer for SAP R/3 is a facility for automatically generating schemas that describe the format of Data Transfer Objects (DXOBs) used in your SAP R/3 system.

An example of the DXOB report is in the examples\packs\sap\_r3\r3\ERP\DXOB directories included in the Pack for SAP Applications installation. This DXOB report file contains the definitions for the Human Resources data transfer object. You can use this file as metadata input to run the SAP:DXOB Importer.

To run the SAP:DXOB Importer:

1. In the Type Designer, select Import from the Tree menu.
  2. Choose SAP:DXOB as the structure for which you want to generate a schema. Click Next.
  3. Continue clicking Next until the File Name field appears in the window. In the File Name field, enter the metadata file you want to import (in this example, it is DXOB0050.dx) to generate a schema.
- [DXOB formats](#)
  - [Understanding the DXOB schema](#)

## DXOB formats

The SAP Data Transfer Object (DXOB) structure is available in two different formats:

Released format

This is the standard format in version 4.0 and above. The second line of the Released format begins with **BEGIN\_DXOB 00200000** and will typically contain **IDENTVALUE** fields at the beginning of each segment.

## Understanding the DXOB schema

The DXOB metadata file produced by the DXOB report contains the definition for a single DXOB. The schema generated by the Importer Wizard will correspond accordingly. Types generated in the schema represent the following:

- Single Session
- 0 is the identifier for the record containing a Session prefix

- 1 is the identifier for the record containing Header data
  - 2 is the identifier for the record containing Document segment data
- Note: A slash (/) character indicates no data.
- 

## Creating a map for DXOB formatted data

Using the Map Designer, create a map that contains the logic required to transform input data into DXOB-formatted data.

To create a map to transform data into DXOB-formatted data:

1. Create the input card (for outbound data) that specifies the source of the input data for the map. Select File for the value of the **Source** setting and specify the card settings.
  2. Create an output card (for inbound data) that specifies the target for the output data of the map. Select File for the value of the Target setting and specify the card settings.
  3. After defining the input and output cards, enter the mapping rules to provide values for each of the fields and screens that you want to create for this transaction. Enter =NONE as the map rule for unused fields and screens.
  4. Continue with the mapping process (build, analyze, and run) until the map is producing the correct output data.
- 

## Transferring the Mapped Data

Executing the map has created the output data. The data needs to be transferred from the PC to the application layer.

- [Transferring the mapped data to the application layer](#)
  - [Creating the Batch Input Session](#)
  - [Processing the Batch Input Session \(SM35\)](#)
- 

## Transferring the mapped data to the application layer

Use the Data Transfer Workbench window to transfer the output data resulting from the map execution from the PC to the application layer.

The steps are based on the 4.7 Enterprise Application design. To transfer the mapped data from the PC to the application layer:

1. Enter /nsxda in the command field and click Enter.  
The Data Transfer Workbench window appears.
  2. From the Goto menu select DX Tools.  
The Data Transfer - Tools window opens.
  3. Click Copy.  
The Copy File window opens.
  4. Select Presentation Server and specify the name of the map output file to transfer. It is the output of the map that contains the required data for the Customer master in the format for the SAP DX object.
  5. Select Application Server and select a data type for the target from the drop-down list in the File type field.
  6. In the File Name field, enter the file name for the target on the R/3 application server in the default location.
  7. Click Enter. The file is copied to the application layer.  
The Data Transfer - Tools window opens.
  8. In the File type field, select a data type for the input file from the drop-down list.
  9. In the File Name field, enter the location of the input file on the R/3 application server.
  10. Click Display.
  11. The File & Transactions window appears which contains the names of the various segments and the data fields associated with those segments. The slash character (/) is used to transfer empty fields into SAP.
  12. Click Save.
- 

## Creating the Batch Input Session

The data created by the map has been transferred from the PC to the application layer. Now, a batch input session will be created for the .txt file now residing on the application server.

The steps above are based on the 4.7 Enterprise Application design. To create the Batch Input Session:

1. Enter /nsxda in the command field and click Enter.  
The Data Transfer Workbench window appears.
2. Click Convert data.
3. Click LSM Workbench.  
The Legacy System Migration Workbench window opens.
4. Click Execute.  
The LSM Workbench window opens.

5. In the LSM Workbench window, click User Menu.  
The User Menu dialog displays.
6. Select Create Batch Input Session and click Enter.  
The Create Batch Input Session option is now added to the LSM Workbench window.
7. Select Create Batch Input Session and click Execute.  
The Batch Input Interface for Customers window opens. The file path name defaults to customer.txt.
8. Enable Check file only.
9. In the Program menu, select Execute to check for any terminations in the batch input session.  
Note: This program can also be executed in the background by selecting Execute in Background.
10. Click Enter.
11. After the .txt file has been validated, execute the batch input interface program for the data transfer.
12. Disable the Check file only check box.
13. Execute the program by selecting Execute from the Program menu.  
The Information dialog box displays, providing status information about the batch input session. The following example shows the batch input session being created.
14. Click Enter.

---

## Processing the Batch Input Session (SM35)

The data created by the map has been transferred from the PC to the application layer. Also, a batch input session has been created for the .txt file now residing on the application server. Now you can access the Batch Input Session and process the data in the .txt file.

To process the Batch Input Sessions:

1. Enter transaction code /nsm35 to access the batch input sessions.
2. In the Session list, select the session you want to process.
3. In the Session menu, select Process session.  
The Process Session dialog box opens.
4. Select one of the options for the Run mode:
  - Process/foreground: Runs the session in the foreground, displaying every screen and field. If you change a screen in this option, the process halts.
  - Display errors only: Runs the session in the foreground, displaying only errors.
  - Background: Runs the session in the background.
5. Select Additional functions as desired. (In this example, TESTDXOB will be processed in Display errors only run mode with Dynpro standard size.)
6. Click Process.  
Note: A session can be terminated at any time by selecting System > Services > Batch Input > Delete Transaction.
7. Click Yes to save the data for the customer master data.  
The batch input session is processed and the Information dialog box opens with the process status.
8. Click Exit.  
The processing is complete. (In this example, one batch input session for each customer master data has been created.)
9. Highlight the BDC session you just processed and click Log to access the results.  
The Batch Input: Log Overview window opens.
10. Select the log and click Display to view the results.  
The Batch Input Log for Session session\_name window opens.

---

## Business Application Programming Interface (BAPI)

SAP R/3 automates implementing interface solutions for Business Application Programming Interface (BAPI) objects. This section discusses the support provided by SAP R/3 for developing BAPI interfaces. The topics discussed include the following:

- [Overview of the interface \(BAPI\)](#)
- [Mapping](#)

Note: The JB API Adapter can only be used in the GET function. It cannot be used in a map input card, map output card, or PUT function.

- [Overview of the Interface \(BAPI\)](#)
- [Mapping](#)

---

## Overview of the Interface (BAPI)

SAP Business Objects reside in the Business Framework and provide the interoperability of software components. Collectively, Business Objects are contained within the Business Object Repository (BOR). They cover a wide range of SAP R/3 business data and processes. BAPIs allow external non-SAP R/3 systems, such as products in the Pack for SAP Applications, to access the methods on a SAP Business Object. SAP Business Objects and their BAPIs provide an object-oriented view of SAP R/3 functionality.

BAPIs are used in mapping rules. They are typically used within other interface methods, which need access to the information provided by the Business Object. A developer can call a synchronous BAPI in the mapping rule of an output card as provided by the R/3 BAPI adapter support for BAPIs. The data returned from the BAPI can be mapped to other output data objects or it can be used for conditional logic. The R/3 adapter has the functionality to call a BAPI on an R/3 system. The R/3 adapter supports BAPI as well as any remote enabled function module (RFC).

BAPIs are accessed in SAP R/3 using the transaction code BAPI.

Vendor is a Business Object in the Financial Accounting area of Business Framework. Expanding the Vendor Business Object exposes the available methods.

ChangePassword is a method (BAPI) on the Vendor Business Object. This method provides the capability to change a vendor's password, which is selected by the key field VendorNo. The BAPI name of this method is found by clicking the Detail tab. The BAPI name is BAPI\_VENDOR\_CHANGE\_PASSWORD, the name you specify in the Importer Wizard.

The dialog box also provides documentation on the parameters and attributes of the BAPI. All BAPIs are composed of a combination of Importing, Exporting, and/or Table parameters. Importing parameters are input, Exporting are output, and Tables are in/out. A BAPI may have any combination of these parameter types. Documentation for the parameters, structures, and scalars of a BAPI can be found in the BAPI Browser, the SAP Assistant, and the Function Module.

## Mapping

The SAP:BAPI (Business Application Programming Interfaces) Importer is a facility for automatically generating schemas that describe the format of data used in BAPI programs in your SAP R/3 system. After you have identified a BAPI to be used as the metadata input for the Importer Wizard, generate the schema, and create the map.

To run the SAP:BAPI Importer:

1. From the Type Designer, select Import a type tree. Click OK.
  2. Choose SAP:BAPI as the structure for which you want to generate a type tree and click Next.
  3. Continue to click Next until the Adapter Command Line field appears in the window. Enter the adapter connect parameters in the Adapter Command Line field. Or, click the Configure button to display the R/3 Connection Settings dialog box in which you can enter the connection values. Click the Diagnostics tab to enable Trace for the adapter, specifying a detailed trace and a trace file name. When finished, click OK.
  4. Click Next.
  5. In the Function Module Name field, enter the name of a valid function module in SAP; or select the appropriate module by clicking the Configure button to display the BAPI Explorer. From the BAPI Explorer, expand a Business Object, select a method, and click OK. The method appears in the Function Module Name field.
  6. Click Next.
  7. The File Name window opens. Enter the name of the type tree you wish to create or select a file name. BAPI is the default for the pull-down menu list.
  8. When naming the type tree, there is an option for creating a metadata file. Enable this option to save the metadata file. Click Next. The Importer Wizard connects to the SAP R/3 system and reads the function module importing and exporting parameters. While the Importer Wizard is connecting with the SAP R/3 System, a message appears and the type tree is generated.
- [Understanding the BAPI schema](#)
  - [Calling a BAPI from a map](#)
  - [Unicode](#)

## Understanding the BAPI schema

The following shows the example schema named bapi\_creditor\_getdetail and provides an overview of the correlation between the structure of the BAPI and the schema generated by the Importer Wizard

You can use these directories to test the Importer Wizard.

- The Group type named BAPI has one component, which is Method BAPI\_CREDITOR\_GETDETAIL.
- Contained in the Method BAPI\_CREDITOR\_GETDETAIL component are the Importing Parameters, Exporting Parameters, and Tables Parameters.
- The Importing Parameters are the importing attributes CREDITORID and COMPANYCODE.
- The Exporting Parameters are the structures CREDITOR\_GENERAL\_DETAIL, CREDITOR\_COMPANY\_DETAIL, and RETURN.
- The Tables Parameter is composed of the structure CREDITOR\_BANK\_DETAIL.

## Calling a BAPI from a map

After the type tree is generated, create a map that calls the BAPI.

To call a BAPI from a map:

1. Using the Map Designer, create an output card that uses the type tree generated by the Importer Wizard, including map rules to provide values for the BAPI importing parameters.  
The object for this card is the group object BAPI of the BOR root.
2. Create a second output card that will use the **GET()** function and a **RUN()** function.  
The **GET()** function is used to call the BAPI adapter passing the importing parameters as echoed data and receiving, in return, a text blob containing the populated BAPI structure.

The card object is the xBAPI group object from the BAPI type tree.

The **RUN()** function is used to pass the exporting parameters as the data for an input card of another map.

See *Functions and Expressions* in the online documentation for more information about using the **GET()** and **RUN()** functions.

Important: If the data passed to the run map does not match the expected definition, your mapping process will not complete successfully. The input data trace file helps you to troubleshoot. To produce an input trace, add `-TI` to the options for your `RUN()` function. `-TI` should only be used for debugging purposes.

Note: BAPI examples are in the `examples\packs\sap_r3\r3\ERP\BAPI` directory included in the Pack for SAP Applications installation.

The BAPI example, `bapi_example.mms`, demonstrates the method to implement a BAPI scenario with a SAP R/3 environment. The BAPI adapter call is made from a map rule on the output card using the `GET()` function. The data returned by the BAPI call can then be used to map to any desired output. This is achieved by using the `RUN()` function to pass the data to the second map.

## Unicode

The JB API adapter is based on JCo, the SAP Java API, and is fully compatible with the previous BAPI adapter. The JB API adapter supports all the functionality of the previous BAPI adapter and provides full Unicode support.

## SAP S/4 HANA Adapter

The SAP S/4 HANA adapter provides support for executing SAP S/4 HANA Cloud APIs using OData V2 or OData V4 interfaces.

The SAP S/4 HANA adapter provides support for connecting to a compliant SAP S/4 HANA Cloud service, discovering entity sets exposed by the services and importing entity definitions in form of native JSON schemas. The adapter further supports fetching entities in managed mode, where the entity data complies with the imported schemas, as well as in raw mode where the SAP S/4 HANA Cloud service is accessed directly and the exchanged data is not constrained and validated by the adapter-provided schema.

The SAP S/4 HANA adapter extends the OData adapter. For implementation details including adapter properties and commands, please refer to: [OData Adapter - IBM Documentation](#), noting that the adapter alias used in GET and PUT rules is "SAPHANA".

## Troubleshooting

The following sections explain troubleshooting tools that are available when you encounter problems using SAP R/3 objects as data sources or targets for a map. Methods for viewing data extracted from a SAP R/3 system or loaded into an SAP R/3 system are also presented.

- [Troubleshooting tools](#)
  - [MapAudit log](#)
  - [R/3 adapter audit files](#)
  - [R/3 adapter trace files](#)
  - [R/3 return codes and error messages](#)
  - [Viewing R/3 source and target data](#)
  - [Remaining temporary data in TIDDATA Directory](#)
- 
- [Troubleshooting tools](#)
  - [MapAudit log](#)
  - [R/3 adapter audit files](#)
  - [R/3 adapter trace files](#)
  - [R/3 return codes and error messages](#)
  - [Viewing R/3 source and target data](#)
  - [Remaining temporary data in TIDDATA directory](#)

## Troubleshooting tools

If you receive an error while generating a type tree in the Importer Wizard, or run a map that uses sources and/or targets and receive a runtime error or do not get the expected output, use the following troubleshooting tools:

- Map Audit Log (`map_name.log`)
- Map Execution Trace file (`map_name.mtr`)
- Map Source and Target Data
- R3 Adapter Audit File (`m4r3adapter.log`)
- R3 Adapter Trace File (`m4r3adapter.mtr`)

## MapAudit log

If the problem encountered was the result of executing a map with sources or targets, you can produce the Execution section of the Audit Log. The MapAudit logs can be enabled from the MapSettings dialog box in the Map Designer, or Windows-based transformation server or the command line.

See Map Designer, or Command Server, in the online documentation for information about how to enable the execution log from the MapSettings dialog box. See the *Execution Command*, in the Map Designer documentation, for information about using an execution command.

The default name for the audit log is the full name of the map with a .log extension. By default, it is located in the same directory as the compiled map file.

The MapAudit can contain four different sections: BurstAudit, SummaryAudit, SettingsAudit, and AuditLocation. The sections produced depend on the Log settings for MapAudit.

- [Data Log](#)
  - [Execution audit](#)
  - [Map Settings](#)
  - [Data settings](#)
- 

## Data Log

The information in the Data Log section of the MapAudit can be configured by using the Data Audit Settings tab in the Organizer. See Map Designer in the online documentation for information about configuring these options and interpreting the information in the Data Log.

---

## Execution audit

When the Execution Log setting is set to ON, the MapAudit log contains an ExecutionLog entry for each burst within the map. The ExecutionSummary section provides a summary of the return codes, sources, targets, and work areas for the map.

- [ExecutionLog per burst](#)
  - [ExecutionSummary per map](#)
- 

## ExecutionLog per burst

When the Data Log or Execution Log setting is set to ON, the MapAudit log contains a section for each burst within the map. If all inputs have a CardMode of Integral, there is a single Burst section.

The ExecutionLog section identifies the return code and elapsed time for the burst, as well as the status of each input or output, including both an adapter return code and a content return code.

---

## ExecutionSummary per map

The ExecutionSummary provides information at the map level.

The execution log provides high-level debugging information, including the following:

### Map return code and message

The map return code and message indicate how the mapping operation completed and whether there were any problems. For example, a map return code of 0 and message of Map completed successfully indicate that no execution errors were encountered. This information helps in analyzing the source information in this log.

### SourceReport and TargetReport

For each source or target, the ExecutionSummary includes information indicating the adapter, the size of the data for the source or target, the adapter return code and message, and so on.

### WorkArea

For each input or output for which a WorkArea is created, the ExecutionSummary includes information such as the location and size.

The Execution log is a good place to start when diagnosing map execution problems because you can quickly determine the sources or targets that are in error. Then, using the information in the log, you can produce more detailed troubleshooting information for only those sources or targets that experienced problems.

---

## Map Settings

mapsThe map settings in the Map Designer contains a list of all MapSettings, including the settings for MapAudit, MapTrace, WorkSpace, Century, Validation, Retry, and Warnings.

This information can be useful during debugging to determine why execution occurred in a certain way.

---

## Data settings

The data settings for input and output cards in the map source file in the Map Designer contains a list of all InputData and OutputData settings, including the FetchAs, WorkArea, Backup, PUT > Target > Command, GET > Source > Command, OnSuccess, OnFailure, Retry, Scope, FetchUnit, and so on.

This information can be useful during debugging to identify whether data should be copied to a backup file, whether the changes to a target should be committed if the map fails, and so on.

---

## R/3 adapter audit files

Additional troubleshooting and diagnostic information is available in the R/3 adapter audit file. Specify the **-AR3** adapter command to create a file that records the adapter activity for each specified SAP R/3 object activity. This command can be used for a source or target, or in a GET or PUT function. This adapter command can be specified for individual input and output cards on a card-by-card basis.

The default is to produce a file named `m4r3adapter.log` in the directory in which the map is located, where `adapter` is the adapter type. Optionally, you can append the audit information to an existing file or specify a name or the full path for the file.

## R/3 adapter trace files

Using the information contained in R/3 adapter trace files (`m4r3adapter.mtr`) is one of the primary tools you can use to assist in troubleshooting. These files contain detailed information generated during map execution. The trace file produced at map execution time records detailed information about the R/3 adapter activity, such as objects retrieved, data source and target activity, and so on.

To produce trace information for specific R/3 data sources or targets, use the Trace (**-T**) adapter command.

For example, to produce an adapter trace, include the **-T** adapter command in the Source Command or Target > Command setting, or use it with the appropriate execution command on the command line.

With this adapter command, trace information is generated in the `.mtr` file.

- [R/3 adapter trace - Verbose option](#)

## R/3 adapter trace - Verbose option

When the Trace command is specified, there are several options you can use including the Verbose (**v**) option, which records detailed trace information.

## R/3 return codes and error messages

The following is a listing of all the codes and messages that can be returned as a result of using the R/3 adapter for sources or targets.

Note: Adapter return codes with positive numbers are warning codes that indicate a successful operation. Adapter return codes with negative numbers are error codes that indicate a failed operation.

Table 1. R/3 adapter return codes and error messages

| Return code | Message                                                                            |
|-------------|------------------------------------------------------------------------------------|
| 0           | OK                                                                                 |
| 1           | No data provided. Create on content specified: no connection attempted.            |
| 2           | Map execution failed, data not sent                                                |
| -1          | Error in data prep                                                                 |
| -1          | Error: cannot get function definition                                              |
| -1          | RFC Open failed                                                                    |
| -1          | Put Data failed                                                                    |
| -1          | Unknown_Error                                                                      |
| -1          | Error in setup                                                                     |
| -1          | TID sent twice so not processed                                                    |
| -1          | R/3 may try again later                                                            |
| -1          | Couldn't init adapter. New_semaphore() failed.                                     |
| -1          | Couldn't init adapter. Watch count exceeded.                                       |
| -1          | Couldn't init watchpoint. New_thread() failed.                                     |
| -1          | Error in data prep (invalid Idocs).                                                |
| -1          | Error in data prep (invalid BAPI data or Idoc missing EDI_DC)                      |
| -1          | Connection Test failed                                                             |
| -1          | Connection Test not possible, handle is invalid                                    |
| -1          | RFC Accept Failed                                                                  |
| -1          | Install Function Failed                                                            |
| -1          | InitRfcConn Failed                                                                 |
| -1          | XXXXX = FAILURE(0)<br>For APIs that return nonzero on success.                     |
| -?          | XXXXX = FAILURE(?)<br>For APIs that return nonzero on success.                     |
| -1          | XXXXX = RFC_FAILURE: Error occurred.                                               |
| -2          | XXXXX = RFC_EXCEPTION: Exception raised.                                           |
| -3          | XXXXX = RFC_SYS_EXCEPTION: System exception raised, connection closed.;            |
| -3          | User data properties not set. IDoc may be corrupted.                               |
| -4          | XXXXX = RFC_CALL: Call received.                                                   |
| -5          | XXXXX = RFC_INTERNAL_COM: Internal communication, repeat<br>For internal use only. |
| -6          | XXXXX = RFC_CLOSED: Connection closed by the other side.                           |
| -7          | XXXXX = RFC_RETRY: No data yet (RfcListen or RfcWaitForRequest only).              |
| -8          | XXXXX = RFC_NO_TID: No Transaction ID available.                                   |
| -9          | XXXXX = RFC_EXECUTED: Function already executed.                                   |

| Return code | Message                                                                    |
|-------------|----------------------------------------------------------------------------|
| -10         | XXXXX = RFC_SYNCHRONIZE: Synchronous Call in Progress (only for Windows).; |
| -10         | Invalid connection, RFCPING failed, see trace file for details.            |
| -11         | XXXXX = RFC_MEMORY_INSUFFICIENT: Memory insufficient.                      |
| -12         | XXXXX = RFC_VERSION_MISMATCH: Version mismatch.                            |
| -13         | XXXXX = RFC_NOT_FOUND: Function not found.<br>For internal use only.       |
| -14         | XXXXX = RFC_CALL_NOT_SUPPORTED: This call is not supported on WINDOWS.     |
| -15         | XXXXX = RFC_NOT_OWNER: Caller does not own the specified handle.           |
| -16         | XXXXX = RFC_NOT_INITIALIZED: RFC not yet initialized.                      |
| -?          | XXXXX = RFC_UNKNOWN: Unknown result code ?.                                |
| -?          | Is an API-specific error code, made negative if >0                         |
| XXXXX       | is a SAP API or TID Mgmt Function                                          |
| -464        | Initialization failure. (sanity check failed)                              |
| -600        | Internal Error: Resource Manager Error                                     |
| -1000       | Initialization failure. (program arguments)                                |
| -2000       | Initialization failure. (backup/log init)                                  |

## Viewing R/3 source and target data

When debugging a map that uses SAP R/3 sources or targets, you cannot view the source or target data in the Map Designer by selecting the Run Results from the View menu. However, you can capture the data that was retrieved from or written to an SAP R/3 object for debugging purposes by using Backup settings.

- [Backup settings](#)

## Backup settings

Backup settings are used to determine when, where, and how the data for a specific card should be copied to a specified backup file. These settings are configured in the Input and Output Card Settings in the Map Designer and the Launcher or Command Settings in the Integration Flow Manager.

Note: Refer to the Map Designer in the online documentation for more information on Backup settings.

## Remaining temporary data in TIDDATA directory

The adapter uses the tiddata directory internally to place intermediate files. This directory is placed in the DTX\_TMP\_DIR/tiddata for Linux® and UNIX platforms.

## Financial Services standards

The IBM Transformation Extender Pack for Financial Payments, uses the strength of the data definitions and transformation capabilities from various key financial services standards.

The following industry standards are supported:

- Financial Information eXchange (FIX) which provide both tag value encoding and FIXML support
- NACHA, which provides support for clients that need to validate, create, or transform ACH messages
- Single Euro Payments Area (SEPA) - based on ISO 20022
- Society for Worldwide Interbank Financial Telecommunication (SWIFT). The SWIFT standards include the following standards:
  - SWIFT MT - based on ISO 7775 and ISO 15022
  - SWIFT MX - based on ISO 20022
- Generic ISO 20022-based XML not covered under the SWIFT or SEPA standards. This supports a tool that can standardize messages across all payment types

## The IBM Transformation Extender product family

The IBM Transformation Extender product family provides a universal transformation engine that can describe and transform any data, work directly in any architecture, and reuse the transformations across the enterprise.

IBM Transformation Extender supports specific industry segments, such as financial services, with industry packs. Together, they provide the capabilities to perform the following:

- Transform, validate, and enrich any document, message or complex data
- Deliver trustworthy information for critical business initiatives
- Meet regulatory compliance requirements
- Support codeless development; universal reuse and deployment

## International Program License Agreement

The *International Program License Agreement* installs with the pack.

The *International Program License Agreement* is provided in English (United States) as well as in various other languages. The license agreement installs in the directory:

<install\_dir>/packs/financial\_payments\_vn.n.n.n/license

where <install\_dir> is the installation location of the Pack for Financial Payments, and n.n.n.n indicates the current version of the Pack for Financial Payments.

---

## Installation and uninstallation

Installation and uninstallation procedures are described in the release notes on the Support website.

Installation and installation procedures are described in the release notes.

<http://www.ibm.com/support/docview.wss?uid=swg27008337>

---

## Overview of the FIX component

The Financial Information eXchange (FIX) protocol is a message standard that was developed to facilitate the electronic exchange of information that is related to securities transactions. It is used between trading partners that want to automate communications.

Many business functions are supported by the message protocol. Initially, FIX was intended to support US domestic trading with message traffic being exchanged between principals. As the FIX protocol evolved, a number of fields were added to enable cross-border trading, derivatives, fixed income and other products. This functionality is expected to continue as subsequent releases of FIX become available.

The Pack for Financial Payments, FIX component consists of a set of IBM Transformation Extender type trees and maps that form the basis for a messaging library for larger solutions that require simple FIX message creation, parsing and validation.

It is important to note that FIX is not a software application. It is instead a specification, around which developers can create commercial, or open-source software as needed to address specific business needs. Today, as the leading trade communication protocol, FIX is integral to an array of other management and trading systems, users of which often benefit from the systems without knowing the language itself.

- [\*\*What the FIX component contains\*\*](#)

FIX type trees, FIXML type trees, and example conversion maps are installed when you install this pack.

---

## What the FIX component contains

FIX type trees, FIXML type trees, and example conversion maps are installed when you install this pack.

Specifically, the following objects are installed when you install the Pack for Financial Payments, FIX component:

- FIX type trees - These type trees support all FIX versions: 4.0, 4.1, 4.2, 4.3, 4.4, 5.0, 5.0 SP1, 5.0 SP2, and 5.0 SP2 Extension Packs (up to EP291). The FIX type trees provide consistency in format and content across the various services engagements that are related to FIX solutions.
- FIXML type trees - These type trees represent XML equivalents of FIX versions: 4.1, 4.2, 4.3, 4.4, 5.0, 5.0 SP1, and 5.0 SP2 (Latest). The type trees are based on DTDs and the schemas that are provided by FIX.
- FIXML schemas - These schemas are copies of the latest published FIXML schemas that support the following FIXML versions: 4.4, 5.0, 5.0 SP1, 5.0 SP2, and 5.0 SP2 Extension Packs (based from EP291).
- FIXT 1.1 type tree - This type tree supports the FIX Session Protocol messages.
- [\*\*Major components of the pack\*\*](#)

---

## Major components of the pack

When you install the Pack for Financial Payments, FIX component, the following subdirectories are created: utilities, examples, schemas, and trees. These directories are created in the following location:

<install\_dir>/packs/financial\_payments\_vn.n.n.n/fx

- [\*\*The utilities subdirectory\*\*](#)  
The utilities subdirectory contains the data, maps and trees subdirectories.
- [\*\*The trees subdirectory\*\*](#)  
The trees subdirectory contains type trees that support various FIX standards. It also contains FIXML type trees that are generated from DTDs, or schemas.
- [\*\*The schemas subdirectory\*\*](#)
- [\*\*FIX component examples\*\*](#)  
The Pack for Financial Payments, FIX component contains examples to assist you in using the pack.

---

## The utilities subdirectory

The utilities subdirectory contains the data, maps and trees subdirectories.

These subdirectories are described as follows:

- data subdirectory - The data subdirectory contains the following sample input data files and map results are also stored in this location:
  - fix\_40.inp - This file is used with the FIX checksum utility maps.
  - fix\_40\_MVS.inp - This version of fix\_40.inp is converted to EBCDIC characters, to be used as input when executing an example on z/OS.
- maps subdirectory - The maps subdirectory contains a map source called fix\_checksum.mms. It consists of the utility example maps described as follows:
  - fxchksm1\_utility - The fxchksm1\_utility map is used to compute the FIX checksum by summing every byte value equivalent of every character that is contained in a FIX message and applying a MODULUS 256. This map requires the FIX message to be formatted in hexadecimal text stream representation.
  - fxchksm2\_validate - The fxchksm2\_validate utility map is used to verify the computed FIX checksum value against the actual FIX message field tag 10 value. The computed FIX checksum is returned by invoking the fxchksm1\_utility map. The map returns a value of PASS if the value on field tag 10 matches the computed FIX checksum. If the value on field tag 10 does not match the computed FIX checksum, a value of FAIL is returned. The map requires the FIX message to have tag 10 to be populated as the last field in the message. The fxchksm1\_utility map must be compiled before execution of this map and the compiled map must be the same location where the fxchksm2\_validate map is being executed. This map can be used for inbound mapping where the field tag 10 value must be verified.
  - fxchksm3\_compute - The fxchksm3\_compute utility map retrieves the computed FIX checksum by calling fxchksm1\_utility map. The map requires a FIX message as input. Any field tag 10 is removed before it is passed to the fxchksm1\_utility map. The first output card contains the computed FIX checksum that results from calling the fxchksm1\_utility map. The second output card contains the computed FIX body length. The body length is the number of characters from body length tag 9 up to and including the delimiter immediately preceding the checksum tag 10. The second output card is optional and provided as a convenience. It is configured with a SINK adapter, which can be overridden when needed. The fxchksm1\_utility map must be compiled before execution of this map and the compiled map must be in the same location where fxchksm3\_compute map is being executed. This map can be used for outbound mapping where the field tag 10 value must be populated.
- trees subdirectory - The trees subdirectory contains fix\_checksum.mtt, a general-purpose type tree, which is used in both the input and output cards of the FIX checksum utility maps.

## The trees subdirectory

The trees subdirectory contains type trees that support various FIX standards. It also contains FIXML type trees that are generated from DTDs, or schemas.

The type trees that are installed in the trees subdirectory are described in the following table:

| Type tree           | FIX version | Comments                                                       |
|---------------------|-------------|----------------------------------------------------------------|
| fix_40.mtt          | 4.0         | Type tree supports FIX 4.0 version messages                    |
| fix_41.mtt          | 4.1         | Type tree supports FIX 4.1 version messages                    |
| fix_42.mtt          | 4.2         | Type tree supports FIX 4.2 version messages                    |
| fix_43.mtt          | 4.3         | Type tree supports FIX 4.3 version messages                    |
| fix_44.mtt          | 4.4         | Type tree supports FIX 4.4 version messages                    |
| fix_50.mtt          | 5.0         | Type tree supports FIX 5.0 version messages                    |
| fix_50_sp1.mtt      | 5.0 SP1     | Type tree supports FIX 5.0 SP1 version messages                |
| fix_50_sp2.mtt      | 5.0 SP2     | Type tree supports FIX 5.0 SP2 version messages                |
| fix_50_sp2_ep.mtt   | 5.0 SP2 EP  | Type tree supports FIX LATEST (up to EP 291) version messages. |
| fixt_11.mtt         | 1.1         | Type tree supports FIXT 1.1 version messages                   |
| fixml_41.mtt        | 4.1         | Type tree which is generated from a FIXML DTD                  |
| fixml_42.mtt        | 4.2         | Type tree which is generated from a FIXML DTD                  |
| fixml_43.mtt        | 4.3         | Type tree which is generated from a FIXML schema               |
| fixml_44.mtt        | 4.4         | Type tree which is generated from a FIXML schema               |
| fixml_50.mtt        | 5.0         | Type tree which is generated from a FIXML schema               |
| fixml_50_sp1.mtt    | 5.0 SP1     | Type tree which is generated from a FIXML schema               |
| fix-main-latest.mtt | FIX LATEST  | Type tree which is generated from a FIXML schema               |

## The schemas subdirectory

The schemas subdirectory contains FIXML schemas in the following subdirectories:

- fixml\_44 - This directory contains the FIXML 4.4 schema: Schema version 20040109 with Revision 1, 2006-10-06
- fixml\_50 - This directory contains the FIXML 5.0 schema: Schema version 20070103
- fixml\_50\_sp1 - This directory contains the FIXML 5.0 SP1 schema: Schema version 20080314
- fixml\_50\_sp2 - This directory contains several folders corresponding to publication dates of the following FIXML 5.0 SP2 schemas:
  - 20131209 – FIXML 5.0 SP2 errata version
  - 20240704 – FIX LATEST (up to EP291)

## FIX component examples

The Pack for Financial Payments, FIX component contains examples to assist you in using the pack.

The list of FIX component examples is as follows:

- Advertisement messages
- Allocation Instruction message
- Allocation Instruction Ack message
- Don't Know Trade (DK) message

- Email message
- Execution Report message
- Indication of interest message
- List Cancel Request message
- List Execute message
- List Status message
- List Status Request message
- New Order List message
- News message
- Order Cancel Reject message
- New Order message
- Order Status Request message
- Quote message
- Quote Request message

See [FIX component examples](#).

## FIX type trees and message structure

The FIX type trees are listed in [The trees subdirectory](#). The documentation provided here describes FIX message protocol and defines message types by FIX version number.

- [FIX message protocol](#)  
The FIX protocol is defined at two levels, session and application.
- [Message types per FIX version number](#)  
This documentation shows the message types that are supported in the Pack for Financial Payments, FIX component. Messages are listed by FIX version.
- [Type tree features](#)  
The type trees contained in the Pack for Financial Payments, FIX component serve to enforce FIX message structure, message syntax, as well as to enforce additional data validation rules.
- [Example message structure](#)  
The documentation provided here shows the FIX message structure by showing an example header, body and trailer.

## FIX message protocol

The FIX protocol is defined at two levels, session and application.

The session level consists of:

- Heartbeat
- Logon
- Logout
- Reject
- Resend Request
- Sequence Reset
- Test Request
- XMLnonFIX

The application level consists of:

- Pre-trade
- Trade
- Post-trade
- Infrastructure

## Message types per FIX version number

This documentation shows the message types that are supported in the Pack for Financial Payments, FIX component. Messages are listed by FIX version.

- [Session messages](#)  
The table shows a summary of the messages under the Session category. The messages are listed by FIX version number.
- [EventCommunication messages](#)  
The table shows a summary of the messages under the EventCommunication category. The messages are listed by FIX version number.
- [Indication messages](#)  
The table shows a summary of the messages under the Indication category. The messages are listed by FIX version number.
- [MarketData messages](#)  
The table shows a summary of the messages under the MarketData category. The messages are listed by FIX version number.
- [QuotationNegotiation messages](#)  
The table shows a summary of the messages under the QuotationNegotiation category. The messages are listed by FIX version number.
- [SecuritiesReferenceData messages](#)  
The table shows a summary of the messages under the SecuritiesReferenceData category. The messages are listed by FIX version number.
- [MarketStructureReferenceData message types](#)  
The table shows the MarketStructureReferenceData message types by FIX version number.

- [\*\*PartiesAction messages\*\*](#)  
The table shows a summary of the messages under the PartiesAction category. The messages are listed by FIX version number.
- [\*\*PartiesReferenceData messages\*\*](#)  
The table shows a summary of the messages under the PartiesReferenceData category. The messages are listed by FIX version.
- [\*\*SingleGeneralOrderHandling messages\*\*](#)  
The table shows a summary of the messages under the SingleGeneralOrderHandling category. The messages are listed by FIX version number.
- [\*\*ProgramTrading messages\*\*](#)  
The table shows a summary of the messages under the ProgramTrading category. The messages are listed by FIX version number.
- [\*\*OrderMassHandling messages\*\*](#)  
The table shows a summary of the messages under the OrderMassHandling category. The messages are listed by FIX version number.
- [\*\*CrossOrders messages\*\*](#)  
The table shows a summary of the messages under the CrossOrders category. The messages are listed by FIX version number.
- [\*\*MultilegOrders messages\*\*](#)  
The table shows a summary of the messages under the MultilegOrders category. The messages are listed by FIX version.
- [\*\*Allocation messages\*\*](#)  
The table shows a summary of the messages under the Allocation category. The messages are listed by FIX version number.
- [\*\*SettlementInstruction messages\*\*](#)  
The table shows a summary of the messages under the SettlementInstruction category. The messages are listed by FIX version number.
- [\*\*SettlementStatusManagement messages\*\*](#)  
The table shows a summary of the messages under the SettlementStatusManagement category. The messages are listed by FIX version number.
- [\*\*RegistrationInstructions messages\*\*](#)  
The table shows a summary of the messages under the RegistrationInstructions category. The messages are listed by FIX version number.
- [\*\*TradeCapture messages\*\*](#)  
The table shows a summary of the messages under the TradeCapture category. The messages are listed by FIX version number.
- [\*\*TradeManagement messages\*\*](#)  
The table shows a summary of the messages under the TradeManagement category. The messages are listed by FIX version number.
- [\*\*Confirmation message types\*\*](#)  
The table shows a summary of the messages under the Confirmation category. The messages are listed by FIX version number.
- [\*\*PayManagement messages\*\*](#)  
The table shows a summary of the messages under the PayManagement category. The messages are listed by FIX version number.
- [\*\*PositionMaintenance messages\*\*](#)  
The table shows a summary of the messages under the PositionMaintenance category. The messages are listed by FIX version number.
- [\*\*CollateralManagement messages\*\*](#)  
The table shows a summary of the messages under the CollateralManagement category. The messages are listed by FIX version number.
- [\*\*MarginRequirementManagement messages\*\*](#)  
The table shows a summary of the messages under the MarginRequirementManagement category. The messages are listed by FIX version number.
- [\*\*AccountReporting messages\*\*](#)  
The table shows a summary of the messages under the AccountReporting category. The messages are listed by FIX version number.
- [\*\*BusinessReject messages\*\*](#)  
The table shows a summary of the messages under the BusinessReject category. The messages are listed by FIX version number.
- [\*\*Network messages\*\*](#)  
The table shows a summary of the messages under the Network category. The messages are listed by FIX version number.
- [\*\*UserManagement messages\*\*](#)  
The table shows a summary of the messages under the UserManagement category. The messages are listed by FIX version number.
- [\*\*Application messages\*\*](#)  
The table shows a summary of the messages under the Application category. The messages are listed by FIX version number.

## Session messages

The table shows a summary of the messages under the Session category. The messages are listed by FIX version number.

| Session messages | FIX<br>4.0 | FIX<br>4.1 | FIX<br>4.2 | FIX<br>4.3 | FIX<br>4.4 | FIX<br>5.0 | FIX<br>5.0<br>SP1 | FIX<br>5.0<br>SP2 | FIX<br>LATEST<br>(EP291) |
|------------------|------------|------------|------------|------------|------------|------------|-------------------|-------------------|--------------------------|
| Heartbeat        | x          | x          | x          | x          | x          | x          | x                 | x                 | x                        |
| Logon            | x          | x          | x          | x          | x          | x          | x                 | x                 | x                        |
| TestRequest      | x          | x          | x          | x          | x          | x          | x                 | x                 | x                        |
| ResendRequest    | x          | x          | x          | x          | x          | x          | x                 | x                 | x                        |
| Reject           | x          | x          | x          | x          | x          | x          | x                 | x                 | x                        |
| SequenceReset    | x          | x          | x          | x          | x          | x          | x                 | x                 | x                        |
| Logout           | x          | x          | x          | x          | x          | x          | x                 | x                 | x                        |
| XMLnonFIX        |            |            |            | x          | x          | x          | x                 | x                 | x                        |

## EventCommunication messages

The table shows a summary of the messages under the EventCommunication category. The messages are listed by FIX version number.

| Event Communication messages | FIX<br>4.0 | FIX<br>4.1 | FIX<br>4.2 | FIX<br>4.3 | FIX<br>4.4 | FIX<br>5.0 | FIX<br>5.0<br>SP1 | FIX<br>5.0<br>SP2 | FIX<br>LATEST<br>(EP291) |
|------------------------------|------------|------------|------------|------------|------------|------------|-------------------|-------------------|--------------------------|
|                              |            |            |            |            |            |            |                   |                   |                          |

| Event Communication messages | FIX               | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
|                              | 4.0 | 4.1 | 4.2 | 4.3 | 4.4 | 5.0 | SP1 | SP2 | LATEST<br>(EP291) |
| Email                        | x   | x   | x   | x   | x   | x   | x   | x   | x                 |
| News                         | x   | x   | x   | x   | x   | x   | x   | x   | x                 |

## Indication messages

The table shows a summary of the messages under the Indication category. The messages are listed by FIX version number.

| Indication messages | FIX               | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
|                     | 4.0 | 4.1 | 4.2 | 4.3 | 4.4 | 5.0 | SP1 | SP2 | LATEST<br>(EP291) |
| Advertisement       | x   | x   | x   | x   | x   | x   | x   | x   | x                 |
| CrossRequest        |     |     |     |     |     |     |     |     | x                 |
| CrossRequestAck     |     |     |     |     |     |     |     |     | x                 |
| IOI                 | x   | x   | x   | x   | x   | x   | x   | x   | x                 |

## MarketData messages

The table shows a summary of the messages under the MarketData category. The messages are listed by FIX version number.

| MarketData messages           | FIX               | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
|                               | 4.0 | 4.1 | 4.2 | 4.3 | 4.4 | 5.0 | SP1 | SP2 | LATEST<br>(EP291) |
| MarketDataIncrementalRefresh  |     |     | x   | x   | x   | x   | x   | x   | x                 |
| MarketDataRequest             |     |     | x   | x   | x   | x   | x   | x   | x                 |
| MarketDataReport              |     |     |     |     |     |     |     |     | x                 |
| MarketDataRequestReject       |     |     | x   | x   | x   | x   | x   | x   | x                 |
| MarketDataSnapshotFullRefresh |     |     | x   | x   | x   | x   | x   | x   | x                 |
| MarketDataStatisticsRequest   |     |     |     |     |     |     |     |     | x                 |
| MarketDataStatisticsReport    |     |     |     |     |     |     |     |     | x                 |
| StreamAssignmentReport        |     |     |     |     |     |     | x   | x   |                   |
| StreamAssignmentReportAck     |     |     |     |     |     |     | x   | x   |                   |
| StreamAssignmentRequest       |     |     |     |     |     |     | x   | x   |                   |

## QuotationNegotiation messages

The table shows a summary of the messages under the QuotationNegotiation category. The messages are listed by FIX version number.

| QuotationNegotiation messages | FIX               | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
|                               | 4.0 | 4.1 | 4.2 | 4.3 | 4.4 | 5.0 | SP1 | SP2 | LATEST<br>(EP291) |
| MassQuote                     |     |     | x   | x   | x   | x   | x   | x   | x                 |
| MassQuoteAck                  |     |     | x   | x   | x   | x   | x   | x   | x                 |
| Quote                         | x   | x   | x   | x   | x   | x   | x   | x   | x                 |
| QuoteAck                      |     |     |     |     |     |     |     |     | x                 |
| QuoteCancel                   |     |     | x   | x   | x   | x   | x   | x   | x                 |
| QuoteRequest                  | x   | x   | x   | x   | x   | x   | x   | x   | x                 |
| QuoteRequestReject            |     |     |     | x   | x   | x   | x   | x   | x                 |
| QuoteResponse                 |     |     |     | x   | x   | x   | x   | x   | x                 |
| QuoteStatusReport             |     |     | x   | x   | x   | x   | x   | x   | x                 |
| QuoteStatusRequest            |     | x   | x   | x   | x   | x   | x   | x   | x                 |
| RFQRequest                    |     |     | x   | x   | x   | x   | x   | x   | x                 |

## SecuritiesReferenceData messages

The table shows a summary of the messages under the SecuritiesReferenceData category. The messages are listed by FIX version number.

| SecuritiesReferenceData messages   | FIX               | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
|                                    | 4.0 | 4.1 | 4.2 | 4.3 | 4.4 | 5.0 | SP1 | SP2 | LATEST<br>(EP291) |
| DerivativeSecurityList             |     |     |     | x   | x   | x   | x   | x   | x                 |
| DerivativeSecurityListRequest      |     |     |     | x   | x   | x   | x   | x   | x                 |
| DerivativeSecurityListUpdateReport |     |     |     |     |     | x   | x   | x   | x                 |
| SecurityDefinition                 |     |     | x   | x   | x   | x   | x   | x   | x                 |
| SecurityDefinitionRequest          |     | x   | x   | x   | x   | x   | x   | x   | x                 |
| SecurityDefinitionUpdateReport     |     |     |     |     |     | x   | x   | x   | x                 |
| SecurityList                       |     |     |     | x   | x   | x   | x   | x   | x                 |
| SecurityListRequest                |     |     |     | x   | x   | x   | x   | x   | x                 |
| SecurityListUpdateReport           |     |     |     |     |     | x   | x   | x   | x                 |
| SecurityMassStatus                 |     |     |     |     |     |     |     |     | x                 |
| SecurityMassStatusRequest          |     |     |     |     |     |     |     |     | x                 |
| SecurityRiskMetricsReport          |     |     |     |     |     |     |     |     | x                 |
| SecurityStatus                     |     | x   | x   | x   | x   | x   | x   | x   | x                 |
| SecurityStatusRequest              |     | x   | x   | x   | x   | x   | x   | x   | x                 |
| SecurityTypeRequest                |     |     | x   | x   | x   | x   | x   | x   | x                 |
| SecurityTypes                      |     |     |     | x   | x   | x   | x   | x   | x                 |

## MarketStructureReferenceData message types

The table shows the MarketStructureReferenceData message types by FIX version number.

| MarketStructureReferenceData message types | FIX               | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
|                                            | 4.0 | 4.1 | 4.2 | 4.3 | 4.4 | 5.0 | SP1 | SP2 | LATEST<br>(EP291) |
| MarketDefinition                           |     |     |     |     |     |     | x   | x   | x                 |
| MarketDefinitionRequest                    |     |     |     |     |     |     | x   | x   | x                 |
| MarketDefinitionUpdateReport               |     |     |     |     |     |     | x   | x   | x                 |
| TradingSessionsList                        |     |     |     |     |     |     | x   | x   | x                 |
| TradingSessionsListRequest                 |     |     |     |     |     |     | x   | x   | x                 |
| TradingSessionListUpdateReport             |     |     |     |     |     |     | x   | x   | x                 |
| TradingSessionStatus                       |     |     |     | x   | x   | x   | x   | x   | x                 |
| TradingSessionStatusRequest                |     |     | x   | x   | x   | x   | x   | x   | x                 |

## PartiesAction messages

The table shows a summary of the messages under the PartiesAction category. The messages are listed by FIX version number.

| PartiesAction messages        | FIX               | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
|                               | 4.0 | 4.1 | 4.2 | 4.3 | 4.4 | 5.0 | SP1 | SP2 | LATEST<br>(EP291) |
| PartyActionReport             |     |     |     |     |     |     |     | x   |                   |
| PartyActionRequest            |     |     |     |     |     |     |     | x   |                   |
| PartyRiskLimitCheckRequest    |     |     |     |     |     |     |     | x   |                   |
| PartyRiskLimitCheckRequestAck |     |     |     |     |     |     |     | x   |                   |

## PartiesReferenceData messages

The table shows a summary of the messages under the PartiesReferenceData category. The messages are listed by FIX version.

| PartiesReferenceData messages    | FIX               | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
|                                  | 4.0 | 4.1 | 4.2 | 4.3 | 4.4 | 5.0 | SP1 | SP2 | LATEST<br>(EP291) |
| PartyDetailsDefinitionRequest    |     |     |     |     |     |     |     | x   |                   |
| PartyDetailsDefinitionRequestAck |     |     |     |     |     |     |     | x   |                   |
| PartyDetailsListReport           |     |     |     |     |     |     |     | x   |                   |
| PartyDetailsListRequest          |     |     |     |     |     |     |     | x   |                   |
| PartyDetailsListUpdateReport     |     |     |     |     |     |     |     | x   |                   |

| PartiesReferenceData messages         | FIX |                   | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
|                                       | 4.0 | 4.1 | 4.2 | 4.3 | 4.4 | 5.0 | 5.0 | SP1 | SP2 | LATEST<br>(EP291) |
| PartyEntitlementsDefinitionRequest    |     |     |     |     |     |     |     |     | x   |                   |
| PartyEntitlementsDefinitionRequestAck |     |     |     |     |     |     |     |     | x   |                   |
| PartyEntitlementsReport               |     |     |     |     |     |     |     |     | x   |                   |
| PartyEntitlementsRequest              |     |     |     |     |     |     |     |     | x   |                   |
| PartyEntitlementsUpdateReport         |     |     |     |     |     |     |     |     | x   |                   |
| PartyRiskLimitsDefinitionRequest      |     |     |     |     |     |     |     |     | x   |                   |
| PartyRiskLimitsDefinitionRequestAck   |     |     |     |     |     |     |     |     | x   |                   |
| PartyRiskLimitsReport                 |     |     |     |     |     |     |     |     | x   |                   |
| PartyRiskLimitsRequest                |     |     |     |     |     |     |     |     | x   |                   |
| PartyRiskLimitsUpdateReport           |     |     |     |     |     |     |     |     | x   |                   |
| PartyRiskLimitsReportAck              |     |     |     |     |     |     |     |     | x   |                   |

## SingleGeneralOrderHandling messages

The table shows a summary of the messages under the SingleGeneralOrderHandling category. The messages are listed by FIX version number.

| SingleGeneralOrderHandling messages | FIX |                   | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
|                                     | 4.0 | 4.1 | 4.2 | 4.3 | 4.4 | 5.0 | 5.0 | SP1 | SP2 | LATEST<br>(EP291) |
| DontKnowTrade                       | x   | x   | x   | x   | x   | x   | x   | x   | x   |                   |
| ExecutionAck                        |     |     |     |     |     | x   | x   | x   | x   |                   |
| ExecutionReport                     | x   | x   | x   | x   | x   | x   | x   | x   | x   |                   |
| NewOrderSingle                      | x   | x   | x   | x   | x   | x   | x   | x   | x   |                   |
| OrderCancelReject                   | x   | x   | x   | x   | x   | x   | x   | x   | x   |                   |
| OrderCancelReplaceReject            | x   | x   | x   | x   | x   | x   | x   | x   | x   |                   |
| OrderCancelRequest                  | x   | x   | x   | x   | x   | x   | x   | x   | x   |                   |
| OrderStatusRequest                  | x   | x   | x   | x   | x   | x   | x   | x   | x   |                   |

## ProgramTrading messages

The table shows a summary of the messages under the ProgramTrading category. The messages are listed by FIX version number.

| ProgramTrading messages | FIX |                   | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
|                         | 4.0 | 4.1 | 4.2 | 4.3 | 4.4 | 5.0 | 5.0 | SP1 | SP2 | LATEST<br>(EP291) |
| BidRequest              |     |     | x   | x   | x   | x   | x   | x   | x   |                   |
| BidResponse             |     |     | x   | x   | x   | x   | x   | x   | x   |                   |
| ListCancelRequest       | x   | x   | x   | x   | x   | x   | x   | x   | x   |                   |
| ListExecute             | x   | x   | x   | x   | x   | x   | x   | x   | x   |                   |
| ListStatus              | x   | x   | x   | x   | x   | x   | x   | x   | x   |                   |
| ListStatusRequest       | x   | x   | x   | x   | x   | x   | x   | x   | x   |                   |
| ListStrikePrice         |     |     | x   | x   | x   | x   | x   | x   | x   |                   |
| NewOrderList            | x   | x   | x   | x   | x   | x   | x   | x   | x   |                   |

## OrderMassHandling messages

The table shows a summary of the messages under the OrderMassHandling category. The messages are listed by FIX version number.

| OrderMassHandling messages | FIX |                   | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
|                            | 4.0 | 4.1 | 4.2 | 4.3 | 4.4 | 5.0 | 5.0 | SP1 | SP2 | LATEST<br>(EP291) |
| OrderMassActionReport      |     |     |     |     |     | x   | x   | x   |     |                   |
| OrderMassActionRequest     |     |     |     |     |     | x   | x   | x   |     |                   |
| OrderMassCancelReport      |     |     |     | x   | x   | x   | x   | x   | x   |                   |
| OrderMassCancelRequest     |     |     |     | x   | x   | x   | x   | x   | x   |                   |
| OrderMassStatusRequest     |     |     |     | x   | x   | x   | x   | x   | x   |                   |
| MassOrder                  |     |     |     |     |     |     |     |     | x   |                   |

| OrderMassHandling messages | FIX     | FIX    | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
|                            | 4.0 | 4.1 | 4.2 | 4.3 | 4.4 | 5.0 | 5.0 | 5.0     | LATEST |
| MassOrderAck               |     |     |     |     |     | SP1 | SP2 | (EP291) | x      |

## CrossOrders messages

The table shows a summary of the messages under the CrossOrders category. The messages are listed by FIX version number.

| CrossOrders messages           | FIX    | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
|                                | 4.0 | 4.1 | 4.2 | 4.3 | 4.4 | 5.0 | 5.0 | 5.0 | LATEST |
| CrossOrderCancelReplaceRequest |     |     |     | x   | x   | x   | x   | x   | x      |
| CrossOrderCancelRequest        |     |     |     | x   | x   | x   | x   | x   | x      |
| NewOrderCross                  |     |     |     | x   | x   | x   | x   | x   | x      |

## MultilegOrders messages

The table shows a summary of the messages under the MultilegOrders category. The messages are listed by FIX version.

| MultilegOrders messages    | FIX    | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
|                            | 4.0 | 4.1 | 4.2 | 4.3 | 4.4 | 5.0 | 5.0 | 5.0 | LATEST |
| MultilegOrderCancelReplace |     |     | x   | x   | x   | x   | x   | x   | x      |
| NewOrderMultileg           |     |     | x   | x   | x   | x   | x   | x   | x      |

## Allocation messages

The table shows a summary of the messages under the Allocation category. The messages are listed by FIX version number.

| Allocation messages                  | FIX    | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
|                                      | 4.0 | 4.1 | 4.2 | 4.3 | 4.4 | 5.0 | 5.0 | 5.0 | LATEST |
| AllocationInstruction                | x   | x   | x   | x   | x   | x   | x   | x   | x      |
| AllocationInstructionAck             | x   | x   | x   | x   | x   | x   | x   | x   | x      |
| AllocationInstructionAlert           |     |     |     |     |     | x   | x   | x   | x      |
| AllocationInstructionAlertRequest    |     |     |     |     |     |     |     |     | x      |
| AllocationInstructionAlertRequestAck |     |     |     |     |     |     |     |     | x      |
| AllocationReport                     |     |     |     |     | x   | x   | x   | x   | x      |
| AllocationReportAck                  |     |     |     |     | x   | x   | x   | x   | x      |

## SettlementInstruction messages

The table shows a summary of the messages under the SettlementInstruction category. The messages are listed by FIX version number.

| SettlementInstruction messages | FIX    | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
|                                | 4.0 | 4.1 | 4.2 | 4.3 | 4.4 | 5.0 | 5.0 | 5.0 | LATEST |
| SettlementInstructionRequest   |     |     |     | x   | x   | x   | x   | x   | x      |
| SettlementInstructions         | x   | x   | x   | x   | x   | x   | x   | x   | x      |
| SettlementObligationReport     |     |     |     |     |     | x   | x   | x   | x      |

## SettlementStatusManagement messages

The table shows a summary of the messages under the SettlementStatusManagement category. The messages are listed by FIX version number.

| SettlementStatusManagement messages | FIX<br>4.0 | FIX<br>4.1 | FIX<br>4.2 | FIX<br>4.3 | FIX<br>4.4 | FIX<br>5.0 | FIX<br>5.0 | FIX<br>LATEST<br>(EP291) |
|-------------------------------------|------------|------------|------------|------------|------------|------------|------------|--------------------------|
|                                     | SP1        | SP2        |            |            |            |            |            |                          |
| SettlementStatusReport              |            |            |            |            |            |            | x          |                          |
| SettlementStatusReportAck           |            |            |            |            |            |            | x          |                          |
| SettlementStatusRequest             |            |            |            |            |            |            | x          |                          |
| SettlementStatusRequestAck          |            |            |            |            |            |            | x          |                          |

## RegistrationInstructions messages

The table shows a summary of the messages under the RegistrationInstructions category. The messages are listed by FIX version number.

| RegistrationInstructions messages | FIX<br>4.0 | FIX<br>4.1 | FIX<br>4.2 | FIX<br>4.3 | FIX<br>4.4 | FIX<br>5.0 | FIX<br>5.0 | FIX<br>LATEST<br>(EP291) |
|-----------------------------------|------------|------------|------------|------------|------------|------------|------------|--------------------------|
|                                   | SP1        | SP2        |            |            |            |            |            |                          |
| RegistrationInstructions          |            |            |            | x          | x          | x          | x          | x                        |
| RegistrationInstructionsResponse  |            |            |            | x          | x          | x          | x          | x                        |

## TradeCapture messages

The table shows a summary of the messages under the TradeCapture category. The messages are listed by FIX version number.

| TradeCapture messages        | FIX<br>4.0 | FIX<br>4.1 | FIX<br>4.2 | FIX<br>4.3 | FIX<br>4.4 | FIX<br>5.0 | FIX<br>5.0 | FIX<br>LATEST<br>(EP291) |
|------------------------------|------------|------------|------------|------------|------------|------------|------------|--------------------------|
|                              | SP1        | SP2        |            |            |            |            |            |                          |
| TradeCaptureReport           |            |            |            | x          | x          | x          | x          | x                        |
| TradeCaptureReportAck        |            |            |            |            | x          | x          | x          | x                        |
| TradeCaptureReportRequest    |            |            |            | x          | x          | x          | x          | x                        |
| TradeCaptureReportRequestAck |            |            |            |            | x          | x          | x          | x                        |
| TradeMatchReport             |            |            |            |            |            |            |            | x                        |
| TradeMatchReportAck          |            |            |            |            |            |            |            | x                        |

## TradeManagement messages

The table shows a summary of the messages under the TradeManagement category. The messages are listed by FIX version number.

| TradeManagement messages | FIX<br>4.0 | FIX<br>4.1 | FIX<br>4.2 | FIX<br>4.3 | FIX<br>4.4 | FIX<br>5.0 | FIX<br>5.0 | FIX<br>LATEST<br>(EP291) |
|--------------------------|------------|------------|------------|------------|------------|------------|------------|--------------------------|
|                          | SP1        | SP2        |            |            |            |            |            |                          |
| TradeAggregationReport   |            |            |            |            |            |            | x          |                          |
| TradeAggregationRequest  |            |            |            |            |            |            | x          |                          |

## Confirmation message types

The table shows a summary of the messages under the Confirmation category. The messages are listed by FIX version number.

| Confirmation message | FIX<br>4.0 | FIX<br>4.1 | FIX<br>4.2 | FIX<br>4.3 | FIX<br>4.4 | FIX<br>5.0 | FIX<br>5.0 | FIX<br>LATEST<br>(EP291) |
|----------------------|------------|------------|------------|------------|------------|------------|------------|--------------------------|
|                      | SP1        | SP2        |            |            |            |            |            |                          |
| Confirmation         |            |            |            | x          | x          | x          | x          | x                        |
| ConfirmationAck      |            |            |            | x          | x          | x          | x          | x                        |
| ConfirmationRequest  |            |            |            | x          | x          | x          | x          | x                        |

## PayManagement messages

The table shows a summary of the messages under the PayManagement category. The messages are listed by FIX version number.

| PayManagement messages  | FIX<br>4.0 | FIX<br>4.1 | FIX<br>4.2 | FIX<br>4.3 | FIX<br>4.4 | FIX<br>5.0 | FIX<br>5.0<br>SP1 | FIX<br>5.0<br>SP2 | FIX<br>LATEST<br>(EP291) |
|-------------------------|------------|------------|------------|------------|------------|------------|-------------------|-------------------|--------------------------|
| PayManagementReport     |            |            |            |            |            |            |                   | x                 |                          |
| PayManagementReportAck  |            |            |            |            |            |            |                   | x                 |                          |
| PayManagementRequest    |            |            |            |            |            |            |                   | x                 |                          |
| PayManagementRequestAck |            |            |            |            |            |            |                   | x                 |                          |

## PositionMaintenance messages

The table shows a summary of the messages under the PositionMaintenance category. The messages are listed by FIX version number.

| PositionMaintenance messages   | FIX<br>4.0 | FIX<br>4.1 | FIX<br>4.2 | FIX<br>4.3 | FIX<br>4.4 | FIX<br>5.0 | FIX<br>5.0<br>SP1 | FIX<br>5.0<br>SP2 | FIX<br>LATEST<br>(EP291) |
|--------------------------------|------------|------------|------------|------------|------------|------------|-------------------|-------------------|--------------------------|
| AdjustedPositionReport         |            |            |            |            |            | x          | x                 | x                 | x                        |
| AssignmentReport               |            |            |            |            | x          | x          | x                 | x                 | x                        |
| ContraryIntentionReport        |            |            |            |            |            | x          | x                 | x                 | x                        |
| PositionMaintenanceReport      |            |            |            |            | x          | x          | x                 | x                 | x                        |
| PositionMaintenanceRequest     |            |            |            |            | x          | x          | x                 | x                 | x                        |
| PositionReport                 |            |            |            |            | x          | x          | x                 | x                 | x                        |
| PositionTransferInstruction    |            |            |            |            |            |            |                   |                   | x                        |
| PositionTransferInstructionAck |            |            |            |            |            |            |                   |                   | x                        |
| PositionTransferReport         |            |            |            |            |            |            |                   |                   | x                        |
| RequestForPositions            |            |            |            |            | x          | x          | x                 | x                 | x                        |
| RequestForPositionsAck         |            |            |            |            | x          | x          | x                 | x                 | x                        |

## CollateralManagement messages

The table shows a summary of the messages under the CollateralManagement category. The messages are listed by FIX version number.

| CollateralManagement messages | FIX<br>4.0 | FIX<br>4.1 | FIX<br>4.2 | FIX<br>4.3 | FIX<br>4.4 | FIX<br>5.0 | FIX<br>5.0<br>SP1 | FIX<br>5.0<br>SP2 | FIX<br>LATEST<br>(EP291) |
|-------------------------------|------------|------------|------------|------------|------------|------------|-------------------|-------------------|--------------------------|
| CollateralAssignment          |            |            |            |            | x          | x          | x                 | x                 | x                        |
| CollateralInquiry             |            |            |            |            | x          | x          | x                 | x                 | x                        |
| CollateralInquiryAck          |            |            |            |            | x          | x          | x                 | x                 | x                        |
| CollateralReport              |            |            |            |            | x          | x          | x                 | x                 | x                        |
| CollateralReportAck           |            |            |            |            |            |            |                   |                   | x                        |
| CollateralRequest             |            |            |            |            | x          | x          | x                 | x                 | x                        |
| CollateralResponse            |            |            |            |            | x          | x          | x                 | x                 | x                        |

## MarginRequirementManagement messages

The table shows a summary of the messages under the MarginRequirementManagement category. The messages are listed by FIX version number.

| MarginRequirementManagement messages | FIX<br>4.0 | FIX<br>4.1 | FIX<br>4.2 | FIX<br>4.3 | FIX<br>4.4 | FIX<br>5.0 | FIX<br>5.0<br>SP1 | FIX<br>5.0<br>SP2 | FIX<br>LATEST<br>(EP291) |
|--------------------------------------|------------|------------|------------|------------|------------|------------|-------------------|-------------------|--------------------------|
| MarginRequirementInquiry             |            |            |            |            |            |            |                   | x                 |                          |
| MarginRequirementInquiryAck          |            |            |            |            |            |            |                   | x                 |                          |
| MarginRequirementReport              |            |            |            |            |            |            |                   | x                 |                          |

## AccountReporting messages

The table shows a summary of the messages under the AccountReporting category. The messages are listed by FIX version number.

| AccountReporting messages | FIX<br>4.0 | FIX<br>4.1 | FIX<br>4.2 | FIX<br>4.3 | FIX<br>4.4 | FIX<br>5.0 | FIX<br>5.0<br>SP1 | FIX<br>5.0<br>SP2 | FIX<br>LATEST<br>(EP291) |
|---------------------------|------------|------------|------------|------------|------------|------------|-------------------|-------------------|--------------------------|
| AccountSummaryReport      |            |            |            |            |            |            |                   | x                 |                          |

## BusinessReject messages

The table shows a summary of the messages under the BusinessReject category. The messages are listed by FIX version number.

| BusinessReject messages | FIX<br>4.0 | FIX<br>4.1 | FIX<br>4.2 | FIX<br>4.3 | FIX<br>4.4 | FIX<br>5.0 | FIX<br>5.0<br>SP1 | FIX<br>5.0<br>SP2 | FIX<br>LATEST<br>(EP291) |
|-------------------------|------------|------------|------------|------------|------------|------------|-------------------|-------------------|--------------------------|
| BusinessMessageReject   |            | x          | x          | x          | x          | x          | x                 | x                 | x                        |

## Network messages

The table shows a summary of the messages under the Network category. The messages are listed by FIX version number.

| Network messages                        | FIX<br>4.0 | FIX<br>4.1 | FIX<br>4.2 | FIX<br>4.3 | FIX<br>4.4 | FIX<br>5.0 | FIX<br>5.0<br>SP1 | FIX<br>5.0<br>SP2 | FIX<br>LATEST<br>(EP291) |
|-----------------------------------------|------------|------------|------------|------------|------------|------------|-------------------|-------------------|--------------------------|
| NetworkCounterpartySystemStatusRequest  |            |            |            |            | x          | x          | x                 | x                 | x                        |
| NetworkCounterpartySystemStatusResponse |            |            |            |            | x          | x          | x                 | x                 | x                        |

## UserManagement messages

The table shows a summary of the messages under the UserManagement category. The messages are listed by FIX version number.

| UserManagement messages | FIX<br>4.0 | FIX<br>4.1 | FIX<br>4.2 | FIX<br>4.3 | FIX<br>4.4 | FIX<br>5.0 | FIX<br>5.0<br>SP1 | FIX<br>5.0<br>SP2 | FIX<br>LATEST<br>(EP291) |
|-------------------------|------------|------------|------------|------------|------------|------------|-------------------|-------------------|--------------------------|
| UserNotification        |            |            |            |            |            | x          | x                 | x                 |                          |
| UserRequest             |            |            |            |            | x          | x          | x                 | x                 | x                        |
| UserResponse            |            |            |            |            | x          | x          | x                 | x                 | x                        |

## Application messages

The table shows a summary of the messages under the Application category. The messages are listed by FIX version number.

| Application messages         | FIX<br>4.0 | FIX<br>4.1 | FIX<br>4.2 | FIX<br>4.3 | FIX<br>4.4 | FIX<br>5.0 | FIX<br>5.0<br>SP1 | FIX<br>5.0<br>SP2 | FIX<br>LATEST<br>(EP291) |
|------------------------------|------------|------------|------------|------------|------------|------------|-------------------|-------------------|--------------------------|
| ApplicationMessageReport     |            |            |            |            |            | x          | x                 | x                 |                          |
| ApplicationMessageRequest    |            |            |            |            |            | x          | x                 | x                 |                          |
| ApplicationMessageRequestAck |            |            |            |            |            | x          | x                 | x                 |                          |

## Type tree features

The type trees contained in the Pack for Financial Payments, FIX component serve to enforce FIX message structure, message syntax, as well as to enforce additional data validation rules.

- [FIX message structure](#)  
A FIX message consists of a header, a message body, and a trailer:
- [Message syntax](#)  
The type tree enforces strict message syntax that is based upon three variable values, <Tag>, <Value>, and <Delimiter>.
- [Additional validation](#)  
In addition to enforcing message structure and syntax, the type trees also enforce a number of other rules.

## FIX message structure

A FIX message consists of a header, a message body, and a trailer:

- Header - The FIX message header identifies the message type, length, sender/destination, sequence number, sending time etc.
- Body - The FIX message body contains specific session and application message content.
- Trailer - The FIX message trailer contains an optional digital signature and the required checksum value.

It should be noted that all fields within the message can be defined in any sequence with the following exceptions:

- tag number 8, 9, and 35 of the header block must be the first three fields in the message
- tag number 10 of the trailer block must be the last field in the message
- fields within a repeating block must appear in the same order as defined in the FIX standard documents
- the number of repeating group instances must immediately precede the repeating group contents
- binary fields must be preceded by corresponding length fields

## Message syntax

The type tree enforces strict message syntax that is based upon three variable values, <Tag>, <Value>, and <Delimiter>.

The field format is: <Tag>=<Value><Delimiter>. Values for these entities are described as follows:

- <Tag> - any integer with no leading zeros
- = (equal sign) - the separator between Tag and Value.
- <Value> - value that represents the actual content of the field
- <Delimiter> - a non-printing ASCII character that is called the SOH character

## Additional validation

In addition to enforcing message structure and syntax, the type trees also enforce a number of other rules.

Additional validation rules are defined in the following table.

| Rule                                 | Description                                                                                                              |
|--------------------------------------|--------------------------------------------------------------------------------------------------------------------------|
| Restriction list verification        | Verifies the code list on FIX tags: examples are country codes, currency codes, exchange, and language data types.       |
| FIX tags numeric data types          | Enforces allowable numeric ranges for the following data types: int, float, Qty, Price, PriceOffset, Amt and Percentage. |
| Body length verification             | Validates that the actual body length is consistent with the value specified in tag number 9 of the FIX message.         |
| Message specific cross tag checking  | Validates the presence or absence of certain tags that depend on certain values of other tags within the same message.   |
| FIX tags without values verification | Verifies the FIX tags that have actual values specified. FIX tags without values fail the FIX type tree validation.      |

Checksum digit validation is no longer being enforced within the FIX type trees. See [Checksum integrity check](#).

The structure of the FIX tags allow restriction lists and numeric ranges to be implemented consistently across message types. These FIX fields are defined as "Group" class in the FIX type trees consisting of a common data type "Item" class. The data type items introduced in the FIX type trees are re-used in other FIX fields having the same data types. Maps built using FIX type trees from previous versions of the WebSphere Transformation Extender Pack for FIX require re-mapping of the FIX fields, specifically those fields that belong to the following data types:

- Country
- Currency
- Exchange
- Float (this includes data types, that is: Qty, Price, PriceOffset, Amt, and Percentage)

## Example message structure

The documentation provided here shows the FIX message structure by showing an example header, body and trailer.

These examples are taken from the example file that installs with the Pack for Financial Payments, FIX component.

- [\*\*Header example\*\*](#)  
The message header fields are shown below. These fields are taken from the example file that installs with the Pack for Financial Payments, FIX component.
- [\*\*Body example\*\*](#)  
The message header fields are shown below. These fields are taken from the example file that installs with the Pack for Financial Payments, FIX component.
- [\*\*Message trailer\*\*](#)  
The following conditions are not validated in the FIX type trees. You can extend the validation in your implementation through more mapping or lookups.
- [\*\*Additional validation recommendations\*\*](#)

## Header example

The message header fields are shown below. These fields are taken from the example file that installs with the Pack for Financial Payments, FIX component.

```

8=FIXT.1.1
9=7385
35=C
1128=7
1129=7.0.2
49=ABC Company
56=XYZ Inc.
115=ACME Bros.
128=XYZ Inc.
90=658
91=-----BEGIN PGP MESSAGE-----
Version: 4.0 Business Edition

hIwCe1B6T5v0aJEB/A/4q4sAo1j2edIg+3EVo7aJLaa1x7ROP8Qo+bFp6Sz3Bbmy+
3E5FqiNdxPmef7gw3ub8kAyCabTNhVVgkILcLtzU1NHd0t1XdmDkCF491gxKYA
BHseF18YHsivEJ0tI61fgfQ/uNoy41Tu1CUHBAic6ordEVMiLwTV7qAxnfYeJP
AAEFIZC6jUExoRE61x2insn75/hULOlzK1ZhNqfr4zVfjT25qddVHOoHMxZKo602
Vn+y+aDg0GjdMFSioIbdd0ZvsJDBr9ff67oXVfdlgl0CMrtPRp6CNwgnw/uRsg
1gUTmtB5XH3V0AQHTI7/qyqYTQ1mGlo6Gi8Z6M8XTwvHXeWK2Yb1Dus6eTgZfPg
p1Cjg/a/ejadICGg7RyJznEkrrpHMDE8XXiypw01dad6jvC8JvqSK8w7ZOI3mgjd
UbAuZ5SWIpBpW3N7sn+WdpB7R108dPdd4UPu4TXWZYbE5vKFwhC98r68GgxZ
184elHJCP8A960GiegKS5UD2TA1z7yIR
=bmt2
-----END PGP MESSAGE-----

34=12345
50=Trader
142=New York Stock Exchange
57=C/O Green Team
143=Chicago Board of Exchange
116=Blue Traders
144=London Stock Exchange
129=XYZ Inc.
145=Tokyo Stock Exchange
43=N
97=N
52=20071109-16:22:40.123
122=20071005-06:12:00.000
212=533
213=<ati:FIXML xmlns="http://www.fixprotocol.org/FIXML-5-0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http://www.fixprotocol.org/FIXML-5-0 ../../schema/fxml-main-4-4.xsd" v="4.4" r="20030618" s="20040109"
 xmlns:ati="http://www.fixprotocol.org/FIXML-5-0">
 <ati:Order ID="123456" Side="2" TxnTm="2001-09-11T09:30:47-05:00" Typ="2"
Px="93.25" Acct="26522154">
 <ati:Hdr Snt="2001-09-11T09:30:47-05:00" PosDup="N" PosRsnd="N" SeqNum="521"
SID="AFUNDMGR" TID="ABROKER"/>
 <ati:Instrmt Sym="IBM" ID="459200101" Src="1"/>
 <ati:OrdQty Qty="1000"/>
 </ati:Order>
</ati:FIXML>

347=UTF-8
369=12344
627=2
628=#1 ABC Company
629=20071109-16:22:40.123
630=12345
628=#2 ABC Company
629=20071109-16:22:40.123
630=12345

```

- Header fields**

The header fields in the example FIX message are described in the following table. In this table, the left hand column shows the actual field number, followed by the example data that is used in the preceding table.

## Header fields

The header fields in the example FIX message are described in the following table. In this table, the left hand column shows the actual field number, followed by the example data that is used in the preceding table.

| Tag Number | Value       | Required | Type of data     |
|------------|-------------|----------|------------------|
| 8          | FIXT.1.1    | Yes      | BeginString      |
| 9          | 7384        | Yes      | BodyLength       |
| 35         | C           | Yes      | MsgType          |
| 1128       | 7           | No       | AppVerID         |
| 1129       | 7.0.1       | No       | CstmAppVerID     |
| 49         | ABC Company | Yes      | SenderCompID     |
| 56         | XYZ Inc.    | Yes      | TargetCompID     |
| 115        | ACME Bros.  | No       | OnBehalfOfCompID |
| 128        | XYZ Inc     | No       | DeliverToCompID  |
| 90         | 658         | No       | SecureDataLen    |
| 91         | PGP Message | No       | SecureData       |
| 34         | 12345       | Yes      | MsgSeqNum        |
| 50         | Trader      | No       | MsgSeqNum        |

| Tag Number | Value                     | Required | Type of data           |
|------------|---------------------------|----------|------------------------|
| 142        | New York Stock Exchange   | No       | SenderLocationID       |
| 57         | C/O Green Team            | No       | TargetSubID            |
| 143        | Chicago Board of Exchange | No       | TargetLocationID       |
| 116        | Blue Traders              | No       | OnBehalfOfSubID        |
| 144        | London Stock Exchange     | No       | OnBehalfOfLocationID   |
| 129        | XYZ Inc.                  | No       | DeliverToSubID         |
| 145        | Tokyo Stock Exchange      | No       | DeliverToLocationID    |
| 43         | N                         | No       | PossDupFlag            |
| 97         | N                         | No       | PossResend             |
| 52         | 20071005-06:12:00.000     | No       | OrigSendingTime        |
| 212        | 533                       | No       | XmlDataLen             |
| 213        | <XML Data>                | No       | XMLData                |
| 347        | UTF-8                     | No       | MessageEncoding        |
| 369        | 12344                     | No       | LastMsgSeqNumProcessed |
| 627        | 2                         | No       | NoHops                 |
| > 628      | #1 ABC Company            | No       | HopCompID              |
| > 629      | 20071109-16:22:40.123     | No       | HopSendingTime         |
| > 630      | 12345                     | No       | HopRefID               |

## Body example

The message header fields are shown below. These fields are taken from the example file that installs with the Pack for Financial Payments, FIX component.

```

164=email #001
94=0
42=20071113-09:40:08
147=This is an email test
356=57
357="ä,ää-ç-!å·åç-ä•ä,ääÿäf"ä,ääf^ä®äftä,ääf^ää$ä,ä,<
215=2
216=3
217=block firm #007
216=1
217=target firm id #222
146=2
55=MCTR
65=WI
48=ISIN 01234567890
22=4
454=2
455=ASCL
456=A
455=MCTR
456=2
460=5
461=ESXXXX
167=CS
762=NONE
200=200303w2
541=20071113
1079=07:39:00
966=Settle on Open
1049=R
965=2
224=20010101
225=19991231
239=FUT
226=90
227=.9525
228=18.5
255=5 STARS
543=PTSAGBB0XXX

```

- [Body fields](#)

Some of the fields in the FIX message body are shown in the table that is provided here. These fields were taken from the example file that installs with the Pack for Financial Payments, FIX component.

## Body fields

Some of the fields in the FIX message body are shown in the table that is provided here. These fields were taken from the example file that installs with the Pack for Financial Payments, FIX component.

Note: The table and field descriptions do not include all of the FIX message body fields. The example and descriptions are provided for illustrative purposes only.

| Tag Number | Value      | Required | Header field |
|------------|------------|----------|--------------|
| 164        | email #001 | Yes      | EmailThread  |
| 94         | 0          | Yes      | EmailType    |

| Tag Number | Value             | Required | Header field               |
|------------|-------------------|----------|----------------------------|
| 42         | 20071113-09:40:08 | No       | OrigTime                   |
| 147        | An email test     | Yes      | Subject                    |
| 356        | 57                | No       | EncodedSubjectLen          |
| 357        | Encoded subject   | No       | EncodedSubject             |
| 215        | 2                 | No       | NoRoutingIDs               |
| > 216      | 3                 | No       | RoutingType                |
| > 217      | block firm #222   | No       | RoutingID                  |
| 146        | 2                 | No       | NoRelatedSym               |
| 55         | MCTR              | No       | Symbol                     |
| 65         | WI                | No       | SymbolFfx                  |
| 48         | ISIN 0123456789   | No       | SecurityID                 |
| 22         | 4                 | No       | SecurityIDSource           |
| 454        | 2                 | No       | NoSecurityAltID            |
| > 455      | ASCL              | No       | SecurityAltID              |
| >456       | A                 | No       | SecurityAltIDSource        |
| 460        | 5                 | No       | Product                    |
| 461        | ESXXX             | No       | CFCCode                    |
| 167        | CS                | No       | SecurityCode               |
| 762        | NONE              | No       | SecuritySubType            |
| 200        | 200303w2          | No       | MaturityMonthYear          |
| 541        | 20071113          | No       | MaturityDate               |
| 1079       | 07:39:00          | No       | MaturityTime               |
| 966        | Settle on Open    | No       | SettleOnOpenFlag           |
| 1049       | R                 | No       | InstrmtAssignmentMethod    |
| 965        | 2                 | No       | SecurityStatus             |
| 224        | 20010101          | No       | CouponPaymentDate          |
| 225        | 19991231          | No       | IssueDate                  |
| 239        | FUT               | No       | RepoCollateralSecurityType |
| 226        | 990               | No       | RepurchaseTerm             |
| 227        | .9525             | No       | RepurchaseRate             |
| 228        | 18.5              | No       | Factor                     |
| 225        | 5 STARS           | No       | CreditRating               |
| 543        | PTSAGBBOXX        | No       | InstrRegistry              |

## Message trailer

The message trailer fields are shown in the example that is provided here. These fields are taken from the example file that installs with the Pack for Financial Payments, FIX component.

```
93=16
89=.....
10=235
```

- **Trailer fields**

The fields in the FIX message trailer are shown in the following table. These fields are taken from the example file that installs with the Pack for Financial Payments, FIX component.

## Trailer fields

The fields in the FIX message trailer are shown in the following table. These fields are taken from the example file that installs with the Pack for Financial Payments, FIX component.

| Tag Number | Value | Required | Header field    |
|------------|-------|----------|-----------------|
| 93         | 16    | No       | SignatureLength |
| 89         | ..... | No       | Signature       |
| 10         | 235   | Yes      | CheckSum        |

## Additional validation recommendations

The following conditions are not validated in the FIX type trees. You can extend the validation in your implementation through more mapping or lookups.

| Condition                                   | Explanation                                                                                                               |
|---------------------------------------------|---------------------------------------------------------------------------------------------------------------------------|
| Uniqueness of sequence number.              | Every message must have a unique sequence number.                                                                         |
| Reference to fields from previous messages. | Certain conditional rules require field checks from message that is sent or previously processed.                         |
| ReservedXXXXPlus data types.                | Pattern data type used to allow additional bilaterally agreed upon enumerations to be defined starting at XXXX and above. |

| Condition                                                                                                                | Explanation                                                                                                                                                                                                                                                                                                                      |
|--------------------------------------------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| User-defined fields.                                                                                                     | Member firms can add their own custom fields, which are published on the FIX website.                                                                                                                                                                                                                                            |
| Bank Identifier Code (BIC), International Securities Identification Numbering System (ISINS) validation, or validations. | Certain fields require a BIC and an International ISINS number or numbers.                                                                                                                                                                                                                                                       |
| Encryption.                                                                                                              | Certain fields are encrypted, that is, tag #91 Secure Data.                                                                                                                                                                                                                                                                      |
| Checksum integrity check.                                                                                                | Verifies the check digit value that is specified in the field tag number 10 of the FIX message, against the computed checksum. The computed checksum is the actual sum of the binary value of each character of the message, up to but not including, the checksum field itself. This sum is then transformed into a modulo 256. |
| Fields that have a data type of UTCTimestamp.                                                                            | Leap Seconds are inserted to account for slowing of the rotation of the earth. During a leap second insertion, a UTCTimestamp field might read "19981231-23:59:59", "19981231-23:59:60", "19990101-00:00:00". Limitation is on Coordinated Universal Time (UTC) leap seconds set to a value of "60".                             |
| Coding schemes                                                                                                           | Restricted set of values that are associated with a URI. The URI can be assigned by an external body, or by FpML.                                                                                                                                                                                                                |
| XID data types                                                                                                           | The constraint added by this data type is that the values of all of the fields that have an ID data type in a FIX message must be unique.                                                                                                                                                                                        |
| XIDREF data types                                                                                                        | Defines a reference to an identifier that is defined by the XID data type.                                                                                                                                                                                                                                                       |

## FIXML type trees

The FIXML type trees are listed in [The trees subdirectory](#). The documentation provided here describes FIXML message protocol and defines message types by FIXML version number

- [FIXML message protocol](#)

The FIX protocol is defined at the application level.

---

## FIXML message protocol

The FIX protocol is defined at the application level.

The application level consists of:

- Pre-trade
- Trade
- Post-trade
- Other
- [Import source](#)  
The IBM Transformation Extender Schema Importer is used to generate the FIXML type trees.
- [Message types per FIXML version number](#)  
This documentation shows the message types that are supported in the Pack for Financial Payments, FIX component. Messages are listed by FIXML version.

---

## Import source

The IBM Transformation Extender Schema Importer is used to generate the FIXML type trees.

The type trees are based upon the following DTDs and schemas:

- fixml\_41.mtt - based from FIXML 4.1 DTD version 20000929
- fixml\_42.mtt - based from FIXML 4.2 DTD version 20010330
- fixml\_43.mtt - based from FIXML 4.3 DTD version 20020920
- fixml\_44.mtt - based from FIXML 4.4 schema version 20040109
- fixml\_50.mtt - based from FIXML 5.0 schema version 20070103
- fixml\_50\_sp1.mtt - based from FIXML 5.0 SP1 schema version 20080314
- fixml-main-Latest.mtt - based from FIXML Latest schema version 20240704

You must obtain these DTDs directly from FIX. The DTDs are copied to the DTDs to the same location where the compiled map that references any of the FIXML 4.1, 4.2 or 4.3 type trees is being executed.

Copies of the FIXML schemas are shipped with the Pack for Financial Payments, FIX component. The FIXML schemas are installed in the following directory:

<install\_dir>/packs/financial\_payments\_vn.n.n.n/fix/schemas

The minimum version of the IBM Transformation Extender base product that is required by the Pack for Financial Payments, FIX component allows maps to work directly with "native" schemas. The pack does not ship "imported" type trees that are based on FIXML 5.0 SP2 schema. An example of such a scenario where FIXML schemas are being used directly in the maps is found in the sample conversion maps between FIX 5.0 SP2 and FIXML 5.0 SP2 under the <install\_dir>/packs/financial\_payments\_vn.n.n.n/fix/example folder.

---

## Message types per FIXML version number

This documentation shows the message types that are supported in the Pack for Financial Payments, FIX component. Messages are listed by FIXML version.

- [\*\*Session messages\*\*](#)  
The table shows a summary of the messages under the Session category. The messages are listed by FIX version number.
- [\*\*EventCommunication messages\*\*](#)  
The table shows a summary of the messages under the Event Communication category. The messages are listed by FIXML version number.
- [\*\*Indication messages\*\*](#)  
The table shows a summary of the messages under the Indication category. The messages are listed by FIXML version number.
- [\*\*QuotationNegotiation messages\*\*](#)  
The table shows a summary of the messages under the QuotationNegotiation category. The messages are listed by FIXML version number.
- [\*\*MarketData messages\*\*](#)  
The table shows a summary of the messages under the MarketData category. The message types are listed by FIXML version number.
- [\*\*SecuritiesReferenceData messages\*\*](#)  
The table shows a summary of the messages under the SecuritiesReferenceData category. The messages are listed by FIXML version number.
- [\*\*MarketStructureReferenceData messages\*\*](#)  
The table shows a summary of the messages under the MarketStructureReferenceData category. The messages are listed by FIXML version number.
- [\*\*PartiesAction messages\*\*](#)  
The table shows a summary of the messages under the PartiesAction category. The messages are listed by FIXML version number.
- [\*\*PartiesReferenceData messages\*\*](#)  
The table shows a summary of the messages under the PartiesReferenceData category. The messages are listed by FIXML version number.
- [\*\*SingleGeneralOrderHandling messages\*\*](#)  
The table shows a summary of the messages under the SingleGeneralOrderHandling category. The messages are listed by FIXML version number.
- [\*\*ProgramTrading messages\*\*](#)  
The table shows a summary of the messages under the ProgramTrading category. The messages are listed by FIXML version number.
- [\*\*OrderMassHandling messages\*\*](#)  
The table shows a summary of the messages under the OrderMassHandling category. The messages are listed by FIXML version number.
- [\*\*CrossOrders messages\*\*](#)  
The table shows a summary of the messages under the CrossOrders category. The messages are listed by FIXML version number.
- [\*\*MultilegOrders messages\*\*](#)  
The table shows a summary of the messages under the MultilegOrders category. The messages are listed by FIXML version number.
- [\*\*Allocation messages\*\*](#)  
The table shows a summary of the messages under the Allocation category. The messages are listed by FIXML version number.
- [\*\*SettlementInstruction messages\*\*](#)  
The table shows a summary of the messages under the SettlementInstruction category. The messages are listed by FIXML version number.
- [\*\*SettlementStatusManagement messages\*\*](#)  
The table shows a summary of the messages under the SettlementStatusManagement category. The messages are listed by FIXML version number.
- [\*\*RegistrationInstructions messages\*\*](#)  
The table shows a summary of the messages under the RegistrationInstructions category. The messages are listed by FIXML version number.
- [\*\*TradeCapture messages\*\*](#)  
The table shows a summary of the messages under the TradeCapture category. The message types are listed by FIXML version number.
- [\*\*TradeManagement messages\*\*](#)  
The table shows a summary of the messages under the TradeManagement category. The message types are listed by FIXML version number.
- [\*\*Confirmation messages\*\*](#)  
The table shows a summary of the messages under the Confirmation category. The message types are listed by FIXML version number.
- [\*\*PayManagement messages\*\*](#)  
The table shows a summary of the messages under the PayManagement category. The messages are listed by FIXML version number.
- [\*\*PositionMaintenance messages\*\*](#)  
The table shows a summary of the messages under the PositionMaintenance category. The messages are listed by FIXML version number.
- [\*\*CollateralManagement messages\*\*](#)  
The table shows a summary of the messages under the CollateralManagement category. The messages are listed by FIXML version number.
- [\*\*MarginRequirementManagement messages\*\*](#)  
The table shows a summary of the messages under the MarginRequirementManagement category. The messages are listed by FIXML version number.
- [\*\*AccountReporting messages\*\*](#)  
The table shows a summary of the messages under the AccountReporting category. The messages are listed by FIXML version number.
- [\*\*BusinessReject messages\*\*](#)  
The table shows a summary of the messages under the BusinessReject category. The messages are listed by FIXML version number.
- [\*\*Network messages\*\*](#)  
The table shows a summary of the messages under the Network category. The messages are listed by FIXML version number.
- [\*\*UserManagement messages\*\*](#)  
The table shows a summary of the messages under the UserManagement category. The messages are listed by FIXML version number.
- [\*\*Application messages\*\*](#)  
The table shows a summary of the messages under the Application category. The messages are listed by FIXML version number.

## Session messages

The table shows a summary of the messages under the Session category. The messages are listed by FIX version number.

| Session messages | FIX<br>4.0 | FIX<br>4.1 | FIX<br>4.2 | FIX<br>4.3 | FIX<br>4.4 | FIX<br>5.0 | FIX<br>SP1 | FIX<br>SP2 | FIX<br>(EP291)<br>LATEST |
|------------------|------------|------------|------------|------------|------------|------------|------------|------------|--------------------------|
| Heartbeat        | x          | x          | x          | x          | x          | x          | x          | x          | x                        |
| Logon            | x          | x          | x          | x          | x          | x          | x          | x          | x                        |
| TestRequest      | x          | x          | x          | x          | x          | x          | x          | x          | x                        |
| ResendRequest    | x          | x          | x          | x          | x          | x          | x          | x          | x                        |
| Reject           | x          | x          | x          | x          | x          | x          | x          | x          | x                        |
| SequenceReset    | x          | x          | x          | x          | x          | x          | x          | x          | x                        |

| Session messages | FIX 4.0 | FIX 4.1 | FIX 4.2 | FIX 4.3 | FIX 4.4 | FIX 5.0 SP1 | FIX 5.0 SP2 | FIX LATEST (EP291) |
|------------------|---------|---------|---------|---------|---------|-------------|-------------|--------------------|
| Logout           | x       | x       | x       | x       | x       | x           | x           | x                  |
| XMLnonFIX        |         |         |         | x       | x       | x           | x           | x                  |

## EventCommunication messages

The table shows a summary of the messages under the Event Communication category. The messages are listed by FIXML version number.

| EventCommunication messages | FIXML 4.1 | FIXML 4.2 | FIXML 4.3 | FIXML 4.4 | FIXML 5.0 | FIXML 5.0 SP1 | FIXML 5.0 SP2 | FIXML LATEST (EP291) |
|-----------------------------|-----------|-----------|-----------|-----------|-----------|---------------|---------------|----------------------|
| Email                       | x         | x         | x         | x         | x         | x             | x             | x                    |
| News                        | x         | x         | x         | x         | x         | x             | x             | x                    |

## Indication messages

The table shows a summary of the messages under the Indication category. The messages are listed by FIXML version number.

| Indication messages | FIX 4.0 | FIX 4.1 | FIX 4.2 | FIX 4.3 | FIX 4.4 | FIX 5.0 | FIX 5.0 SP1 | FIX 5.0 SP2 | FIX LATEST (EP291) |
|---------------------|---------|---------|---------|---------|---------|---------|-------------|-------------|--------------------|
| Advertisement       | x       | x       | x       | x       | x       | x       | x           | x           | x                  |
| CrossRequest        |         |         |         |         |         |         |             |             | x                  |
| CrossRequestAck     |         |         |         |         |         |         |             |             | x                  |
| IOI                 | x       | x       | x       | x       | x       | x       | x           | x           | x                  |

## QuotationNegotiation messages

The table shows a summary of the messages under the QuotationNegotiation category. The messages are listed by FIXML version number.

| QuotationNegotiation messages | FIXML 4.1 | FIXML 4.2 | FIXML 4.3 | FIXML 4.4 | FIXML 5.0 | FIXML 5.0 SP1 | FIXML 5.0 SP2 | FIXML LATEST (EP291) |
|-------------------------------|-----------|-----------|-----------|-----------|-----------|---------------|---------------|----------------------|
| MassQuote                     |           | x         | x         | x         | x         | x             | x             | x                    |
| MassQuoteAck                  |           | x         | x         | x         | x         | x             | x             | x                    |
| Quote                         | x         | x         | x         | x         | x         | x             | x             | x                    |
| QuoteAck                      |           |           |           |           |           |               |               | x                    |
| QuoteCancel                   |           | x         | x         | x         | x         | x             | x             | x                    |
| QuoteRequest                  | x         | x         | x         | x         | x         | x             | x             | x                    |
| QuoteRequestReject            |           |           | x         | x         | x         | x             | x             | x                    |
| QuoteResponse                 |           |           |           | x         | x         | x             | x             | x                    |
| QuoteStatusReport             |           |           | x         | x         | x         | x             | x             | x                    |
| QuoteStatusRequest            |           | x         | x         | x         | x         | x             | x             | x                    |
| RFQRequest                    |           |           | x         | x         | x         | x             | x             | x                    |

## MarketData messages

The table shows a summary of the messages under the MarketData category. The message types are listed by FIXML version number.

| MarketData messages          | FIXML 4.1 | FIXML 4.2 | FIXML 4.3 | FIXML 4.4 | FIXML 5.0 | FIXML 5.0 SP1 | FIXML 5.0 SP2 | FIXML LATEST (EP291) |
|------------------------------|-----------|-----------|-----------|-----------|-----------|---------------|---------------|----------------------|
| MarketDataIncrementalRefresh |           | x         | x         | x         | x         | x             | x             | x                    |
| MarketDataReport             |           |           |           |           |           |               |               | x                    |
| MarketDataRequest            |           | x         | x         | x         |           |               | x             | x                    |

| MarketData messages           | <b>FIXML 4.1</b> | <b>FIXML 4.2</b> | <b>FIXML 4.3</b> | <b>FIXML 4.4</b> | <b>FIXML 5.0</b> | <b>FIXML 5.0 SP1</b> | <b>FIXML 5.0 SP2</b> | <b>FIXML LATEST (EP291)</b> |
|-------------------------------|------------------|------------------|------------------|------------------|------------------|----------------------|----------------------|-----------------------------|
| MarketDataRequestReject       |                  | x                | x                | x                | x                | x                    | x                    | x                           |
| MarketDataSnapshotFullRefresh |                  | x                | x                | x                | x                | x                    | x                    | x                           |
| MarketDataStatisticsReport    |                  |                  |                  |                  |                  |                      |                      | x                           |
| MarketDataStatisticsRequest   |                  |                  |                  |                  |                  |                      |                      | x                           |
| StreamAssignmentReport        |                  |                  |                  |                  |                  |                      | x                    | x                           |
| StreamAssignmentReportAck     |                  |                  |                  |                  |                  |                      | x                    | x                           |
| StreamAssignmentRequest       |                  |                  |                  |                  |                  |                      | x                    | x                           |

## SecuritiesReferenceData messages

The table shows a summary of the messages under the SecuritiesReferenceData category. The messages are listed by FIXML version number.

| SecuritiesReference Data messages  | <b>FIXML 4.1</b> | <b>FIXML 4.2</b> | <b>FIXML 4.3</b> | <b>FIXML 4.4</b> | <b>FIXML 5.0</b> | <b>FIXML 5.0 SP1</b> | <b>FIXML 5.0 SP2</b> | <b>FIXML LATEST (EP291)</b> |
|------------------------------------|------------------|------------------|------------------|------------------|------------------|----------------------|----------------------|-----------------------------|
| DerivativeSecurityList             |                  |                  | x                | x                | x                | x                    | x                    | x                           |
| DerivativeSecurityListRequest      |                  |                  | x                | x                | x                | x                    | x                    | x                           |
| DerivativeSecurityListUpdateReport |                  |                  |                  |                  |                  | x                    | x                    | x                           |
| SecurityDefinition                 | x                | x                | x                | x                | x                | x                    | x                    | x                           |
| SecurityDefinitionRequest          | x                | x                | x                | x                | x                | x                    | x                    | x                           |
| SecurityDefinitionUpdateReport     |                  |                  |                  |                  | x                | x                    | x                    | x                           |
| SecurityList                       |                  |                  | x                | x                | x                | x                    | x                    | x                           |
| SecurityListRequest                |                  |                  | x                | x                | x                | x                    | x                    | x                           |
| SecurityListUpdateReport           |                  |                  |                  |                  | x                | x                    | x                    | x                           |
| SecurityMassStatus                 |                  |                  |                  |                  |                  |                      |                      | x                           |
| SecurityMassStatusRequest          |                  |                  |                  |                  |                  |                      |                      | x                           |
| SecurityRiskMetricsReport          |                  |                  |                  |                  |                  |                      |                      | x                           |
| SecurityStatus                     | x                | x                | x                | x                | x                | x                    | x                    | x                           |
| SecurityStatusRequest              | x                | x                | x                | x                | x                | x                    | x                    | x                           |
| SecurityTypeRequest                |                  |                  | x                | x                | x                | x                    | x                    | x                           |
| SecurityTypes                      |                  |                  | x                | x                | x                | x                    | x                    | x                           |

## MarketStructureReferenceData messages

The table shows a summary of the messages under the MarketStructureReferenceData category. The messages are listed by FIXML version number.

| MarketStructureReferenceData messages | <b>FIXML 4.1</b> | <b>FIXML 4.2</b> | <b>FIXML 4.3</b> | <b>FIXML 4.4</b> | <b>FIXML 5.0</b> | <b>FIXML 5.0 SP1</b> | <b>FIXML 5.0 SP2</b> | <b>FIXML LATEST (EP291)</b> |
|---------------------------------------|------------------|------------------|------------------|------------------|------------------|----------------------|----------------------|-----------------------------|
| MarketDefinition                      |                  |                  |                  |                  |                  | x                    | x                    | x                           |
| MarketDefinitionRequest               |                  |                  |                  |                  |                  | x                    | x                    | x                           |
| MarketDefinitionUpdateReport          |                  |                  |                  |                  |                  | x                    | x                    | x                           |
| TradingSessionList                    |                  |                  |                  |                  | x                | x                    | x                    | x                           |
| TradingSessionListRequest             |                  |                  |                  |                  | x                | x                    | x                    | x                           |

| MarketStructureReferenceData messages | <b>FIXML<br/>4.1</b> | <b>FIXML<br/>4.2</b> | <b>FIXML<br/>4.3</b> | <b>FIXML<br/>4.4</b> | <b>FIXML<br/>5.0</b> | <b>FIXML<br/>5.0<br/>SP1</b> | <b>FIXML<br/>5.0<br/>SP2</b> | <b>FIXML<br/>LATEST<br/>(EP291)</b> |
|---------------------------------------|----------------------|----------------------|----------------------|----------------------|----------------------|------------------------------|------------------------------|-------------------------------------|
| TradingSessionListUpdateRequest       |                      |                      |                      |                      |                      | x                            | x                            | x                                   |
| TradingSessionStatus                  |                      | x                    | x                    | x                    | x                    | x                            | x                            | x                                   |
| TradingSessionStatusRequest           |                      | x                    | x                    | x                    | x                    | x                            | x                            | x                                   |

## PartiesAction messages

The table shows a summary of the messages under the PartiesAction category. The messages are listed by FIXML version number.

| PartiesAction messages        | <b>FIXML<br/>4.1</b> | <b>FIXML<br/>4.2</b> | <b>FIXML<br/>4.3</b> | <b>FIXML<br/>4.4</b> | <b>FIXML<br/>5.0</b> | <b>FIXML<br/>5.0<br/>SP1</b> | <b>FIXML<br/>5.0<br/>SP2</b> | <b>FIXML<br/>LATEST<br/>(EP291)</b> |
|-------------------------------|----------------------|----------------------|----------------------|----------------------|----------------------|------------------------------|------------------------------|-------------------------------------|
| PartyRiskLimitCheckRequest    |                      |                      |                      |                      |                      |                              |                              | x                                   |
| PartyRiskLimitCheckRequestAck |                      |                      |                      |                      |                      |                              |                              | x                                   |
| PartyActionRequest            |                      |                      |                      |                      |                      |                              |                              | x                                   |
| PartyActionReport             |                      |                      |                      |                      |                      |                              |                              | x                                   |

## PartiesReferenceData messages

The table shows a summary of the messages under the PartiesReferenceData category. The messages are listed by FIXML version number.

| PartiesReferenceData messages         | <b>FIXML<br/>4.1</b> | <b>FIXML<br/>4.2</b> | <b>FIXML<br/>4.3</b> | <b>FIXML<br/>4.4</b> | <b>FIXML<br/>5.0</b> | <b>FIXML<br/>5.0<br/>SP1</b> | <b>FIXML<br/>5.0<br/>SP2</b> | <b>FIXML<br/>LATEST<br/>(EP291)</b> |
|---------------------------------------|----------------------|----------------------|----------------------|----------------------|----------------------|------------------------------|------------------------------|-------------------------------------|
| PartyDetailsDefinitionRequest         |                      |                      |                      |                      |                      |                              |                              | x                                   |
| PartyDetailsDefinitionRequestAck      |                      |                      |                      |                      |                      |                              |                              | x                                   |
| PartyDetailsListReport                |                      |                      |                      |                      |                      |                              |                              | x                                   |
| PartyDetailsListRequest               |                      |                      |                      |                      |                      |                              |                              | x                                   |
| PartyDetailsListUpdateReport          |                      |                      |                      |                      |                      |                              |                              | x                                   |
| PartyEntitlementsDefinitionRequest    |                      |                      |                      |                      |                      |                              |                              | x                                   |
| PartyEntitlementsDefinitionRequestAck |                      |                      |                      |                      |                      |                              |                              | x                                   |
| PartyEntitlementsReport               |                      |                      |                      |                      |                      |                              |                              | x                                   |
| PartyEntitlementsRequest              |                      |                      |                      |                      |                      |                              |                              | x                                   |
| PartyEntitlementsUpdateReport         |                      |                      |                      |                      |                      |                              |                              | x                                   |
| PartyRiskLimitsDefinitionRequest      |                      |                      |                      |                      |                      |                              |                              | x                                   |
| PartyRiskLimitsDefinitionRequestAck   |                      |                      |                      |                      |                      |                              |                              | x                                   |
| PartyRiskLimitsReport                 |                      |                      |                      |                      |                      |                              |                              | x                                   |
| PartyRiskLimitsRequest                |                      |                      |                      |                      |                      |                              |                              | x                                   |
| PartyRiskLimitsUpdateReport           |                      |                      |                      |                      |                      |                              |                              | x                                   |
| PartyRiskLimitsReportAck              |                      |                      |                      |                      |                      |                              |                              | x                                   |

## SingleGeneralOrderHandling messages

The table shows a summary of the messages under the SingleGeneralOrderHandling category. The messages are listed by FIXML version number.

| SingleGeneralOrderHandling messages | <b>FIXML 4.1</b> | <b>FIXML 4.2</b> | <b>FIXML 4.3</b> | <b>FIXML 4.4</b> | <b>FIXML 5.0</b> | <b>FIXML 5.0 SP1</b> | <b>FIXML 5.0 SP2</b> | <b>FIXML LATEST (EP291)</b> |
|-------------------------------------|------------------|------------------|------------------|------------------|------------------|----------------------|----------------------|-----------------------------|
| DontKnowTrade                       | x                | x                | x                | x                | x                | x                    | x                    | x                           |
| ExecutionAck                        |                  |                  |                  |                  | x                | x                    | x                    | x                           |
| ExecutionReport                     | x                | x                | x                | x                | x                | x                    | x                    | x                           |
| NewOrderSingle                      | x                | x                | x                | x                | x                | x                    | x                    | x                           |
| OrderCancelReject                   | x                | x                | x                | x                | x                | x                    | x                    | x                           |
| OrderCancelReplaceRequest           | x                | x                | x                | x                | x                | x                    | x                    | x                           |
| OrderCancelRequest                  | x                | x                | x                | x                | x                | x                    | x                    | x                           |
| OrderStatusRequest                  | x                | x                | x                | x                | x                | x                    | x                    | x                           |

## ProgramTrading messages

The table shows a summary of the messages under the ProgramTrading category. The messages are listed by FIXML version number.

| ProgramTrading messages | <b>FIXML 4.1</b> | <b>FIXML 4.2</b> | <b>FIXML 4.3</b> | <b>FIXML 4.4</b> | <b>FIXML 5.0</b> | <b>FIXML 5.0 SP1</b> | <b>FIXML 5.0 SP2</b> | <b>FIXML LATEST (EP291)</b> |
|-------------------------|------------------|------------------|------------------|------------------|------------------|----------------------|----------------------|-----------------------------|
| BidRequest              |                  | x                | x                | x                | x                | x                    | x                    | x                           |
| BidResponse             |                  | x                | x                | x                | x                | x                    | x                    | x                           |
| ListCancelRequest       | x                | x                | x                | x                | x                | x                    | x                    | x                           |
| ListExecute             | x                | x                | x                | x                | x                | x                    | x                    | x                           |
| ListStatus              | x                | x                | x                | x                | x                | x                    | x                    | x                           |
| ListStatusRequest       | x                | x                | x                | x                | x                | x                    | x                    | x                           |
| ListStrikePrice         |                  | x                | x                | x                | x                | x                    | x                    | x                           |
| NewOrderList            | x                | x                | x                | x                | x                | x                    | x                    | x                           |

## OrderMassHandling messages

The table shows a summary of the messages under the OrderMassHandling category. The messages are listed by FIXML version number.

| OrderMassHandling messages | <b>FIXML 4.1</b> | <b>FIXML 4.2</b> | <b>FIXML 4.3</b> | <b>FIXML 4.4</b> | <b>FIXML 5.0</b> | <b>FIXML 5.0 SP1</b> | <b>FIXML 5.0 SP2</b> | <b>FIXML LATEST (EP291)</b> |
|----------------------------|------------------|------------------|------------------|------------------|------------------|----------------------|----------------------|-----------------------------|
| MassOrder                  |                  |                  |                  |                  |                  |                      |                      | x                           |
| MassOrderAck               |                  |                  |                  |                  |                  |                      |                      | x                           |
| OrderMassActionReport      |                  |                  |                  |                  |                  | x                    | x                    | x                           |
| OrderMassActionRequest     |                  |                  |                  |                  |                  | x                    | x                    | x                           |
| OrderMassCancelReport      |                  |                  | x                | x                | x                | x                    | x                    | x                           |
| OrderMassCancelRequest     |                  |                  | x                | x                | x                | x                    | x                    | x                           |
| OrderMassStatusRequest     |                  |                  | x                | x                | x                | x                    | x                    | x                           |

## CrossOrders messages

The table shows a summary of the messages under the CrossOrders category. The messages are listed by FIXML version number.

| CrossOrders messages           | <b>FIXML 4.1</b> | <b>FIXML 4.2</b> | <b>FIXML 4.3</b> | <b>FIXML 4.4</b> | <b>FIXML 5.0</b> | <b>FIXML 5.0 SP1</b> | <b>FIXML 5.0 SP2</b> | <b>FIXML LATEST (EP291)</b> |
|--------------------------------|------------------|------------------|------------------|------------------|------------------|----------------------|----------------------|-----------------------------|
| CrossOrderCancelReplaceRequest |                  |                  | x                | x                | x                | x                    | x                    | x                           |

| CrossOrders messages    | <b>FIXML 4.1</b> | <b>FIXML 4.2</b> | <b>FIXML 4.3</b> | <b>FIXML 4.4</b> | <b>FIXML 5.0</b> | <b>FIXML 5.0 SP1</b> | <b>FIXML 5.0 SP2</b> | <b>FIXML LATEST (EP291)</b> |
|-------------------------|------------------|------------------|------------------|------------------|------------------|----------------------|----------------------|-----------------------------|
| CrossOrderCancelRequest |                  |                  | x                | x                | x                | x                    | x                    | x                           |
| NewOrderCross           |                  |                  | x                | x                | x                | x                    | x                    | x                           |

## MultilegOrders messages

The table shows a summary of the messages under the MultilegOrders category. The messages are listed by FIXML version number.

| MultilegOrders messages    | <b>FIXML 4.1</b> | <b>FIXML 4.2</b> | <b>FIXML 4.3</b> | <b>FIXML 4.4</b> | <b>FIXML 5.0</b> | <b>FIXML 5.0 SP1</b> | <b>FIXML 5.0 SP2</b> | <b>FIXML LATEST (EP291)</b> |
|----------------------------|------------------|------------------|------------------|------------------|------------------|----------------------|----------------------|-----------------------------|
| MultilegOrderCancelReplace |                  |                  | x                | x                | x                | x                    | x                    | x                           |
| NewOrderMultileg           |                  |                  | x                | x                | x                | x                    | x                    | x                           |

## Allocation messages

The table shows a summary of the messages under the Allocation category. The messages are listed by FIXML version number.

| Allocation messages                  | <b>FIXML 4.1</b> | <b>FIXML 4.2</b> | <b>FIXML 4.3</b> | <b>FIXML 4.4</b> | <b>FIXML 5.0</b> | <b>FIXML 5.0 SP1</b> | <b>FIXML 5.0 SP2</b> | <b>FIXML LATEST (EP291)</b> |
|--------------------------------------|------------------|------------------|------------------|------------------|------------------|----------------------|----------------------|-----------------------------|
| AllocationInstruction                | x                | x                | x                | x                | x                | x                    | x                    | x                           |
| AllocationInstructionAck             | x                | x                | x                | x                | x                | x                    | x                    | x                           |
| AllocationInstructionAlert           |                  |                  |                  |                  | x                | x                    | x                    | x                           |
| AllocationInstructionAlertRequest    |                  |                  |                  |                  |                  |                      |                      | x                           |
| AllocationInstructionAlertRequestAck |                  |                  |                  |                  |                  |                      |                      | x                           |
| AllocationReport                     |                  |                  |                  | x                | x                | x                    | x                    | x                           |
| AllocationReportAck                  |                  |                  |                  | x                | x                | x                    | x                    | x                           |

## SettlementInstruction messages

The table shows a summary of the messages under the SettlementInstruction category. The messages are listed by FIXML version number.

| SettlementInstruction messages | <b>FIXML 4.1</b> | <b>FIXML 4.2</b> | <b>FIXML 4.3</b> | <b>FIXML 4.4</b> | <b>FIXML 5.0</b> | <b>FIXML 5.0 SP1</b> | <b>FIXML 5.0 SP2</b> | <b>FIXML LATEST (EP291)</b> |
|--------------------------------|------------------|------------------|------------------|------------------|------------------|----------------------|----------------------|-----------------------------|
| SettlementInstructionRequest   |                  |                  |                  | x                | x                | x                    | x                    | x                           |
| SettlementInstructions         | x                | x                | x                | x                | x                | x                    | x                    | x                           |
| SettlementObligationReport     |                  |                  |                  |                  |                  | x                    | x                    | x                           |

## SettlementStatusManagement messages

The table shows a summary of the messages under the SettlementStatusManagement category. The messages are listed by FIXML version number.

| SettlementStatusManagement messages | <b>FIX 4.0</b> | <b>FIX 4.1</b> | <b>FIX 4.2</b> | <b>FIX 4.3</b> | <b>FIX 4.4</b> | <b>FIX 5.0</b> | <b>FIX 5.0 SP1</b> | <b>FIX 5.0 SP2</b> | <b>LATEST (EP291)</b> |
|-------------------------------------|----------------|----------------|----------------|----------------|----------------|----------------|--------------------|--------------------|-----------------------|
| SettlementStatusReport              |                |                |                |                |                |                |                    | x                  |                       |

| SettlementStatusManagement messages | FIX<br>4.0 | FIX<br>4.1 | FIX<br>4.2 | FIX<br>4.3 | FIX<br>4.4 | FIX<br>5.0 | FIX<br>SP1 | FIX<br>SP2 | LATEST<br>(EP291) |
|-------------------------------------|------------|------------|------------|------------|------------|------------|------------|------------|-------------------|
| SettlementStatusReportAck           |            |            |            |            |            |            |            | x          |                   |
| SettlementStatusRequest             |            |            |            |            |            |            |            | x          |                   |
| SettlementStatusRequestAck          |            |            |            |            |            |            |            | x          |                   |

## RegistrationInstructions messages

The table shows a summary of the messages under the RegistrationInstructions category. The messages are listed by FIXML version number.

| RegistrationInstructions messages | FIXML<br>4.1 | FIXML<br>4.2 | FIXML<br>4.3 | FIXML<br>4.4 | FIXML<br>5.0 | FIXML<br>5.0<br>SP1 | FIXML<br>5.0<br>SP2 | FIXML<br>LATEST<br>(EP291) |
|-----------------------------------|--------------|--------------|--------------|--------------|--------------|---------------------|---------------------|----------------------------|
| RegistrationInstructions          |              |              | x            | x            | x            | x                   | x                   | x                          |
| RegistrationInstructionsResponse  |              |              | x            | x            | x            | x                   | x                   | x                          |

## TradeCapture messages

The table shows a summary of the messages under the TradeCapture category. The message types are listed by FIXML version number.

| TradeCapture messages        | FIXML<br>4.1 | FIXML<br>4.2 | FIXML<br>4.3 | FIXML<br>4.4 | FIXML<br>5.0 | FIXML<br>5.0<br>SP1 | FIXML<br>5.0<br>SP2 | FIXML<br>LATEST<br>(EP291) |
|------------------------------|--------------|--------------|--------------|--------------|--------------|---------------------|---------------------|----------------------------|
| TradeCaptureReport           |              |              | x            | x            | x            | x                   | x                   | x                          |
| TradeCaptureReportAck        |              |              |              | x            | x            | x                   | x                   | x                          |
| TradeCaptureReportRequest    |              |              | x            | x            | x            | x                   | x                   | x                          |
| TradeCaptureReportRequestAck |              |              |              | x            | x            | x                   | x                   | x                          |
| TradeMatchReport             |              |              |              |              |              |                     |                     | x                          |
| TradeMatchReportAck          |              |              |              |              |              |                     |                     | x                          |

## TradeManagement messages

The table shows a summary of the messages under the TradeManagement category. The message types are listed by FIXML version number.

| TradeManagement messages | FIXML<br>4.1 | FIXML<br>4.2 | FIXML<br>4.3 | FIXML<br>4.4 | FIXML<br>5.0 | FIXML<br>5.0<br>SP1 | FIXML<br>5.0<br>SP2 | FIXML<br>LATEST<br>(EP291) |
|--------------------------|--------------|--------------|--------------|--------------|--------------|---------------------|---------------------|----------------------------|
| TradeAggregationReport   |              |              |              |              |              |                     |                     | x                          |
| TradeAggregationRequest  |              |              |              |              |              |                     |                     | x                          |

## Confirmation messages

The table shows a summary of the messages under the Confirmation category. The message types are listed by FIXML version number.

| Confirmation messages | FIXML<br>4.1 | FIXML<br>4.2 | FIXML<br>4.3 | FIXML<br>4.4 | FIXML<br>5.0 | FIXML<br>5.0<br>SP1 | FIXML<br>5.0<br>SP2 | FIXML<br>LATEST<br>(EP291) |
|-----------------------|--------------|--------------|--------------|--------------|--------------|---------------------|---------------------|----------------------------|
| Confirmation          |              |              |              | x            | x            | x                   | x                   | x                          |
| ConfirmationAck       |              |              |              | x            | x            | x                   | x                   | x                          |
| ConfirmationRequest   |              |              |              | x            | x            | x                   | x                   | x                          |

## PayManagement messages

The table shows a summary of the messages under the PayManagement category. The messages are listed by FIXML version number.

| PayManagement messages  | FIXML 4.1 | FIXML 4.2 | FIXML 4.3 | FIXML 4.4 | FIXML 5.0 | FIXML 5.0 SP1 | FIXML 5.0 SP2 | FIXML LATEST (EP291) |
|-------------------------|-----------|-----------|-----------|-----------|-----------|---------------|---------------|----------------------|
| PayManagementReport     |           |           |           |           |           |               |               | x                    |
| PayManagementReportAck  |           |           |           |           |           |               |               | x                    |
| PayManagementRequest    |           |           |           |           |           |               |               | x                    |
| PayManagementRequestAck |           |           |           |           |           |               |               | x                    |

## PositionMaintenance messages

The table shows a summary of the messages under the PositionMaintenance category. The messages are listed by FIXML version number.

| PositionMaintenance messages   | FIXML 4.1 | FIXML 4.2 | FIXML 4.3 | FIXML 4.4 | FIXML 5.0 | FIXML 5.0 SP1 | FIXML 5.0 SP2 | FIXML LATEST (EP291) |
|--------------------------------|-----------|-----------|-----------|-----------|-----------|---------------|---------------|----------------------|
| AdjustedPositionReport         |           |           |           |           | x         | x             | x             | x                    |
| AssignmentReport               |           |           |           | x         | x         | x             | x             | x                    |
| ContraryIntentionReport        |           |           |           |           | x         | x             | x             | x                    |
| PositionMaintenanceReport      |           |           |           | x         | x         | x             | x             | x                    |
| PositionMaintenanceRequest     |           |           |           | x         | x         | x             | x             | x                    |
| PositionReport                 |           |           |           | x         | x         | x             | x             | x                    |
| PositionTransferInstruction    |           |           |           |           |           |               |               | x                    |
| PositionTransferInstructionAck |           |           |           |           |           |               |               | x                    |
| PositionTransferReport         |           |           |           |           |           |               |               | x                    |
| RequestForPositions            |           |           |           | x         | x         | x             | x             | x                    |
| RequestForPositionsAck         |           |           |           | x         | x         | x             | x             | x                    |

## CollateralManagement messages

The table shows a summary of the messages under the CollateralManagement category. The messages are listed by FIXML version number.

| CollateralManagement messages | FIXML 4.1 | FIXML 4.2 | FIXML 4.3 | FIXML 4.4 | FIXML 5.0 | FIXML 5.0 SP1 | FIXML 5.0 SP2 | FIXML LATEST (EP291) |
|-------------------------------|-----------|-----------|-----------|-----------|-----------|---------------|---------------|----------------------|
| CollateralAssignment          |           |           |           | x         | x         | x             | x             | x                    |
| CollateralInquiry             |           |           |           | x         | x         | x             | x             | x                    |
| CollateralInquiryAck          |           |           |           | x         | x         | x             | x             | x                    |
| CollateralReport              |           |           |           | x         | x         | x             | x             | x                    |
| CollateralReportAck           |           |           |           |           |           |               |               | x                    |
| CollateralRequest             |           |           |           | x         | x         | x             | x             | x                    |
| CollateralResponse            |           |           |           | x         | x         | x             | x             | x                    |

## MarginRequirementManagement messages

The table shows a summary of the messages under the MarginRequirementManagement category. The messages are listed by FIXML version number.

| MarginRequirementManagement messages | <b>FIXML<br/>4.1</b> | <b>FIXML<br/>4.2</b> | <b>FIXML<br/>4.3</b> | <b>FIXML<br/>4.4</b> | <b>FIXML<br/>5.0</b> | <b>FIXML<br/>5.0<br/>SP1</b> | <b>FIXML<br/>5.0<br/>SP2</b> | <b>FIXML<br/>LATEST<br/>(EP291)</b> |
|--------------------------------------|----------------------|----------------------|----------------------|----------------------|----------------------|------------------------------|------------------------------|-------------------------------------|
| MarginRequirementInquiry             |                      |                      |                      |                      |                      |                              |                              | x                                   |
| MarginRequirementInquiryAck          |                      |                      |                      |                      |                      |                              |                              | x                                   |
| MarginRequirementReport              |                      |                      |                      |                      |                      |                              |                              | x                                   |

## AccountReporting messages

The table shows a summary of the messages under the AccountReporting category. The messages are listed by FIXML version number.

| AccountReporting messages | <b>FIXML<br/>4.1</b> | <b>FIXML<br/>4.2</b> | <b>FIXML<br/>4.3</b> | <b>FIXML<br/>4.4</b> | <b>FIXML<br/>5.0</b> | <b>FIXML<br/>5.0<br/>SP1</b> | <b>FIXML<br/>5.0<br/>SP2</b> | <b>FIXML<br/>LATEST<br/>(EP291)</b> |
|---------------------------|----------------------|----------------------|----------------------|----------------------|----------------------|------------------------------|------------------------------|-------------------------------------|
| AccountSummaryReport      |                      |                      |                      |                      |                      |                              |                              | x                                   |

## BusinessReject messages

The table shows a summary of the messages under the BusinessReject category. The messages are listed by FIXML version number.

| BusinessReject messages | <b>FIXML<br/>4.1</b> | <b>FIXML<br/>4.2</b> | <b>FIXML<br/>4.3</b> | <b>FIXML<br/>4.4</b> | <b>FIXML<br/>5.0</b> | <b>FIXML<br/>5.0<br/>SP1</b> | <b>FIXML<br/>5.0<br/>SP2</b> | <b>FIXML<br/>LATEST<br/>(EP291)</b> |
|-------------------------|----------------------|----------------------|----------------------|----------------------|----------------------|------------------------------|------------------------------|-------------------------------------|
| BusinessMessageReject   |                      | x                    | x                    | x                    | x                    | x                            | x                            | x                                   |

## Network messages

The table shows a summary of the messages under the Network category. The messages are listed by FIXML version number.

| Network messages                       | <b>FIXML<br/>4.1</b> | <b>FIXML<br/>4.2</b> | <b>FIXML<br/>4.3</b> | <b>FIXML<br/>4.4</b> | <b>FIXML<br/>5.0</b> | <b>FIXML<br/>5.0<br/>SP1</b> | <b>FIXML<br/>5.0<br/>SP2</b> | <b>FIXML<br/>LATEST<br/>(EP291)</b> |
|----------------------------------------|----------------------|----------------------|----------------------|----------------------|----------------------|------------------------------|------------------------------|-------------------------------------|
| NetworkCounterpartySystemStatusRequest |                      |                      |                      | x                    | x                    | x                            | x                            | x                                   |
| NetworkCounterpartySystemStatusRequest |                      |                      |                      | x                    | x                    | x                            | x                            | x                                   |

## UserManagement messages

The table shows a summary of the messages under the UserManagement category. The messages are listed by FIXML version number.

| UserManagement messages | <b>FIXML<br/>4.1</b> | <b>FIXML<br/>4.2</b> | <b>FIXML<br/>4.3</b> | <b>FIXML<br/>4.4</b> | <b>FIXML<br/>5.0</b> | <b>FIXML<br/>5.0<br/>SP1</b> | <b>FIXML<br/>5.0<br/>SP2</b> | <b>FIXML<br/>LATEST<br/>(EP291)</b> |
|-------------------------|----------------------|----------------------|----------------------|----------------------|----------------------|------------------------------|------------------------------|-------------------------------------|
| UserNotification        |                      |                      |                      |                      |                      | x                            | x                            | x                                   |
| UserRequest             |                      |                      |                      | x                    | x                    | x                            | x                            | x                                   |
| UserResponse            |                      |                      |                      | x                    | x                    | x                            | x                            | x                                   |

## Application messages

The table shows a summary of the messages under the Application category. The messages are listed by FIXML version number.

| <b>Application messages</b>  | <b>FIXML 4.1</b> | <b>FIXML 4.2</b> | <b>FIXML 4.3</b> | <b>FIXML 4.4</b> | <b>FIXML 5.0</b> | <b>FIXML 5.0 SP1</b> | <b>FIXML 5.0 SP2</b> | <b>FIXML LATEST (EP291)</b> |
|------------------------------|------------------|------------------|------------------|------------------|------------------|----------------------|----------------------|-----------------------------|
| ApplicationMessageReport     |                  |                  |                  |                  |                  | x                    | x                    | x                           |
| ApplicationMessageRequest    |                  |                  |                  |                  |                  | x                    | x                    | x                           |
| ApplicationMessageRequestAck |                  |                  |                  |                  |                  | x                    | x                    | x                           |

## FIX component examples

The Pack for Financial Payments, FIX component contains examples to assist you in using the pack. Instructions for using the examples are included in the documentation provided here.

The following conversion examples that install with the pack:

- FIX 5.0 mapping from and to FIXML 5.0
- FIX 5.0 SP1 mapping from and to FIXML 5.0 SP1
- FIX 5.0 SP2 mapping from and to FIXML 5.0 SP2
- FIX 5.0 SP2 EP mapping from and to FIXML 5.0 SP2 EP

The mapping rules used between FIX and FIXML standards are not exhaustive. These rules are intended to give you a starting point in building your own transformations for your own business requirements.

It also includes the example scenarios on how to use the FIX checksum utility maps.

- FIX checksum validation
- FIX checksum generation

On UNIX platforms, be aware of file paths and case sensitivity for schemas and DTD's. Modifications might be needed to ensure that the naming conventions match the case. The input test data (\*.inp and \*.xml) must be posted as binary.

- [\*\*Running the examples\*\*](#)

The example conversion maps use the FIX checksum utility maps for generating a FIX checksum, and also for validating value set to FIX checksum tag 10.

- [\*\*Example for Advertisement message\*\*](#)

This example demonstrates the FIX Advertisement to FIXML translation and FIXML to FIX translation using maps only.

- [\*\*Example for Allocation Instruction message\*\*](#)

This example demonstrates the FIX AllocationInstruction to FIXML translation and FIXML to FIX translation using maps only.

- [\*\*Example for Allocation Instruction Ack message\*\*](#)

This example demonstrates the FIX AllocationInstructionAck to FIXML translation and FIXML to FIX translation using maps only.

- [\*\*Example for Don't Know Trade \(DK\) message\*\*](#)

This example demonstrates the FIX DontKnowTrade to FIXML translation and FIXML to FIX translation using maps only.

- [\*\*Example for Email message\*\*](#)

This example demonstrates the FIX Email to FIXML translation and FIXML to FIX translation using maps only.

- [\*\*Example for Execution Report message\*\*](#)

This example demonstrates the FIX ExecutionReport to FIXML translation and FIXML to FIX translation using maps only.

- [\*\*Example for Indication of interest message\*\*](#)

This example demonstrates the FIX IOI to FIXML translation and FIXML to FIX translation using maps only.

- [\*\*Example for List Cancel Request message\*\*](#)

This example demonstrates the FIX ListCancelRequest to FIXML translation and FIXML to FIX translation using maps only.

- [\*\*Example for List Execute message\*\*](#)

This example demonstrates the FIX ListExecute to FIXML translation and FIXML to FIX translation using maps only.

- [\*\*Example for List Status message\*\*](#)

This example demonstrates the FIX ListStatus to FIXML translation and FIXML to FIX translation using maps only.

- [\*\*Example for List Status Request message\*\*](#)

This example demonstrates the FIX ListStatusRequest to FIXML translation and FIXML to FIX translation using maps only.

- [\*\*Example for New Order List message\*\*](#)

This example demonstrates the FIX NewOrderList to FIXML translation and FIXML to FIX translation using maps only.

- [\*\*Example for News message\*\*](#)

This example demonstrates the FIX News to FIXML translation and FIXML to FIX translation using maps only.

- [\*\*Example for Order Cancel Reject message\*\*](#)

This example demonstrates the FIX OrderCancelReject to FIXML translation and FIXML to FIX translation using maps only.

- [\*\*Example for New Order message\*\*](#)

This example demonstrates the FIX NewOrderSingle to FIXML translation and FIXML to FIX translation using maps only.

- [\*\*Example for Order Status Request message\*\*](#)

This example demonstrates the FIX OrderStatusRequest to FIXML translation and FIXML to FIX translation using maps only.

- [\*\*Example for Quote message\*\*](#)

This example demonstrates the FIX Quote to FIXML translation and FIXML to FIX translation using maps only.

- [\*\*Example for Quote Request message\*\*](#)

This example demonstrates the FIX QuoteRequest to FIXML translation and FIXML to FIX translation using maps only.

## Running the examples

The example conversion maps use the FIX checksum utility maps for generating a FIX checksum, and also for validating value set to FIX checksum tag 10.

- [Before running the example](#)
- [FIX mapping from and to FIXML examples](#)
  - The FIX mapping from and to the FIXML example represents conversions from and to corresponding FIX protocol messages.
- [Running the examples on z/OS](#)
  - This documentation describes how to run this example on z/OS.
- [FIX checksum validation](#)
- [FIX checksum generation](#)

---

## Before running the example

### About this task

Complete the following steps to build the FIX checksum utility maps and have the utility maps ready for use by the example conversion maps.

### Procedure

1. Use IBM Sterling Transformation Extender Design Studio to open the FIX checksum utility map source file. The map source file is in the following location:  
`<install_dir>/packs/financial_payments_vn.n.n.n/fix/examples/fixFML<msg>/maps/fix_checksum.mms`  
Where `<msg>` is any of the FIX messages listed under the FIX component examples section.
  2. Build all of the maps in the FIX checksum utility map source file:
    - fxchksm1\_utility
    - fxchksm2\_validate
    - fxchksm3\_compute
- Note: On UNIX platforms, rename the compiled map extension to .mmc.

---

## FIX mapping from and to FIXML examples

The FIX mapping from and to the FIXML example represents conversions from and to corresponding FIX protocol messages.

### About this task

Run the example maps as described in the following steps:

Note: Throughout this documentation, `<install_dir>` is the installation location of the Pack for Financial Payments, and `n.n.n.n` is the current version of the Pack for Financial Payments.

### Procedure

1. Build the FIX checksum utility maps. See [Before running the example](#).
2. Use IBM Sterling Transformation Extender Design Studio to open the selected map source files in the  
`<install_dir>/packs/financial_payments_vn.n.n.n/fix/examples/fixFML<msg>/maps` directory.  
Where `<msg>` is any of the FIX messages listed under the FIX component examples section.
3. Build all the selected maps in the source files, and then run the transformation maps.
  - [Example scenarios](#)
  - [Example assumptions](#)
    - The following assumptions are made regarding FIX mapping from and to FIXML examples.
  - [Expected results](#)

---

## Example scenarios

The example maps illustrate the following conditions:

- How to convert to and from a FIX message and a FIXML message.
- How to validate a FIX checksum from a FIX message before conversion.
- How to populate a FIX checksum for a FIX message that is to be created.

---

## Example assumptions

The following assumptions are made regarding FIX mapping from and to FIXML examples.

The following assumptions are made when the FIX to FIXML maps are run:

- The example FIX input files are validated, for example, fix\_email\_50.inp, which is in the  
`<install_dir>/packs/financial_payments_vn.n.n.n/fix/examples/fixFMLemail/data` directory.

- The input file is validated against the corresponding FIX type trees, for example, fix\_50.mtt, which is in the `<install_dir>/packs/financial_payments_vn.n.n/n/fix/examples/fixFMLEmail/trees` directory.
- The input file FIX checksum in tag 10, is validated by calling one of the FIX checksum utility maps, for example, fxchksm2\_validate.mmc.

The following assumptions are made when the FIXML to FIX maps are run:

- The example input file is validated, for example, fixml\_email\_50.xml, which is in the `<install_dir>/packs/financial_payments_vn.n.n/n/fix/examples/fixFMLEmail/data` directory.
- The input file is validated against the FIXML type trees, for example, fixml\_50.mtt, that is in the `<install_dir>/packs/financial_payments_vn.n.n/n/fix/examples/fixFMLEmail/trees` directory.  
Note: The FIX 5.0 SP2 and FIX 5.0 SPE EP messages work directly with schemas without the need for a corresponding type tree.
- The FIXML input file message is a well-formed XML message and conforms against the FIXML schema called, for example, fixml-main-5-0.xsd, in the `<install_dir>/packs/financial_payments_vn.n.n/n/fix/examples/fixFMLE<msg>/schemas` directory.  
Where `<msg>` is any of the FIX messages listed under the FIX component examples section.
- Output file FIX checksum in tag 10, is populated by calling one of the FIX checksum utility maps, for example, fxchksm3\_compute.mmc.

## Expected results

The following results are expected when a FIX to FIXML map is run:

- The FIX input file contents are transformed to an XML file, for example, fixml\_email\_50\_out.xml, which is located in the `<install_dir>/packs/financial_payments_vn.n.n/n/fix/examples/fixFMLEmail/data` directory.
- The output XML is represented by the FIXML type tree, for example, fixml\_50.mtt, in the `<install_dir>/packs/financial_payments_vn.n.n/n/fix/examples/fixFMLEmail/trees` directory.  
Note: The FIX 5.0 SP2 and FIX 5.0 SPE EP messages work directly with schemas without the need for a corresponding type tree.
- The resulting output FIXML message is a well-formed XML message and conforms against an FIXML schema, for example, fixml-main-5-0.xsd, in the `<install_dir>/packs/financial_payments_vn.n.n/n/fix/examples/fixFMLEmail/schemas` directory.
- The output file corresponds to one valid FIX message.

## Running the examples on z/OS

This documentation describes how to run this example on z/OS.

### About this task

Be aware of the following points when these files are used on z/OS operating systems:

- When ported to z/OS, every map, schema, and file that is ported to z/OS must have a unique eight character DDNAME in the JCL.
- After you build the maps for the z/OS operating system, you must rename the \*.mvs compiled map to a unique eight character name.
- The compiled map file must be uploaded as a PDS member that contains maps. The schemas can either be uploaded as a traditional MVS data set to be referenced in a DSN option, or in a z/OS UNIX file, such as a Hierarchical File System (zFS or HFS) environment to be referenced in a PATH option. Both maps and schemas must be uploaded as binary.

Note: Throughout this documentation, `<install_dir>` is the installation location of the Pack for Financial Payments, and `n.n.n.n` is the current version of the Pack for Financial Payments.

### Procedure

1. Create a copy of the `<install_dir>/packs/financial_payments_vn.n.n/n/fix` directory to hold the z/OS modified files.
2. Review the FIX to FIXML conversion maps. The conversion maps are in the `<install_dir>/packs/financial_payments_vn.n.n/n/fix/examples/fixFMLE<msg>/maps` directory.  
Where `<msg>` is any of the FIX messages listed under the FIX component examples section.  
Resolve the input card for FIX to FIXML.

For example:

```
fxfm0001_email_fix_to_fixml from map source fixfml_50_email.mms.
```

The map uses one input card, called fix\_50. The fix\_50 card contains file email\_50.inp. This file is in the `<install_dir>/packs/financial_payments_vn.n.n/n/fix/examples/fixFMLEmail/data` directory.

The file resolves to **DD:fix#50** as the default **DD** name. This default **DD** name can be overridden through use of the execution command.

For example:

```
MAPDDS /IF1 FIX50IN /OF1 FIXML50T /TIO /AE
```

The values **FIX50IN** and **FIXML50T** are z/OS **DD** overrides for the input and output cards that are defined for the map.

3. Review the FIXML to FIX conversion maps. The conversion maps are in `<install_dir>/packs/financial_payments_vn.n.n/n/fix/examples/fixFMLE<msg>/maps` directory.  
Where `<msg>` is any of the FIX messages listed under the FIX component examples section.  
Resolve the input cards for FIXML to FIX map.

For example:

fmx0001\_email\_fixml\_to\_fix from map source fmlfix\_50\_email.mms.

The map uses one input card called fixml\_50. The fixml\_50 card contains the file fixml\_email\_50.xml. This file is in <install\_dir>/packs/financial\_payments\_vn.n.n/n/fix/examples/fixFMLEmail/data directory.

The file resolves to **DD:fixml#50** as the default **DD** name. This default **DD** name can be overridden through the execution command.

For example:

**MAPDDS /IF1 FXML50IN /OF6 FIX50OUT /TIO /AE**

The values **FXML50IN** and **FIX50OUT** are z/OS **DD** overrides for the input and output cards that are defined for the map.

Note: Running any of the FIXML to FIX example maps requires the FIX checksum utility RUN maps to be referenced within JCL.  
For example:

• **//FIXCHKSM1 DD DSN=&CHKSM1,DISP=SHR**

where

**&CHKSM1** refers to compiled map fxchksm1\_utility

• **//FIXCHKSM3 DD DSN=&CHKSM3,DISP=SHR**

where

**&CHKSM3** refers to compiled map fxchksm3\_compute

4. The example conversion maps references metadata values that specify the FIXML schemas. There are two options about how to reference the schemas within a JCL.

- [Option 1: Schemas on an MVS data set](#)
- [Option 2: Schemas on a UNIX hierarchical file system](#)

- [Option 1: Schemas on an MVS data set](#)
- [Option 2: Schemas on a UNIX hierarchical file system](#)

---

## Option 1: Schemas on an MVS data set

### About this task

The following steps describe how to use the schemas on an MVS data set.

### Procedure

1. Modify the cards with a Doc XSD element and specify a **DD** name for the primary schema.  
For example:

**Metadata=/DD:FIXML50**

An alternative is to remove the schema from the Map cards and to modify the target type tree attribute for the Doc group.

For example:

Modify the target type tree attributes for the Doc group as follows:

**Doc > Intent > Validate As > Metadata=/DD:FIXML50**

2. Build all of the maps for z/OS and rename the maps to short names for upload.

3. Copy the schemas to short names for upload.

For example:

**fixml-main-5-0.xsd = FIXML50**

4. Modify the schemas that you are going to upload and replace the import schema locations with the short names for the schemas.  
For example:

**<xsd:import namespace="http://www.w3.org/XML/1998/namespace" schemaLocation="/DD:xml#ns>**

5. Update JCL and add the DD statements for the schemas.

For example:

**//FIXML50 DD DSN\_USER.DSTX841.MAPLIB(FIXML50),DISP=SHR**

---

## Option 2: Schemas on a UNIX hierarchical file system

### About this task

The following steps describe how to use the schemas on a UNIX hierarchical file system.

## Procedure

---

1. Use the **M'filename'** execution command line option to override the metadata (XML schema or DTD) file location on the map card settings at run time.  
For example:

```
MAPPDS /IF1 INPUT /OF1 OUTPUT /O1M '/DD:FIXMLXSD' /AE /TIO
```

2. Build all maps for z/OS and rename the map names to short names for upload.
3. Modify the schemas that you are going to upload and replace the import schema locations to reference the full directory path of the schemas.  
For example:

```
<xs:include schemaLocation="/xsd/50/fixml-pretrade-5-0.xsd"/>
```

4. Update JCL and add the DD statements for the main schema.  
`//FIXML50 DD PATH=/xsd/50/fixml-main-5-0.xsd`

- [Reminders](#)
- 

## Reminders

Keep the following points in mind when you use Option 2:

- Upload maps and schemas as binary.
- Upload input data files as binary.
- The XML parser log file **DSTXXLOG** must be allocated to a data set.

For example:

```
//DSTXXLOG DD DSN=USER.DSTXXLOG.XML,DISP=(MOD,CATLG,DELETE),
// DCB=(RECFM=VB,LRECL=32500),
// UNIT=SYSDA,SPACE=(CYL,(20,1),RLSE)
```

---

## FIX checksum validation

### About this task

The IBM Sterling Transformation Extender Pack for Financial Payments ships with FIX checksum utility maps. The fxchksm2\_validate utility map is used to verify the computed FIX checksum value against the actual value that is set to the FIX message field tag 10.

Note: Throughout this documentation, `<install_dir>` is the installation location of the Pack for Financial Payments, and `n.n.n.n` is the current version of the Pack for Financial Payments.

The following steps describe how to use the FIX checksum validation example.

## Procedure

---

1. Compile the following maps under map source fix\_checksum.mms.
  - fxchksm1\_utility
  - fxchksm2\_validate

Note: For UNIX operating systems, assign a .mmc extension to the compiled maps.

2. Run the utility map fxchksm2\_validate.

- [Example scenarios](#)
  - [Example assumptions](#)
  - [Expected results](#)
- 

## Example scenarios

The example map illustrates validation of the value that is set in a FIX tag 10 checksum.

## Example assumptions

The following assumptions are made regarding the FIX checksum validation example:

Example assumptions when the fxchksm2\_validate map is run:

- The example map validates the checksum of a FIX input file, fix\_40.inp, that is in the `<install_dir>/packs/financial_payments_vn.n.n.n/fix/utilities/data` directory.  
Note: On a z/OS environment, use fix\_40\_MVS.inp located in the same directory.
- The example map accepts any FIX message as input, and uses a general purpose type tree called, fix\_checksum.mtt, that is in the `<install_dir>/packs/financial_payments_vn.n.n.n/fix/utilities/trees` directory.

- The example map returns a value of PASS or FAIL. A value of PASS means that the tag 10 value is correct, and it matches the computed FIX checksum value. If not, the map returns a FAIL value.

## Expected results

The expected results, when the fxchksm2\_validate map is run, is PASS.

## FIX checksum generation

### About this task

The IBM Sterling Transformation Extender Pack for Financial Payments ships with FIX checksum utility maps. The fxchksm3\_compute utility map is used to generate the FIX tag 10 value by applying a FIX checksum computation value against the actual FIX message, ignoring the FIX tag 10 on the FIX message. The utility map also computes the FIX message body length. It ignores the FIX tag 9 value of the actual FIX message.

Note: Throughout this documentation, *<install\_dir>* is the installation location of the Pack for Financial Payments, and *n.n.n.n* is the current version of the Pack for Financial Payments.

### Procedure

- Compile the following maps under map source fix\_checksum.mms.
  - fxchksm1\_utility
  - fxchksm3\_compute

Note: For UNIX operating systems, assign a .mmc extension to the compiled maps.

- Run the utility map fxchksm3\_compute.

- [Example scenarios](#)
- [Example assumptions](#)
- [Expected results](#)

## Example scenarios

The example map illustrates the following:

- The creation of a FIX checksum based on a FIX message.
- The creation of a FIX message length based on a FIX message.

## Example assumptions

The following assumptions are made regarding the FIX checksum generation example:

Example assumptions when the fxchksm3\_compute map is run:

- The example map returns the checksum of a FIX input file, fix\_40.inp, that is in the *<install\_dir>/packs/financial\_payments\_vn.n.n.n/fix/utilities/data* directory.  
Note: On the z/OS operating system, use fix\_40\_MVS.inp located in the same directory.
- The example map accepts any FIX message as input, and uses a general-purpose type tree called, fix\_checksum.mtt, that is in the *<install\_dir>/packs/financial\_payments\_vn.n.n.n/fix/utilities/trees* directory.
- The example map also returns the computed message body length of a FIX message.

## Expected results

The expected results when the fxchksm3\_compute map is run:

- The expected checksum result is 34 on the first output card.
- The expected message body length is 335.

Note: For the MVS environment, the expected checksum is 207. The expected message body length is 335 on the second output card. The optional message length is configured to use the **SINK** adapter, but can be overridden, that is, changed to a **FILE** adapter if a FIX message body length is needed.

## Example for Advertisement message

This example demonstrates the FIX Advertisement to FIXML translation and FIXML to FIX translation using maps only.

Advertisement messages are used to announce completed transactions.

- [What the example contains](#)

This example contains the following directories and files:

- [How to run the example in Design Studio](#)

- [How to run the example in Design Server](#)

---

## What the example contains

This example contains the following directories and files:

### Maps

The maps directory contains the following map sources:

- fixfml\_<FIX\_version>\_adv.mms - Map to transform FIX format into FIXML format.
- fmlfix\_<FIX\_version>\_adv.mms - Map to transform FIXML format into FIX format.

Note: <FIX\_version> = '50' or '50\_sp1' or '50\_sp2'

### Schemas

The schemas directory contains the following main files:

- fixml-main-5-0.xsd
- fixml-main-5-0-SP1.xsd
- fixml-main-5-0-SP2.xsd

### Trees

The trees directory contains the following files:

- fix\_50.mtt - Metadata that represents the FIX version 5.0
- fix\_50\_sp1.mtt - Metadata that represents the FIX version 5.0 SP1
- fix\_50\_sp2.mtt - Metadata that represents the FIX version 5.0 SP2
- fixml\_50.mtt - Metadata that represents the FIXML compatible with FIX version 5.0
- fixml\_50\_sp1.mtt - Metadata that represents the FIXML compatible with FIX version 5.0 SP1
- fix\_checksum.mtt - Metadata that represents fields for checksum and body length

### Data

The data directory contains the following file.

Sample FIX and FIXML valid file:

- fix\_adv\_50.inp
- fix\_adv\_50\_sp1.inp
- fix\_adv\_50\_sp2.inp
- fixml\_adv\_50.xml
- fixml\_adv\_50\_sp1.xml
- fixml\_adv\_50\_sp2.xml
- fix\_adv\_50\_MVS.inp
- fix\_adv\_50\_sp1\_MVS.inp
- fix\_adv\_50\_sp2\_MVS.inp

---

## How to run the example in Design Studio

Running FIX to FIXML translation map:

1. Build all maps in the .mms files listed.
2. Replace input card 1 in example map fixfml\_<FIX\_version>\_adv.mms with desired input file.
3. Run main example map, fixfml\_<FIX\_version>\_adv.mms
4. See output file, on data folder called fixml\_adv\_<FIX\_version>.out.xml

Note: <FIX\_version> = '50' or '50\_sp1' or '50\_sp2'

Running FIXML to FIX translation map:

1. Build all maps in the .mms files listed.
2. Replace input card 1 in example map fmlfix\_<FIX\_version>\_adv.mms with desired input file.
3. Run main example map, fmlfix\_<FIX\_version>\_adv.mms
4. See output file, on data folder called fix\_adv\_<FIX\_version>.out

Note: <FIX\_version> = '50' or '50\_sp1' or '50\_sp2'

# How to run the example in Design Server

## About this task

Use the following steps to run the example from the Design Server:

## Procedure

1. Open a browser pointing to the installation of the IBM Sterling Transformation Extender.
2. Enter user and password credentials.
3. If project does not exist, import fixFMLadv.zip
4. If absent, create a package containing all Maps, Files and Flows.
5. Build package in desired server (to build all maps).
6. Open project.
7. In Maps tab, open map based on translation and version:
  - For FIX to FIXML use map - fixfml\_<FIX\_version>\_adv
  - For FIXML to FIX use map - fmlfix\_<FIX\_version>\_adv
8. Modify settings for input card 1 on design canvas to point to the desired test file.
9. Save, build and run the map.
10. Right click on the output card on canvas and select View data.

Note: <FIX\_version> = '50' or '50\_sp1' or '50\_sp2'

## Example for Allocation Instruction message

This example demonstrates the FIX AllocationInstruction to FIXML translation and FIXML to FIX translation using maps only.

The Allocation Instruction message provides the ability to specify how an order or set of orders should be subdivided amongst one or more accounts.

- [What the example contains](#)  
This example contains the following directories and files:
- [How to run the example in Design Studio](#)
- [How to run the example in Design Server](#)

## What the example contains

This example contains the following directories and files:

### Maps

The maps directory contains the following map sources:

- fixfml\_<FIX\_version>\_allocinstrtn.mms - Map to transform FIX format into FIXML format.
- fmlfix\_<FIX\_version>\_allocinstrtn.mms - Map to transform FIXML format into FIX format.

Note: <FIX\_version> = '50' or '50\_sp1' or '50\_sp2'.

### Schemas

The schemas directory contains the following main files:

- fixml-main-5-0.xsd
- fixml-main-5-0-SP1.xsd
- fixml-main-5-0-SP2.xsd

### Trees

The trees directory contains the following files:

- fix\_50.mtt - Metadata that represents the FIX version 5.0
- fix\_50\_sp1.mtt - Metadata that represents the FIX version 5.0 SP1
- fix\_50\_sp2.mtt - Metadata that represents the FIX version 5.0 SP2
- fixml\_50.mtt - Metadata that represents the FIXML compatible with FIX version 5.0
- fixml\_50\_sp1.mtt - Metadata that represents the FIXML compatible with FIX version 5.0 SP1
- fix\_checksum.mtt - Metadata that represents fields for checksum and body length

### Data

The data directory contains the following file:

Sample FIX and FIXML valid files:

- fix\_allocinstrctn\_50.inp
- fix\_allocinstrctn\_50\_sp1.inp
- fix\_allocinstrctn\_50\_sp2.inp
- fixml\_allocinstrctn\_50.xml
- fixml\_allocinstrctn\_50\_sp1.xml
- fixml\_allocinstrctn\_50\_sp2.xml
- fix\_allocinstrctn\_50\_MVS.inp
- fix\_allocinstrctn\_50\_sp1\_MVS.inp
- fix\_allocinstrctn\_50\_sp2\_MVS.inp

## How to run the example in Design Studio

Running FIX to FIXML translation map:

1. Build all maps in the .mms files.
  2. Replace input card 1 in example map fixfml\_<FIX\_version>\_allocinstrctn.mms with desired input file.
  3. Run main example map, fixfml\_<FIX\_version>\_allocinstrctn.mms
  4. See output file, on data folder called fixml\_allocinstrctn\_<FIX\_version>\_out.xml
- Note: <FIX\_version> = '50' or '50\_sp1' or '50\_sp2'.

Running FIXML to FIX translation map:

1. Build all maps in the .mms files.
  2. Replace input card 1 in example map fmlfix\_<FIX\_version>\_allocinstrctn.mms with desired input file.
  3. Run main example map, fmlfix\_<FIX\_version>\_allocinstrctn.mms
  4. See output file, on data folder called fix\_allocinstrctn\_<FIX\_version>.out
- Note: <FIX\_version> = '50' or '50\_sp1' or '50\_sp2'

## How to run the example in Design Server

### About this task

Use the following steps to run the example from the Design Server:

### Procedure

1. Open a browser pointing to the installation of the IBM Sterling Transformation Extender.
  2. Enter user and password credentials.
  3. If project does not exist, import fixFMLallocinstrctn.zip
  4. If absent, create a package containing all Maps, Files and Flows.
  5. Build package in desired server (to build all maps).
  6. Open project.
  7. In Maps tab, open map based on translation and version:
    - For FIX to FIXML use map - fixfml\_<FIX\_version>\_allocinstrctn
    - For FIXML to FIX use map - fmlfix\_<FIX\_version>\_allocinstrctn
  8. Modify settings for input card 1 on design canvas to point to the desired test file.
  9. Save, build and run the map.
  10. Right click on the output card on canvas and select View data.
- Note: <FIX\_version> = '50' or '50\_sp1' or '50\_sp2'

## Example for Allocation Instruction Ack message

This example demonstrates the FIX AllocationInstructionAck to FIXML translation and FIXML to FIX translation using maps only.

The Allocation Instruction Ack message is used to acknowledge the receipt of and provide status for an Allocation Instruction message.

- [What the example contains](#)  
This example contains the following directories and files:
- [How to run the example in Design Studio](#)
- [How to run the example in Design Server](#)

### What the example contains

This example contains the following directories and files:

### Maps

The maps directory contains the following map sources:

- fixfml\_<FIX\_version>\_allocinstrcnack.mms - Map to transform FIX format into FIXML format.
- fmlfix\_<FIX\_version>\_allocinstrcnack.mms - Map to transform FIXML format into FIX format.

Note: <FIX\_version> = '50' or '50\_sp1' or '50\_sp2'

## Schemas

---

The schemas directory contains the following main files:

- fixml-main-5-0.xsd
- fixml-main-5-0-SP1.xsd
- fixml-main-5-0-SP2.xsd

## Trees

---

The trees directory contains the following files:

- fix\_50.mtt - Metadata that represents the FIX version 5.0
- fix\_50\_sp1.mtt - Metadata that represents the FIX version 5.0 SP1
- fix\_50\_sp2.mtt - Metadata that represents the FIX version 5.0 SP2
- fixml\_50.mtt - Metadata that represents the FIXML compatible with FIX version 5.0
- fixml\_50\_sp1.mtt - Metadata that represents the FIXML compatible with FIX version 5.0 SP1
- fix\_checksum.mtt - Metadata that represents fields for checksum and body length

## Data

---

The data directory contains the following file.

Sample FIX and FIXML valid file:

- fix\_allocinstrcnack\_50.inp
- fix\_allocinstrcnack\_50\_sp1.inp
- fix\_allocinstrcnack\_50\_sp2.inp
- fixml\_allocinstrcnack\_50.xml
- fixml\_allocinstrcnack\_50\_sp1.xml
- fixml\_allocinstrcnack\_50\_sp2.xml
- fix\_allocinstrcnack\_50\_MVS.inp
- fix\_allocinstrcnack\_50\_sp1\_MVS.inp
- fix\_allocinstrcnack\_50\_sp2\_MVS.inp

## How to run the example in Design Studio

---

Running FIX to FIXML translation map:

1. Build all maps in the .mms files listed.
2. Replace input card 1 in example map fixfml\_<FIX\_version>\_allocinstrcnack.mms with desired input file.
3. Run main example map, fixfml\_<FIX\_version>\_allocinstrcnack.mms
4. See output file, on data folder called fixml\_allocinstrcnack\_<FIX\_version>\_out.xml

Note: <FIX\_version> = '50' or '50\_sp1' or '50\_sp2'

Running FIXML to FIX translation map:

1. Build all maps in the .mms files listed.
2. Replace input card 1 in example map fmlfix\_<FIX\_version>\_allocinstrcnack.mms with desired input file.
3. Run main example map, fmlfix\_<FIX\_version>\_allocinstrcnack.mms
4. See output file, on data folder called fix\_allocinstrcnack\_<FIX\_version>.out

Note: <FIX\_version> = '50' or '50\_sp1' or '50\_sp2'

## How to run the example in Design Server

---

### About this task

---

Use the following steps to run the example from the Design Server:

### Procedure

---

1. Open a browser pointing to the installation of the IBM Sterling Transformation Extender.
2. Enter user and password credentials.
3. If project does not exist, import fixFMLallocinstrcnack.zip
4. If absent, create a package containing all Maps, Files and Flows.
5. Build package in desired server (to build all maps).
6. Open project.

7. In Maps tab, open map based on translation and version:
  - For FIX to FIXML use map - fixfml\_<FIX\_version>\_allocinstrcnack
  - For FIXML to FIX use map - fmlfix\_<FIX\_version>\_allocinstrcnack
8. Modify settings for input card 1 on design canvas to point to the desired test file.
9. Save, build and run the map.
10. Right click on the output card on canvas and select View data.  
Note: <FIX\_version> = '50' or '50\_sp1' or '50\_sp2'

---

## Example for Don't Know Trade (DK) message

This example demonstrates the FIX DontKnowTrade to FIXML translation and FIXML to FIX translation using maps only.

The Don't Know Trade (DK) message notifies a trading partner that an electronically received execution has been rejected.

- [What the example contains](#)  
This example contains the following directories and files:
- [How to run the example in Design Studio](#)
- [How to run the example in Design Server](#)

---

## What the example contains

This example contains the following directories and files:

### Maps

The maps directory contains the following map sources:

- fixfml\_<FIX\_version>\_dktrd.mms - Map to transform FIX format into FIXML format.
- fmlfix\_<FIX\_version>\_dktrd.mms - Map to transform FIXML format into FIX format.

Note: <FIX\_version> = '50' or '50\_sp1' or '50\_sp2'

### Schemas

The schemas directory contains the following main files:

- fixml-main-5-0.xsd
- fixml-main-5-0-SP1.xsd
- fixml-main-5-0-SP2.xsd

### Trees

The trees directory contains the following files:

- fix\_50.mtt - Metadata that represents the FIX version 5.0
- fix\_50\_sp1.mtt - Metadata that represents the FIX version 5.0 SP1
- fix\_50\_sp2.mtt - Metadata that represents the FIX version 5.0 SP2
- fixml\_50.mtt - Metadata that represents the FIXML compatible with FIX version 5.0
- fixml\_50\_sp1.mtt - Metadata that represents the FIXML compatible with FIX version 5.0 SP1
- fix\_checksum.mtt - Metadata that represents fields for checksum and body length

### Data

The data directory contains the following file.

Sample FIX and FIXML valid file:

- fix\_dktrd\_50.inp
- fix\_dktrd\_50\_sp1.inp
- fix\_dktrd\_50\_sp2.inp
- fixml\_dktrd\_50.xml
- fixml\_dktrd\_50\_sp1.xml
- fixml\_dktrd\_50\_sp2.xml
- fix\_dktrd\_50\_MVS.inp
- fix\_dktrd\_50\_sp1\_MVS.inp
- fix\_dktrd\_50\_sp2\_MVS.inp

---

## How to run the example in Design Studio

Running FIX to FIXML translation map:

1. Build all maps in the .mms files listed.
2. Replace input card 1 in example map fixfml\_<FIX\_version>\_dktrd.mms with desired input file.
3. Run main example map, fixfml\_<FIX\_version>\_dktrd.mms
4. See output file, on data folder called fixxml\_dktrd\_<FIX\_version>\_out.xml

Note: <FIX\_version> = '50' or '50\_sp1' or '50\_sp2'

Running FIXML to FIX translation map:

1. Build all maps in the .mms files listed.
2. Replace input card 1 in example map fmlfix\_<FIX\_version>\_dktrd.mms with desired input file.
3. Run main example map, fmlfix\_<FIX\_version>\_dktrd.mms
4. See output file, on data folder called fix\_dktrd\_<FIX\_version>.out

Note: <FIX\_version> = '50' or '50\_sp1' or '50\_sp2'

## How to run the example in Design Server

### About this task

Use the following steps to run the example from the Design Server:

### Procedure

1. Open a browser pointing to the installation of the IBM Sterling Transformation Extender.
2. Enter user and password credentials.
3. If project does not exist, import fixFMLdktrd.zip
4. If absent, create a package containing all Maps, Files and Flows.
5. Build package in desired server (to build all maps).
6. Open project.
7. In Maps tab, open map based on translation and version:
  - For FIX to FIXML use map - fixfml\_<FIX\_version>\_dktrd
  - For FIXML to FIX use map - fmlfix\_<FIX\_version>\_dktrd
8. Modify settings for input card 1 on design canvas to point to the desired test file.
9. Save, build and run the map.
10. Right click on the output card on canvas and select View data.

Note: <FIX\_version> = '50' or '50\_sp1' or '50\_sp2'

## Example for Email message

This example demonstrates the FIX Email to FIXML translation and FIXML to FIX translation using maps only.

The email message is similar to the format and purpose of the News message, however, it is intended for private use between two parties.

- [What the example contains](#)  
This example contains the following directories and files:
- [How to run the example in Design Studio](#)
- [How to run the example in Design Server](#)

## What the example contains

This example contains the following directories and files:

### Maps

The maps directory contains the following map sources:

- fixfml\_<FIX\_version>\_email.mms - Map to transform FIX format into FIXML format.
- fmlfix\_<FIX\_version>\_email.mms - Map to transform FIXML format into FIX format.

Note: <FIX\_version> = '50' or '50\_sp1' or '50\_sp2' or '50\_sp2\_ep'

### Schemas

The schemas directory contains the following main files:

- fixml-main-5-0.xsd
- fixml-main-5-0-SP1.xsd
- fixml-main-5-0-SP2.xsd
- fixml-main-Latest.xsd

## Trees

---

The trees directory contains the following files:

- fix\_50.mtt - Metadata that represents the FIX version 5.0
- fix\_50\_sp1.mtt - Metadata that represents the FIX version 5.0 SP1
- fix\_50\_sp2.mtt - Metadata that represents the FIX version 5.0 SP2
- fix\_50\_sp2\_ep.mtt - metadata that represents the FIX version 5.0 SP2 EP / Latest EP
- fixml\_50.mtt - Metadata that represents the FIXML compatible with FIX version 5.0
- fixml\_50\_sp1.mtt - Metadata that represents the FIXML compatible with FIX version 5.0 SP1
- fixml-main-Latest.mtt - metadata that represents the FIXML compatible with FIX version 5.0 SP2 EP / Latest EP
- fix\_checksum.mtt - Metadata that represents fields for checksum and body length

## Data

---

The data directory contains the following file.

Sample FIX and FIXML valid file:

- fix\_email\_50.inp
- fix\_email\_50\_sp1.inp
- fix\_email\_50\_sp2.inp
- fix\_email\_50\_sp2\_ep.inp
- fixml\_email\_50.xml
- fixml\_email\_50\_sp1.xml
- fixml\_email\_50\_sp2.xml
- fixml\_email\_50\_sp2\_ep.xml
- fix\_email\_50\_MVS.inp
- fix\_email\_50\_sp1\_MVS.inp
- fix\_email\_50\_sp2\_MVS.inp
- fix\_email\_50\_sp2\_ep\_MVS.inp

## How to run the example in Design Studio

---

Running FIX to FIXML translation map:

1. Build all maps in the .mms files listed.
2. Replace input card 1 in example map fixfml\_<FIX\_version>\_email.mms with desired input file.
3. Run main example map, fixfml\_<FIX\_version>\_email.mms
4. See output file, on data folder called fixml\_email\_<FIX\_version>\_out.xml

Note: <FIX\_version> = '50' or '50\_sp1' or '50\_sp2' or '50\_sp2\_ep'

Running FIXML to FIX translation map:

1. Build all maps in the .mms files listed.
2. Replace input card 1 in example map fmlfix\_<FIX\_version>\_email.mms with desired input file.
3. Run main example map, fmlfix\_<FIX\_version>\_email.mms
4. See output file, on data folder called fix\_email\_<FIX\_version>.out

Note: <FIX\_version> = '50' or '50\_sp1' or '50\_sp2' or '50\_sp2\_ep'

## How to run the example in Design Server

---

### About this task

---

Use the following steps to run the example from the Design Server:

### Procedure

---

1. Open a browser pointing to the installation of the IBM Sterling Transformation Extender.
2. Enter user and password credentials.
3. If project does not exist, import fixFMLEmail.zip
4. If absent, create a package containing all Maps, Files and Flows.
5. Build package in desired server (to build all maps).
6. Open project.
7. In Maps tab, open map based on translation and version:
  - For FIX to FIXML use map - fixfml\_<FIX\_version>\_email
  - For FIXML to FIX use map - fmlfix\_<FIX\_version>\_email
8. Modify settings for input card 1 on design canvas to point to the desired test file.
9. Save, build and run the map.
10. Right click on the output card on canvas and select View data.  
Note: <FIX\_version> = '50' or '50\_sp1' or '50\_sp2' or '50\_sp2\_ep'

## Example for Execution Report message

This example demonstrates the FIX ExecutionReport to FIXML translation and FIXML to FIX translation using maps only.

The execution report message is used to:

- Confirm the receipt of an order
  - Confirm changes to an existing order (accept, cancel, and replace requests)
  - Relay order status information
  - Relay fill information on working orders
  - Relay fill information on tradeable or restricted tradeable quotes
  - Reject orders
  - Report post-trade fees calculations associated with a trade
- [What the example contains](#)  
This example contains the following directories and files:
- [How to run the example in Design Studio](#)
  - [How to run the example in Design Server](#)

## What the example contains

This example contains the following directories and files:

### Maps

The maps directory contains the following map sources:

- fixml\_<FIX\_version>\_execrpt.mms - Map to transform FIX format into FIXML format.
- fmlfix\_<FIX\_version>\_execrpt.mms - Map to transform FIXML format into FIX format.

Note: <FIX\_version> = '50' or '50\_sp1' or '50\_sp2'

### Schemas

The schemas directory contains the following main files:

- fixml-main-5-0.xsd
- fixml-main-5-0-SP1.xsd
- fixml-main-5-0-SP2.xsd

### Trees

The trees directory contains the following files:

- fix\_50.mtt - Metadata that represents the FIX version 5.0
- fix\_50\_sp1.mtt - Metadata that represents the FIX version 5.0 SP1
- fix\_50\_sp2.mtt - Metadata that represents the FIX version 5.0 SP2
- fixml\_50.mtt - Metadata that represents the FIXML compatible with FIX version 5.0
- fixml\_50\_sp1.mtt - Metadata that represents the FIXML compatible with FIX version 5.0 SP1
- fix\_checksum.mtt - Metadata that represents fields for checksum and body length

### Data

The data directory contains the following file.

Sample FIX and FIXML valid file:

- fix\_execrpt\_50.inp
- fix\_execrpt\_50\_sp1.inp
- fix\_execrpt\_50\_sp2.inp
- fixml\_execrpt\_50.xml
- fixml\_execrpt\_50\_sp1.xml
- fixml\_execrpt\_50\_sp2.xml
- fix\_execrpt\_50\_MVS.inp
- fix\_execrpt\_50\_sp1\_MVS.inp
- fix\_execrpt\_50\_sp2\_MVS.inp

## How to run the example in Design Studio

Running FIX to FIXML translation map:

1. Build all maps in the .mms files listed.

2. Replace input card 1 in example map fixfml\_<FIX\_version>\_execrpt.mms with desired input file.
3. Run main example map, fixfml\_<FIX\_version>\_execrpt.mms
4. See output file, on data folder called fixxml\_execrpt\_<FIX\_version>.out.xml

Note: <FIX\_version> = '50' or '50\_sp1' or '50\_sp2'  
Running FIXML to FIX translation map:

1. Build all maps in the .mms files listed.
2. Replace input card 1 in example map fmlfix\_<FIX\_version>\_execrpt.mms with desired input file.
3. Run main example map, fmlfix\_<FIX\_version>\_execrpt.mms
4. See output file, on data folder called fix\_execrpt\_<FIX\_version>.out

Note: <FIX\_version> = '50' or '50\_sp1' or '50\_sp2'

---

## How to run the example in Design Server

### About this task

Use the following steps to run the example from the Design Server:

### Procedure

1. Open a browser pointing to the installation of the IBM Sterling Transformation Extender.
2. Enter user and password credentials.
3. If project does not exist, import fixFMLExecrpt.zip
4. If absent, create a package containing all Maps, Files and Flows.
5. Build package in desired server (to build all maps).
6. Open project.
7. In Maps tab, open map based on translation and version:
  - For FIX to FIXML use map - fixfml\_<FIX\_version>\_execrpt
  - For FIXML to FIX use map - fmlfix\_<FIX\_version>\_execrpt
8. Modify settings for input card 1 on design canvas to point to the desired test file.
9. Save, build and run the map.
10. Right click on the output card on canvas and select View data.

Note: <FIX\_version> = '50' or '50\_sp1' or '50\_sp2'

---

## Example for Indication of interest message

This example demonstrates the FIX IOI to FIXML translation and FIXML to FIX translation using maps only.

Indication of interest messages are used to market merchandise which the broker is buying or selling in either a proprietary or agency capacity.

- [What the example contains](#)  
This example contains the following directories and files:
- [How to run the example in Design Studio](#)
- [How to run the example in Design Server](#)

---

## What the example contains

This example contains the following directories and files:

### Maps

The maps directory contains the following map sources:

- fixfml\_<FIX\_version>\_ioi.mms - Map to transform FIX format into FIXML format.
- fmlfix\_<FIX\_version>\_ioi.mms - Map to transform FIXML format into FIX format.

Note: <FIX\_version> = '50' or '50\_sp1' or '50\_sp2'

### Schemas

The schemas directory contains the following main files:

- fixml-main-5-0.xsd
- fixml-main-5-0-SP1.xsd
- fixml-main-5-0-SP2.xsd

### Trees

The trees directory contains the following files:

- fix\_50.mtt - Metadata that represents the FIX version 5.0
- fix\_50\_sp1.mtt - Metadata that represents the FIX version 5.0 SP1
- fix\_50\_sp2.mtt - Metadata that represents the FIX version 5.0 SP2
- fixml\_50.mtt - Metadata that represents the FIXML compatible with FIX version 5.0
- fixml\_50\_sp1.mtt - Metadata that represents the FIXML compatible with FIX version 5.0 SP1
- fix\_checksum.mtt - Metadata that represents fields for checksum and body length

## Data

---

The data directory contains the following file.

Sample FIX and FIXML valid file:

- fix\_ioi\_50.inp
- fix\_ioi\_50\_sp1.inp
- fix\_ioi\_50\_sp2.inp
- fixml\_ioi\_50.xml
- fixml\_ioi\_50\_sp1.xml
- fixml\_ioi\_50\_sp2.xml
- fix\_ioi\_50\_MVS.inp
- fix\_ioi\_50\_sp1\_MVS.inp
- fix\_ioi\_50\_sp2\_MVS.inp

## How to run the example in Design Studio

Running FIX to FIXML translation map:

1. Build all maps in the .mms files listed.
2. Replace input card 1 in example map fixfml\_<FIX\_version>\_ioi.mms with desired input file.
3. Run main example map, fixfml\_<FIX\_version>\_ioi.mms
4. See output file, on data folder called fixml\_ioi\_<FIX\_version>\_out.xml

Note: <FIX\_version> = '50' or '50\_sp1' or '50\_sp2'

Running FIXML to FIX translation map:

1. Build all maps in the .mms files listed.
2. Replace input card 1 in example map fmlfix\_<FIX\_version>\_ioi.mms with desired input file.
3. Run main example map, fmlfix\_<FIX\_version>\_ioi.mms
4. See output file, on data folder called fix\_ioi\_<FIX\_version>.out

Note: <FIX\_version> = '50' or '50\_sp1' or '50\_sp2'

## How to run the example in Design Server

### About this task

---

Use the following steps to run the example from the Design Server:

### Procedure

---

1. Open a browser pointing to the installation of the IBM Sterling Transformation Extender.
2. Enter user and password credentials.
3. If project does not exist, import fixFMLioi.zip
4. If absent, create a package containing all Maps, Files and Flows.
5. Build package in desired server (to build all maps).
6. Open project.
7. In Maps tab, open map based on translation and version:
  - For FIX to FIXML use map - fixfml\_<FIX\_version>\_ioi
  - For FIXML to FIX use map - fmlfix\_<FIX\_version>\_ioi
8. Modify settings for input card 1 on design canvas to point to the desired test file.
9. Save, build and run the map.
10. Right click on the output card on canvas and select View data.  
Note: <FIX\_version> = '50' or '50\_sp1' or '50\_sp2'

## Example for List Cancel Request message

---

This example demonstrates the FIX ListCancelRequest to FIXML translation and FIXML to FIX translation using maps only.

The List Cancel Request message type is used by institutions wishing to cancel previously submitted lists either before or during execution.

- [What the example contains](#)

This example contains the following directories and files:

- [How to run the example in Design Studio](#)
- [How to run the example in Design Server](#)

---

## What the example contains

This example contains the following directories and files:

### Maps

---

The maps directory contains the following map sources:

- fixfml\_<FIX\_version>\_listcxlreq.mms - Map to transform FIX format into FIXML format.
- fmlfix\_<FIX\_version>\_listcxlreq.mms - Map to transform FIXML format into FIX format.

Note: <FIX\_version> = '50' or '50\_sp1' or '50\_sp2'

### Schemas

---

The schemas directory contains the following main files:

- fixml-main-5-0.xsd
- fixml-main-5-0-SP1.xsd
- fixml-main-5-0-SP2.xsd

### Trees

---

The trees directory contains the following files:

- fix\_50.mtt - Metadata that represents the FIX version 5.0
- fix\_50\_sp1.mtt - Metadata that represents the FIX version 5.0 SP1
- fix\_50\_sp2.mtt - Metadata that represents the FIX version 5.0 SP2
- fixml\_50.mtt - Metadata that represents the FIXML compatible with FIX version 5.0
- fixml\_50\_sp1.mtt - Metadata that represents the FIXML compatible with FIX version 5.0 SP1
- fix\_checksum.mtt - Metadata that represents fields for checksum and body length

### Data

---

The data directory contains the following file.

Sample FIX and FIXML valid file:

- fix\_listcxlreq\_50.inp
- fix\_listcxlreq\_50\_sp1.inp
- fix\_listcxlreq\_50\_sp2.inp
- fixml\_listcxlreq\_50.xml
- fixml\_listcxlreq\_50\_sp1.xml
- fixml\_listcxlreq\_50\_sp2.xml
- fix\_listcxlreq\_50\_MVS.inp
- fix\_listcxlreq\_50\_sp1\_MVS.inp
- fix\_listcxlreq\_50\_sp2\_MVS.inp

---

## How to run the example in Design Studio

Running FIX to FIXML translation map:

1. Build all maps in the .mms files listed.
2. Replace input card 1 in example map fixfml\_<FIX\_version>\_listcxlreq.mms with desired input file.
3. Run main example map, fixfml\_<FIX\_version>\_listcxlreq.mms
4. See output file, on data folder called fixml\_listcxlreq\_<FIX\_version>.out.xml

Note: <FIX\_version> = '50' or '50\_sp1' or '50\_sp2'

Running FIXML to FIX translation map:

1. Build all maps in the .mms files listed.
2. Replace input card 1 in example map fmlfix\_<FIX\_version>\_listcxlreq.mms with desired input file.
3. Run main example map, fmlfix\_<FIX\_version>\_listcxlreq.mms
4. See output file, on data folder called fix\_listcxlreq\_<FIX\_version>.out

Note: <FIX\_version> = '50' or '50\_sp1' or '50\_sp2'

---

## How to run the example in Design Server

## About this task

---

Use the following steps to run the example from the Design Server:

## Procedure

---

1. Open a browser pointing to the installation of the IBM Sterling Transformation Extender.
2. Enter user and password credentials.
3. If project does not exist, import fixFMLlistcxlreq.zip
4. If absent, create a package containing all Maps, Files and Flows.
5. Build package in desired server (to build all maps).
6. Open project.
7. In Maps tab, open map based on translation and version:
  - For FIX to FIXML use map - fixfml\_<FIX\_version>\_listcxlreq
  - For FIXML to FIX use map - fmlfix\_<FIX\_version>\_listcxlreq
8. Modify settings for input card 1 on design canvas to point to the desired test file.
9. Save, build and run the map.
10. Right click on the output card on canvas and select View data.  
Note: <FIX\_version> = '50' or '50\_sp1' or '50\_sp2'

---

## Example for List Execute message

This example demonstrates the FIX ListExecute to FIXML translation and FIXML to FIX translation using maps only.

The List Execute message type is used by institutions to instruct the broker to begin execution of a previously submitted list.

- [What the example contains](#)  
This example contains the following directories and files:
- [How to run the example in Design Studio](#)
- [How to run the example in Design Server](#)

---

## What the example contains

This example contains the following directories and files:

### Maps

---

The maps directory contains the following map sources:

- fixfml\_<FIX\_version>\_listext.mms - Map to transform FIX format into FIXML format.
- fmlfix\_<FIX\_version>\_listext.mms - Map to transform FIXML format into FIX format.

Note: <FIX\_version> = '50' or '50\_sp1' or '50\_sp2'

### Schemas

---

The schemas directory contains the following main files:

- fixml-main-5-0.xsd
- fixml-main-5-0-SP1.xsd
- fixml-main-5-0-SP2.xsd

### Trees

---

The trees directory contains the following files:

- fix\_50.mtt - Metadata that represents the FIX version 5.0
- fix\_50\_sp1.mtt - Metadata that represents the FIX version 5.0 SP1
- fix\_50\_sp2.mtt - Metadata that represents the FIX version 5.0 SP2
- fixml\_50.mtt - Metadata that represents the FIXML compatible with FIX version 5.0
- fixml\_50\_sp1.mtt - Metadata that represents the FIXML compatible with FIX version 5.0 SP1
- fix\_checksum.mtt - Metadata that represents fields for checksum and body length

### Data

---

The data directory contains the following file:

Sample FIX and FIXML valid file:

- fix\_listext\_50.inp
- fix\_listext\_50\_sp1.inp
- fix\_listext\_50\_sp2.inp
- fixml\_listext\_50.xml

- fixml\_listextc\_50\_sp1.xml
- fixml\_listextc\_50\_sp2.xml
- fix\_listextc\_50\_MVS.inp
- fix\_listextc\_50\_sp1\_MVS.inp
- fix\_listextc\_50\_sp2\_MVS.inp

---

## How to run the example in Design Studio

Running FIX to FIXML translation map:

1. Build all maps in the .mms files listed.
2. Replace input card 1 in example map fixml\_<FIX\_version>\_listextc.mms with desired input file.
3. Run main example map, fixml\_<FIX\_version>\_listextc.mms
4. See output file, on data folder called fixml\_listextc\_<FIX\_version>.out.xml

Note: <FIX\_version> = '50' or '50\_sp1' or '50\_sp2'

Running FIXML to FIX translation map:

1. Build all maps in the .mms files listed.
2. Replace input card 1 in example map fmlfix\_<FIX\_version>\_listextc.mms with desired input file.
3. Run main example map, fmlfix\_<FIX\_version>\_listextc.mms
4. See output file, on data folder called fix\_listextc\_<FIX\_version>.out

Note: <FIX\_version> = '50' or '50\_sp1' or '50\_sp2'

---

## How to run the example in Design Server

### About this task

Use the following steps to run the example from the Design Server:

### Procedure

1. Open a browser pointing to the installation of the IBM Sterling Transformation Extender.
2. Enter user and password credentials.
3. If project does not exist, import fixFMLlistextc.zip
4. If absent, create a package containing all Maps, Files and Flows.
5. Build package in desired server (to build all maps).
6. Open project.
7. In Maps tab, open map based on translation and version:
  - For FIX to FIXML use map - fixml\_<FIX\_version>\_listextc
  - For FIXML to FIX use map - fmlfix\_<FIX\_version>\_listextc
8. Modify settings for input card 1 on design canvas to point to the desired test file.
9. Save, build and run the map.
10. Right click on the output card on canvas and select View data.

Note: <FIX\_version> = '50' or '50\_sp1' or '50\_sp2'

---

## Example for List Status message

This example demonstrates the FIX ListStatus to FIXML translation and FIXML to FIX translation using maps only.

The list status message is issued as the response to a List Status Request message sent in an unsolicited fashion by the sell-side.

- [What the example contains](#)  
This example contains the following directories and files:
- [How to run the example in Design Studio](#)
- [How to run the example in Design Server](#)

---

## What the example contains

This example contains the following directories and files:

### Maps

The maps directory contains the following map sources:

- fixml\_<FIX\_version>\_liststat.mms - Map to transform FIX format into FIXML format.
- fmlfix\_<FIX\_version>\_liststat.mms - Map to transform FIXML format into FIX format.

Note: <FIX\_version> = '50' or '50\_sp1' or '50\_sp2'

## Schemas

---

The schemas directory contains the following main files:

- fixml-main-5-0.xsd
- fixml-main-5-0-SP1.xsd
- fixml-main-5-0-SP2.xsd

## Trees

---

The trees directory contains the following files:

- fix\_50.mtt - Metadata that represents the FIX version 5.0
- fix\_50\_sp1.mtt - Metadata that represents the FIX version 5.0 SP1
- fix\_50\_sp2.mtt - Metadata that represents the FIX version 5.0 SP2
- fixml\_50.mtt - Metadata that represents the FIXML compatible with FIX version 5.0
- fixml\_50\_sp1.mtt - Metadata that represents the FIXML compatible with FIX version 5.0 SP1
- fix\_checksum.mtt - Metadata that represents fields for checksum and body length

## Data

---

The data directory contains the following file.

Sample FIX and FIXML valid file:

- fix\_liststat\_50.inp
- fix\_liststat\_50\_sp1.inp
- fix\_liststat\_50\_sp2.inp
- fixml\_liststat\_50.xml
- fixml\_liststat\_50\_sp1.xml
- fixml\_liststat\_50\_sp2.xml
- fix\_liststat\_50\_MVS.inp
- fix\_liststat\_50\_sp1\_MVS.inp
- fix\_liststat\_50\_sp2\_MVS.inp

## How to run the example in Design Studio

---

Running FIX to FIXML translation map:

1. Build all maps in the .mms files listed.
2. Replace input card 1 in example map fixfml\_<FIX\_version>\_liststat.mms with desired input file.
3. Run main example map, fixfml\_<FIX\_version>\_liststat.mms
4. See output file, on data folder called fixml\_liststat\_<FIX\_version>.out.xml

Note: <FIX\_version> = '50' or '50\_sp1' or '50\_sp2'

Running FIXML to FIX translation map:

1. Build all maps in the .mms files listed.
2. Replace input card 1 in example map fmlfix\_<FIX\_version>\_liststat.mms with desired input file.
3. Run main example map, fmlfix\_<FIX\_version>\_liststat.mms
4. See output file, on data folder called fix\_liststat\_<FIX\_version>.out

Note: <FIX\_version> = '50' or '50\_sp1' or '50\_sp2'

## How to run the example in Design Server

---

### About this task

---

Use the following steps to run the example from the Design Server:

### Procedure

---

1. Open a browser pointing to the installation of the IBM Sterling Transformation Extender.
2. Enter user and password credentials.
3. If project does not exist, import fixFMLliststat.zip
4. If absent, create a package containing all Maps, Files and Flows.
5. Build package in desired server (to build all maps).
6. Open project.
7. In Maps tab, open map based on translation and version:
  - For FIX to FIXML use map - fixfml\_<FIX\_version>\_liststat
  - For FIXML to FIX use map - fmlfix\_<FIX\_version>\_liststat

8. Modify settings for input card 1 on design canvas to point to the desired test file.
9. Save, build and run the map.
10. Right click on the output card on canvas and select View data.  
Note: <FIX\_version> = '50' or '50\_sp1' or '50\_sp2'

---

## Example for List Status Request message

This example demonstrates the FIX ListStatusRequest to FIXML translation and FIXML to FIX translation using maps only.

The list status request message type is used by institutions to instruct the broker to generate status messages for a list.

- [What the example contains](#)  
This example contains the following directories and files:
- [How to run the example in Design Studio](#)
- [How to run the example in Design Server](#)

---

## What the example contains

This example contains the following directories and files:

### Maps

The maps directory contains the following map sources:

- fixfml\_<FIX\_version>\_liststatreq.mms - Map to transform FIX format into FIXML format.
- fmlfix\_<FIX\_version>\_liststatreq.mms - Map to transform FIXML format into FIX format.

Note: <FIX\_version> = '50' or '50\_sp1' or '50\_sp2'

### Schemas

The schemas directory contains the following main files:

- fixml-main-5-0.xsd
- fixml-main-5-0-SP1.xsd
- fixml-main-5-0-SP2.xsd

### Trees

The trees directory contains the following files:

- fix\_50.mtt - Metadata that represents the FIX version 5.0
- fix\_50\_sp1.mtt - Metadata that represents the FIX version 5.0 SP1
- fix\_50\_sp2.mtt - Metadata that represents the FIX version 5.0 SP2
- fixml\_50.mtt - Metadata that represents the FIXML compatible with FIX version 5.0
- fixml\_50\_sp1.mtt - Metadata that represents the FIXML compatible with FIX version 5.0 SP1
- fix\_checksum.mtt - Metadata that represents fields for checksum and body length

### Data

The data directory contains the following file.

Sample FIX and FIXML valid file:

- fix\_liststatreq\_50.inp
- fix\_liststatreq\_50\_sp1.inp
- fix\_liststatreq\_50\_sp2.inp
- fixml\_liststatreq\_50.xml
- fixml\_liststatreq\_50\_sp1.xml
- fixml\_liststatreq\_50\_sp2.xml
- fix\_liststatreq\_50\_MVS.inp
- fix\_liststatreq\_50\_sp1\_MVS.inp
- fix\_liststatreq\_50\_sp2\_MVS.inp

---

## How to run the example in Design Studio

Running FIX to FIXML translation map:

1. Build all maps in the .mms files listed.
2. Replace input card 1 in example map fixfml\_<FIX\_version>\_liststatreq.mms with desired input file.
3. Run main example map, fixfml\_<FIX\_version>\_liststatreq.mms

4. See output file, on data folder called fixml\_liststareq\_<FIX\_version>.out.xml

Note: <FIX\_version> = '50' or '50\_sp1' or '50\_sp2'

Running FIXML to FIX translation map:

1. Build all maps in the .mms files listed.
2. Replace input card 1 in example map fmlfix\_<FIX\_version>\_liststareq.mms with desired input file.
3. Run main example map, fmlfix\_<FIX\_version>\_liststareq.mms
4. See output file, on data folder called fix\_liststareq\_<FIX\_version>.out

Note: <FIX\_version> = '50' or '50\_sp1' or '50\_sp2'

## How to run the example in Design Server

### About this task

Use the following steps to run the example from the Design Server:

### Procedure

1. Open a browser pointing to the installation of the IBM Sterling Transformation Extender.
2. Enter user and password credentials.
3. If project does not exist, import fixFMLliststareq.zip
4. If absent, create a package containing all Maps, Files and Flows.
5. Build package in desired server (to build all maps).
6. Open project.
7. In Maps tab, open map based on translation and version:
  - For FIX to FIXML use map - fixfml\_<FIX\_version>\_liststareq
  - For FIXML to FIX use map - fmlfix\_<FIX\_version>\_liststareq
8. Modify settings for input card 1 on design canvas to point to the desired test file.
9. Save, build and run the map.
10. Right click on the output card on canvas and select View data.

Note: <FIX\_version> = '50' or '50\_sp1' or '50\_sp2'

## Example for New Order List message

This example demonstrates the FIX NewOrderList to FIXML translation and FIXML to FIX translation using maps only.

The NewOrderList Message can be used in one of two ways depending on which market conventions are being followed.

- [What the example contains](#)  
This example contains the following directories and files:
  - [How to run the example in Design Studio](#)
  - [How to run the example in Design Server](#)

## What the example contains

This example contains the following directories and files:

### Maps

The maps directory contains the following map sources:

- fixfml\_<FIX\_version>\_newordlist.mms - Map to transform FIX format into FIXML format.
- fmlfix\_<FIX\_version>\_newordlist.mms - Map to transform FIXML format into FIX format.

Note: <FIX\_version> = '50' or '50\_sp1' or '50\_sp2'

### Schemas

The schemas directory contains the following main files:

- fixml-main-5-0.xsd
- fixml-main-5-0-SP1.xsd
- fixml-main-5-0-SP2.xsd

### Trees

The trees directory contains the following files:

- fix\_50.mtt - Metadata that represents the FIX version 5.0

- fix\_50\_sp1.mtt - Metadata that represents the FIX version 5.0 SP1
- fix\_50\_sp2.mtt - Metadata that represents the FIX version 5.0 SP2
- fixml\_50.mtt - Metadata that represents the FIXML compatible with FIX version 5.0
- fixml\_50\_sp1.mtt - Metadata that represents the FIXML compatible with FIX version 5.0 SP1
- fix\_checksum.mtt - Metadata that represents fields for checksum and body length

## Data

---

The data directory contains the following file.

Sample FIX and FIXML valid file:

- fix\_newordlist\_50.inp
- fix\_newordlist\_50\_sp1.inp
- fix\_newordlist\_50\_sp2.inp
- fixml\_newordlist\_50.xml
- fixml\_newordlist\_50\_sp1.xml
- fixml\_newordlist\_50\_sp2.xml
- fix\_newordlist\_50\_MVS.inp
- fix\_newordlist\_50\_sp1\_MVS.inp
- fix\_newordlist\_50\_sp2\_MVS.inp

## How to run the example in Design Studio

---

Running FIX to FIXML translation map:

1. Build all maps in the .mms files listed.
2. Replace input card 1 in example map fixml\_<FIX\_version>\_newordlist.mms with desired input file.
3. Run main example map, fixml\_<FIX\_version>\_newordlist.mms
4. See output file, on data folder called fixml\_newordlist\_<FIX\_version>.out.xml

Note: <FIX\_version> = '50' or '50\_sp1' or '50\_sp2'

Running FIXML to FIX translation map:

1. Build all maps in the .mms files listed.
2. Replace input card 1 in example map fmlfix\_<FIX\_version>\_newordlist.mms with desired input file.
3. Run main example map, fmlfix\_<FIX\_version>\_newordlist.mms
4. See output file, on data folder called fix\_newordlist\_<FIX\_version>.out

Note: <FIX\_version> = '50' or '50\_sp1' or '50\_sp2'

## How to run the example in Design Server

---

### About this task

---

Use the following steps to run the example from the Design Server:

### Procedure

---

1. Open a browser pointing to the installation of the IBM Sterling Transformation Extender.
2. Enter user and password credentials.
3. If project does not exist, import fixMLnewordlist.zip
4. If absent, create a package containing all Maps, Files and Flows.
5. Build package in desired server (to build all maps).
6. Open project.
7. In Maps tab, open map based on translation and version:
  - For FIX to FIXML use map - fixml\_<FIX\_version>\_newordlist
  - For FIXML to FIX use map - fmlfix\_<FIX\_version>\_newordlist
8. Modify settings for input card 1 on design canvas to point to the desired test file.
9. Save, build and run the map.
10. Right click on the output card on canvas and select View data.

Note: <FIX\_version> = '50' or '50\_sp1' or '50\_sp2'

## Example for News message

---

This example demonstrates the FIX News to FIXML translation and FIXML to FIX translation using maps only.

The news message is a general free format message between the broker and institution.

- [What the example contains](#)

This example contains the following directories and files:

- [How to run the example in Design Studio](#)

- [How to run the example in Design Server](#)

## What the example contains

This example contains the following directories and files:

### Maps

The maps directory contains the following map sources:

- fixfml\_<FIX\_version>\_news.mms - Map to transform FIX format into FIXML format.
- fmlfix\_<FIX\_version>\_news.mms - Map to transform FIXML format into FIX format.

Note: <FIX\_version> = '50' or '50\_sp1' or '50\_sp2'

### Schemas

The schemas directory contains the following main files:

- fixml-main-5-0.xsd
- fixml-main-5-0-SP1.xsd
- fixml-main-5-0-SP2.xsd

### Trees

The trees directory contains the following files:

- fix\_50.mtt - Metadata that represents the FIX version 5.0
- fix\_50\_sp1.mtt - Metadata that represents the FIX version 5.0 SP1
- fix\_50\_sp2.mtt - Metadata that represents the FIX version 5.0 SP2
- fixml\_50.mtt - Metadata that represents the FIXML compatible with FIX version 5.0
- fixml\_50\_sp1.mtt - Metadata that represents the FIXML compatible with FIX version 5.0 SP1
- fix\_checksum.mtt - Metadata that represents fields for checksum and body length

### Data

The data directory contains the following file.

Sample FIX and FIXML valid file:

- fix\_news\_50.inp
- fix\_news\_50\_sp1.inp
- fix\_news\_50\_sp2.inp
- fixml\_news\_50.xml
- fixml\_news\_50\_sp1.xml
- fixml\_news\_50\_sp2.xml
- fix\_news\_50\_MVS.inp
- fix\_news\_50\_sp1\_MVS.inp
- fix\_news\_50\_sp2\_MVS.inp

## How to run the example in Design Studio

Running FIX to FIXML translation map:

1. Build all maps in the .mms files listed.
2. Replace input card 1 in example map fixfml\_<FIX\_version>\_news.mms with desired input file.
3. Run main example map, fixfml\_<FIX\_version>\_news.mms
4. See output file, on data folder called fixml\_news\_<FIX\_version>\_out.xml

Note: <FIX\_version> = '50' or '50\_sp1' or '50\_sp2'

Running FIXML to FIX translation map:

1. Build all maps in the .mms files listed.
2. Replace input card 1 in example map fmlfix\_<FIX\_version>\_news.mms with desired input file.
3. Run main example map, fmlfix\_<FIX\_version>\_news.mms
4. See output file, on data folder called fix\_news\_<FIX\_version>.out

Note: <FIX\_version> = '50' or '50\_sp1' or '50\_sp2'

## How to run the example in Design Server

## About this task

---

Use the following steps to run the example from the Design Server:

## Procedure

---

1. Open a browser pointing to the installation of the IBM Sterling Transformation Extender.
2. Enter user and password credentials.
3. If project does not exist, import fixFMLnews.zip
4. If absent, create a package containing all Maps, Files and Flows.
5. Build package in desired server (to build all maps).
6. Open project.
7. In Maps tab, open map based on translation and version:
  - For FIX to FIXML use map - fixfml\_<FIX\_version>\_news
  - For FIXML to FIX use map - fmlfix\_<FIX\_version>\_news
8. Modify settings for input card 1 on design canvas to point to the desired test file.
9. Save, build and run the map.
10. Right click on the output card on canvas and select View data.  
Note: <FIX\_version> = '50' or '50\_sp1' or '50\_sp2'

## Example for Order Cancel Reject message

---

This example demonstrates the FIX OrderCancelReject to FIXML translation and FIXML to FIX translation using maps only.

The order cancel reject message is issued by the broker upon receipt of a cancel request or cancel/replace request message which cannot be honored.

- [What the example contains](#)  
This example contains the following directories and files:
- [How to run the example in Design Studio](#)
- [How to run the example in Design Server](#)

## What the example contains

---

This example contains the following directories and files:

### Maps

---

The maps directory contains the following map sources:

- fixfml\_<FIX\_version>\_ordcxlrej.mms - Map to transform FIX format into FIXML format.
- fmlfix\_<FIX\_version>\_ordcxlrej.mms - Map to transform FIXML format into FIX format.

Note: <FIX\_version> = '50' or '50\_sp1' or '50\_sp2'

### Schemas

---

The schemas directory contains the following main files:

- fixml-main-5-0.xsd
- fixml-main-5-0-SP1.xsd
- fixml-main-5-0-SP2.xsd

### Trees

---

The trees directory contains the following files:

- fix\_50.mtt - Metadata that represents the FIX version 5.0
- fix\_50\_sp1.mtt - Metadata that represents the FIX version 5.0 SP1
- fix\_50\_sp2.mtt - Metadata that represents the FIX version 5.0 SP2
- fixml\_50.mtt - Metadata that represents the FIXML compatible with FIX version 5.0
- fixml\_50\_sp1.mtt - Metadata that represents the FIXML compatible with FIX version 5.0 SP1
- fix\_checksum.mtt - Metadata that represents fields for checksum and body length

### Data

---

The data directory contains the following file:

Sample FIX and FIXML valid file:

- fix\_ordcxlrej\_50.inp
- fix\_ordcxlrej\_50\_sp1.inp
- fix\_ordcxlrej\_50\_sp2.inp
- fixml\_ordcxlrej\_50.xml

- fixml\_ordcxlrej\_50\_sp1.xml
- fixml\_ordcxlrej\_50\_sp2.xml
- fix\_ordcxlrej\_50\_MVS.inp
- fix\_ordcxlrej\_50\_sp1\_MVS.inp
- fix\_ordcxlrej\_50\_sp2\_MVS.inp

---

## How to run the example in Design Studio

Running FIX to FIXML translation map:

1. Build all maps in the .mms files listed.
2. Replace input card 1 in example map fixml\_<FIX\_version>\_ordcxlrej.mms with desired input file.
3. Run main example map, fixml\_<FIX\_version>\_ordcxlrej.mms
4. See output file, on data folder called fixml\_<FIX\_version>\_out.xml

Note: <FIX\_version> = '50' or '50\_sp1' or '50\_sp2'

Running FIXML to FIX translation map:

1. Build all maps in the .mms files listed.
2. Replace input card 1 in example map fmlfix\_<FIX\_version>\_ordcxlrej.mms with desired input file.
3. Run main example map, fmlfix\_<FIX\_version>\_ordcxlrej.mms
4. See output file, on data folder called fix\_ordcxlrej\_<FIX\_version>.out

Note: <FIX\_version> = '50' or '50\_sp1' or '50\_sp2'

---

## How to run the example in Design Server

### About this task

Use the following steps to run the example from the Design Server:

### Procedure

1. Open a browser pointing to the installation of the IBM Sterling Transformation Extender.
2. Enter user and password credentials.
3. If project does not exist, import fixFMLordcxlrej.zip
4. If absent, create a package containing all Maps, Files and Flows.
5. Build package in desired server (to build all maps).
6. Open project.
7. In Maps tab, open map based on translation and version:
  - For FIX to FIXML use map - fixml\_<FIX\_version>\_ordcxlrej
  - For FIXML to FIX use map - fmlfix\_<FIX\_version>\_ordcxlrej
8. Modify settings for input card 1 on design canvas to point to the desired test file.
9. Save, build and run the map.
10. Right click on the output card on canvas and select View data.

Note: <FIX\_version> = '50' or '50\_sp1' or '50\_sp2'

---

## Example for New Order message

This example demonstrates the FIX NewOrderSingle to FIXML translation and FIXML to FIX translation using maps only.

The new order message type is used by institutions wishing to electronically submit securities and forex orders to a broker for execution.

- [What the example contains](#)  
This example contains the following directories and files:
- [How to run the example in Design Studio](#)
- [How to run the example in Design Server](#)

---

## What the example contains

This example contains the following directories and files:

### Maps

The maps directory contains the following map sources:

- fixml\_<FIX\_version>\_order.mms - Map to transform FIX format into FIXML format.
- fmlfix\_<FIX\_version>\_order.mms - Map to transform FIXML format into FIX format.

Note: <FIX\_version> = '50' or '50\_sp1' or '50\_sp2'

## Schemas

---

The schemas directory contains the following main files:

- fixml-main-5-0.xsd
- fixml-main-5-0-SP1.xsd
- fixml-main-5-0-SP2.xsd

## Trees

---

The trees directory contains the following files:

- fix\_50.mtt - Metadata that represents the FIX version 5.0
- fix\_50\_sp1.mtt - Metadata that represents the FIX version 5.0 SP1
- fix\_50\_sp2.mtt - Metadata that represents the FIX version 5.0 SP2
- fixml\_50.mtt - Metadata that represents the FIXML compatible with FIX version 5.0
- fixml\_50\_sp1.mtt - Metadata that represents the FIXML compatible with FIX version 5.0 SP1
- fix\_checksum.mtt - Metadata that represents fields for checksum and body length

## Data

---

The data directory contains the following file.

Sample FIX and FIXML valid file:

- fix\_order\_50.inp
- fix\_order\_50\_sp1.inp
- fix\_order\_50\_sp2.inp
- fixml\_order\_50.xml
- fixml\_order\_50\_sp1.xml
- fixml\_order\_50\_sp2.xml
- fix\_order\_50\_MVS.inp
- fix\_order\_50\_sp1\_MVS.inp
- fix\_order\_50\_sp2\_MVS.inp

## How to run the example in Design Studio

---

Running FIX to FIXML translation map:

1. Build all maps in the .mms files listed.
2. Replace input card 1 in example map fixml\_<FIX\_version>\_order.mms with desired input file.
3. Run main example map, fixml\_<FIX\_version>\_order.mms
4. See output file, on data folder called fixml\_order\_<FIX\_version>\_out.xml

Note: <FIX\_version> = '50' or '50\_sp1' or '50\_sp2'

Running FIXML to FIX translation map:

1. Build all maps in the .mms files listed.
2. Replace input card 1 in example map fmlfix\_<FIX\_version>\_order.mms with desired input file.
3. Run main example map, fmlfix\_<FIX\_version>\_order.mms
4. See output file, on data folder called fix\_order\_<FIX\_version>.out

Note: <FIX\_version> = '50' or '50\_sp1' or '50\_sp2'

## How to run the example in Design Server

---

### About this task

---

Use the following steps to run the example from the Design Server:

### Procedure

---

1. Open a browser pointing to the installation of the IBM Sterling Transformation Extender.
2. Enter user and password credentials.
3. If project does not exist, import fixFMLorder.zip
4. If absent, create a package containing all Maps, Files and Flows.
5. Build package in desired server (to build all maps).
6. Open project.
7. In Maps tab, open map based on translation and version:
  - For FIX to FIXML use map - fixml\_<FIX\_version>\_order
  - For FIXML to FIX use map - fmlfix\_<FIX\_version>\_order

8. Modify settings for input card 1 on design canvas to point to the desired test file.
9. Save, build and run the map.
10. Right click on the output card on canvas and select View data.  
Note: <FIX\_version> = '50' or '50\_sp1' or '50\_sp2'

---

## Example for Order Status Request message

This example demonstrates the FIX OrderStatusRequest to FIXML translation and FIXML to FIX translation using maps only.

The order status request message is used by the institution to generate an order status message back from the broker.

- [What the example contains](#)

This example contains the following directories and files:

- [How to run the example in Design Studio](#)

- [How to run the example in Design Server](#)

---

## What the example contains

This example contains the following directories and files:

### Maps

The maps directory contains the following map sources:

- fixfml\_<FIX\_version>\_orderstatreq.mms - Map to transform FIX format into FIXML format.
- fmlfix\_<FIX\_version>\_orderstatreq.mms - Map to transform FIXML format into FIX format.

Note: <FIX\_version> = '50' or '50\_sp1' or '50\_sp2'

---

### Schemas

The schemas directory contains the following main files:

- fixml-main-5-0.xsd
- fixml-main-5-0-SP1.xsd
- fixml-main-5-0-SP2.xsd

---

### Trees

The trees directory contains the following files:

- fix\_50.mtt - Metadata that represents the FIX version 5.0
- fix\_50\_sp1.mtt - Metadata that represents the FIX version 5.0 SP1
- fix\_50\_sp2.mtt - Metadata that represents the FIX version 5.0 SP2
- fixml\_50.mtt - Metadata that represents the FIXML compatible with FIX version 5.0
- fixml\_50\_sp1.mtt - Metadata that represents the FIXML compatible with FIX version 5.0 SP1
- fix\_checksum.mtt - Metadata that represents fields for checksum and body length

---

### Data

The data directory contains the following file.

Sample FIX and FIXML valid file:

- fix\_orderstatreq\_50.inp
- fix\_orderstatreq\_50\_sp1.inp
- fix\_orderstatreq\_50\_sp2.inp
- fixml\_orderstatreq\_50.xml
- fixml\_orderstatreq\_50\_sp1.xml
- fixml\_orderstatreq\_50\_sp2.xml
- fix\_orderstatreq\_50\_MVS.inp
- fix\_orderstatreq\_50\_sp1\_MVS.inp
- fix\_orderstatreq\_50\_sp2\_MVS.inp

---

## How to run the example in Design Studio

Running FIX to FIXML translation map:

1. Build all maps in the .mms files listed.
2. Replace input card 1 in example map fixfml\_<FIX\_version>\_orderstatreq.mms with desired input file.
3. Run main example map, fixfml\_<FIX\_version>\_orderstatreq.mms

4. See output file, on data folder called fixml\_orderstatreq\_<FIX\_version>.out.xml

Note: <FIX\_version> = '50' or '50\_sp1' or '50\_sp2'

Running FIXML to FIX translation map:

1. Build all maps in the .mms files listed.
2. Replace input card 1 in example map fmlfix\_<FIX\_version>\_orderstatreq.mms with desired input file.
3. Run main example map, fmlfix\_<FIX\_version>\_orderstatreq.mms
4. See output file, on data folder called fix\_orderstatreq\_<FIX\_version>.out

Note: <FIX\_version> = '50' or '50\_sp1' or '50\_sp2'

## How to run the example in Design Server

### About this task

Use the following steps to run the example from the Design Server:

### Procedure

1. Open a browser pointing to the installation of the IBM Sterling Transformation Extender.
2. Enter user and password credentials.
3. If project does not exist, import fixFMLorderstatreq.zip
4. If absent, create a package containing all Maps, Files and Flows.
5. Build package in desired server (to build all maps).
6. Open project.
7. In Maps tab, open map based on translation and version:
  - For FIX to FIXML use map - fixfml\_<FIX\_version>\_orderstatreq
  - For FIXML to FIX use map - fmlfix\_<FIX\_version>\_orderstatreq
8. Modify settings for input card 1 on design canvas to point to the desired test file.
9. Save, build and run the map.
10. Right click on the output card on canvas and select View data.

Note: <FIX\_version> = '50' or '50\_sp1' or '50\_sp2'

## Example for Quote message

This example demonstrates the FIX Quote to FIXML translation and FIXML to FIX translation using maps only.

The Quote message is used as the response to a Quote Request or a Quote Response message in both indicative, tradeable, and restricted tradeable quoting markets.

- [What the example contains](#)

This example contains the following directories and files:

- [How to run the example in Design Studio](#)
- [How to run the example in Design Server](#)

## What the example contains

This example contains the following directories and files:

### Maps

The maps directory contains the following map sources:

- fixfml\_<FIX\_version>\_quot.mms - Map to transform FIX format into FIXML format.
- fmlfix\_<FIX\_version>\_quot.mms - Map to transform FIXML format into FIX format.

Note: <FIX\_version> = '50' or '50\_sp1' or '50\_sp2'

### Schemas

The schemas directory contains the following main files:

- fixml-main-5-0.xsd
- fixml-main-5-0-SP1.xsd
- fixml-main-5-0-SP2.xsd

### Trees

The trees directory contains the following files:

- fix\_50.mtt - Metadata that represents the FIX version 5.0

- fix\_50\_sp1.mtt - Metadata that represents the FIX version 5.0 SP1
- fix\_50\_sp2.mtt - Metadata that represents the FIX version 5.0 SP2
- fixml\_50.mtt - Metadata that represents the FIXML compatible with FIX version 5.0
- fixml\_50\_sp1.mtt - Metadata that represents the FIXML compatible with FIX version 5.0 SP1
- fix\_checksum.mtt - Metadata that represents fields for checksum and body length

## Data

---

The data directory contains the following file.

Sample FIX and FIXML valid file:

- fix\_quot\_50.inp
- fix\_quot\_50\_sp1.inp
- fix\_quot\_50\_sp2.inp
- fixml\_quot\_50.xml
- fixml\_quot\_50\_sp1.xml
- fixml\_quot\_50\_sp2.xml
- fix\_quot\_50\_MVS.inp
- fix\_quot\_50\_sp1\_MVS.inp
- fix\_quot\_50\_sp2\_MVS.inp

## How to run the example in Design Studio

---

Running FIX to FIXML translation map:

1. Build all maps in the .mms files listed.
2. Replace input card 1 in example map fixml\_<FIX\_version>\_quot.mms with desired input file.
3. Run main example map, fixml\_<FIX\_version>\_quot.mms
4. See output file, on data folder called fixml\_quot\_<FIX\_version>\_out.xml

Note: <FIX\_version> = '50' or '50\_sp1' or '50\_sp2'

Running FIXML to FIX translation map:

1. Build all maps in the .mms files listed.
2. Replace input card 1 in example map fmlfix\_<FIX\_version>\_quot.mms with desired input file.
3. Run main example map, fmlfix\_<FIX\_version>\_quot.mms
4. See output file, on data folder called fix\_quot\_<FIX\_version>.out

Note: <FIX\_version> = '50' or '50\_sp1' or '50\_sp2'

## How to run the example in Design Server

---

### About this task

---

Use the following steps to run the example from the Design Server:

### Procedure

---

1. Open a browser pointing to the installation of the IBM Sterling Transformation Extender.
2. Enter user and password credentials.
3. If project does not exist, import fixFMLquot.zip
4. If absent, create a package containing all Maps, Files and Flows.
5. Build package in desired server (to build all maps).
6. Open project.
7. In Maps tab, open map based on translation and version:
  - For FIX to FIXML use map - fixml\_<FIX\_version>\_quot
  - For FIXML to FIX use map - fmlfix\_<FIX\_version>\_quot
8. Modify settings for input card 1 on design canvas to point to the desired test file.
9. Save, build and run the map.
10. Right click on the output card on canvas and select View data.

Note: <FIX\_version> = '50' or '50\_sp1' or '50\_sp2'

## Example for Quote Request message

---

This example demonstrates the FIX QuoteRequest to FIXML translation and FIXML to FIX translation using maps only.

In some markets it is the practice to request quotes from brokers prior to placement of an order. The quote request message is used for this purpose.

- [What the example contains](#)

This example contains the following directories and files:

- [How to run the example in Design Studio](#)

- [How to run the example in Design Server](#)
- 

## What the example contains

This example contains the following directories and files:

### Maps

---

The maps directory contains the following map sources:

- fixfml\_<FIX\_version>\_quotreq.mms - Map to transform FIX format into FIXML format.
- fmlfix\_<FIX\_version>\_quotreq.mms - Map to transform FIXML format into FIX format.

Note: <FIX\_version> = '50' or '50\_sp1' or '50\_sp2'

### Schemas

---

The schemas directory contains the following main files:

- fixml-main-5-0.xsd
- fixml-main-5-0-SP1.xsd
- fixml-main-5-0-SP2.xsd

### Trees

---

The trees directory contains the following files:

- fix\_50.mtt - Metadata that represents the FIX version 5.0
- fix\_50\_sp1.mtt - Metadata that represents the FIX version 5.0 SP1
- fix\_50\_sp2.mtt - Metadata that represents the FIX version 5.0 SP2
- fixml\_50.mtt - Metadata that represents the FIXML compatible with FIX version 5.0
- fixml\_50\_sp1.mtt - Metadata that represents the FIXML compatible with FIX version 5.0 SP1
- fix\_checksum.mtt - Metadata that represents fields for checksum and body length

### Data

---

The data directory contains the following file.

Sample FIX and FIXML valid file:

- fix\_quotreq\_50.inp
- fix\_quotreq\_50\_sp1.inp
- fix\_quotreq\_50\_sp2.inp
- fixml\_quotreq\_50.xml
- fixml\_quotreq\_50\_sp1.xml
- fixml\_quotreq\_50\_sp2.xml
- fix\_quotreq\_50\_MVS.inp
- fix\_quotreq\_50\_sp1\_MVS.inp
- fix\_quotreq\_50\_sp2\_MVS.inp

## How to run the example in Design Studio

Running FIX to FIXML translation map:

1. Build all maps in the .mms files listed.
2. Replace input card 1 in example map fixfml\_<FIX\_version>\_quotreq.mms with desired input file.
3. Run main example map, fixfml\_<FIX\_version>\_quotreq.mms
4. See output file, on data folder called fixml\_quotreq\_<FIX\_version>.out.xml

Note: <FIX\_version> = '50' or '50\_sp1' or '50\_sp2'

Running FIXML to FIX translation map:

1. Build all maps in the .mms files listed.
2. Replace input card 1 in example map fmlfix\_<FIX\_version>\_quotreq.mms with desired input file.
3. Run main example map, fmlfix\_<FIX\_version>\_quotreq.mms
4. See output file, on data folder called fix\_quotreq\_<FIX\_version>.out

Note: <FIX\_version> = '50' or '50\_sp1' or '50\_sp2'

## How to run the example in Design Server

## About this task

---

Use the following steps to run the example from the Design Server:

## Procedure

---

1. Open a browser pointing to the installation of the IBM Sterling Transformation Extender.
2. Enter user and password credentials.
3. If project does not exist, import fixFMLquotreq.zip
4. If absent, create a package containing all Maps, Files and Flows.
5. Build package in desired server (to build all maps).
6. Open project.
7. In Maps tab, open map based on translation and version:
  - For FIX to FIXML use map - fixfml\_<FIX\_version>\_quotreq
  - For FIXML to FIX use map - fmlfix\_<FIX\_version>\_quotreq
8. Modify settings for input card 1 on design canvas to point to the desired test file.
9. Save, build and run the map.
10. Right click on the output card on canvas and select View data.  
Note: <FIX\_version> = '50' or '50\_sp1' or '50\_sp2'

---

## Overview of the NACHA component

NACHA is a national, not-for-profit organization that develops operating rules and business practices for electronic payments. The Pack for Financial Payments, NACHA component provides support for the exchange of ACH payments.

### Introduction to the Pack for Financial Payments NACHA component

---

The Pack for Financial Payments NACHA component is used for validation and mapping of files that conform to the National Automated Clearing House Association (NACHA) Operating Rules. The NACHA component can be integrated into larger solutions.

The NACHA component includes a ITX type tree that complies with the ACH file. It also includes record and field level specifications and example maps to help you use the type tree.

The pack delivers extensive compliance by use of the validation rules and content restrictions included in the type tree definitions.

Maps are included for conversion on the following ACH entries:

- Corporate Credit or Debit (CCD)
- Corporate Trade Exchange (CTX)
- Death Notification Entry (DNE)
- International ACH Transaction (IAT)
- Prearranged Payments and Deposits (PPD)
- Returns
- Notification of Change
- Refused Notification of Change
- Contested Dishonored Returns

Maps are included to support the NACHA endorsed banking conventions for ACH entries with payment-related information.

It further includes examples that can be executed in the IBM Sterling B2B Integrator ACH service.

Introductory and reference documentation is included to implement the Pack for Financial Payments NACHA component as part of an overall electronics payments solution.

## The NACHA standard

---

Members of the NACHA organization define the rules that cover the Automated Clearing House (ACH) network in the United States. The ACH Network is a nationwide electronic funds transfer system that is a batch-oriented store and forward system. In this system, transactions received by financial institutions are stored and processed later in the day in a batch mode, rather than receiving and processing each payment individually.

- [\*\*ACH network terminology\*\*](#)  
Participants that are involved in electronic payments are the individuals or companies that request the payment transaction, the banks that are involved, and clearing houses. Clearing houses are central facilities through which the banks can send or receive the transaction information.
- [\*\*NACHA credit transfer and direct debit processes\*\*](#)  
This documentation describes the credit transfer and direct debit processes that are used with NACHA.
- [\*\*The ACH file\*\*](#)  
An ACH transaction flows through the portions of the processes in an ACH file. The ACH file has a format as defined in the technical specifications of the NACHA Operating Rules.

---

## ACH network terminology

Participants that are involved in electronic payments are the individuals or companies that request the payment transaction, the banks that are involved, and clearing houses. Clearing houses are central facilities through which the banks can send or receive the transaction information.

The following list introduces the specific terminology that NACHA uses to define these roles.

- *Originator*. – The company or individual that initiates the transaction. The transaction can be either credit or debit.
- *Originating Depository Financial Institution (ODFI)*. – This institution receives transaction information from the Originator and forwards the transaction information to ACH Operator.
- *ACH Operator* – Federal Reserve Bank (FRB) or private organization.
- *Receiving Depository Financial Institution (RDFI)*. – Any financial institution that is qualified to receive ACH entries that agree to abide by the NACHA Operating Rules and Guidelines. The RDFI is responsible for posting entries and for providing funds availability, which are determined by the Settlement Date in the Company Batch Header Record.
- *Receiver* – The company or individual that authorizes the originator to initiate ACH to the receiver's account.
- *Third Party Service Provider* – The Third Party Service Provider provides services on behalf of the originator, ODFI or RDFI.

## NACHA credit transfer and direct debit processes

This documentation describes the credit transfer and direct debit processes that are used with NACHA.

- **Credit processes**  
Within the credit process, the initiator is always crediting the receiver's account with funds. The credit process includes both consumer and corporate payments.
- **Debit process**  
Within the debit process, the initiator always collects funds from the receiver's account and transfers funds to the originator's account.

## Credit processes

Within the credit process, the initiator is always crediting the receiver's account with funds. The credit process includes both consumer and corporate payments.

This list shows some types (but not all types) of credits:

- Annuities
- Dividends
- Payroll - direct deposit

## Debit process

Within the debit process, the initiator always collects funds from the receiver's account and transfers funds to the originator's account.

This list shows types debits:

- Club dues
- Cash concentration
- Mortgage payments
- Utility payments

## The ACH file

An ACH transaction flows through the portions of the processes in an ACH file. The ACH file has a format as defined in the technical specifications of the NACHA Operating Rules.

This file can consist of multiple ACH entries that are defined by an application type. The NACHA rules support a number of Consumer and Corporate payment application types.

## Configuration

This documentation describes the system configuration.

- **Directory structure**  
When you install the Pack for Financial Payments NACHA component, the examples, and trees directories are created.

## Directory structure

When you install the Pack for Financial Payments NACHA component, the examples, and trees directories are created.

The directories are installed at this location:

<install\_dir>/packs/financial\_payments\_vn.n.n.n/nacha

where <install\_dir> is the installation location of the Pack for Financial Payments, and n.n.n indicates the current version of the Pack for Financial Payments.

- [examples directory](#)  
Under the examples directory are subdirectories that contain examples that show how to use the pack.
- [trees directory](#)  
The type tree that is contained in the trees directory represents the technical specifications of the ACH rules. These rules are defined in the appendixes of the *Operating Rules of the National Automated Clearing House Association*.

## examples directory

Under the examples directory are subdirectories that contain examples that show how to use the pack.

For a complete description of how to run the examples, see [NACHA component examples](#).

This table shows the directory structure of the examples.

| Directory structure                                         | Description                                                                                                 |
|-------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------|
| ACH Contested Dishonored Return Entry example               |                                                                                                             |
| ..examples/ach_cntstd_dshnrd_rtrn                           | Top level folder for the example.                                                                           |
| ..examples/ach_cntstd_dshnrd_rtrn/data                      | Placeholder for the input test data.                                                                        |
| ..examples/ach_cntstd_dshnrd_rtrn/maps                      | Placeholder for the maps.                                                                                   |
| ACH Dishonored Return Entry example                         |                                                                                                             |
| ..examples/ach_dshnrd_rtrn                                  | Top level folder for the example.                                                                           |
| ..examples/ach_dshnrd_rtrn/data                             | Placeholder for the input test data.                                                                        |
| ..examples/ach_dshnrd_rtrn/maps                             | Placeholder for the maps.                                                                                   |
| ACH conversion from/to EDI example                          |                                                                                                             |
| ..examples/ach_edi                                          | Top level folder for the example.                                                                           |
| ..examples/ach_edi/data                                     | Placeholder for the input test data.                                                                        |
| ..examples/ach_edi/maps                                     | Placeholder for the maps.                                                                                   |
| ACH mapping from and to ISO 20022 examples                  |                                                                                                             |
| ..examples/ach_iso20002                                     | Contains the ACH conversion from/to ISO 20022 examples, assumptions tables, and shared maps and type trees. |
| ..examples/ach_iso20002/data                                | Placeholder for the input test data for multiple examples.                                                  |
| ..examples/ach_iso20002/trees                               | Placeholder for the type trees used by multiple examples.                                                   |
| ..examples/ach_iso20002/scenarios                           | Contains the examples and assumptions tables.                                                               |
| ACH to ISO 20022 example assumptions tables                 |                                                                                                             |
| ..examples/ach_iso20002/scenarios/assumptions               | Contains assumptions tables used with the ACH/ISO 20022 examples                                            |
| ACH to ISO 20022 Payment Initiation Credit Transfer example |                                                                                                             |
| ..examples/ach_iso20002/scenario/credit_transfer            | Top level folder for the example.                                                                           |
| ..examples/ach_iso20002/scenario/credit_transfer/data       | Placeholder for the input test data.                                                                        |
| ..examples/ach_iso20002/scenario/credit_transfer/maps       | Placeholder for the maps.                                                                                   |
| ..examples/scenario/credit_transfer/schemas                 | Placeholder for the example schemas.                                                                        |
| ACH to ISO 20022 Payment Initiation Direct Debit example    |                                                                                                             |
| ..examples/ach_iso20002/scenario/direct_debit               | Top level folder for the example.                                                                           |
| ..examples/ach_iso20002/scenario/direct_debit/data          | Placeholder for the input test data.                                                                        |
| ..examples/ach_iso20002/scenario/direct_debit/maps          | Placeholder for the maps.                                                                                   |
| ..examples/ach_iso20002/scenario/direct_debit/schemas       | Placeholder for the example schemas.                                                                        |
| ACH Reject to ISO 20022 transformation example              |                                                                                                             |
| ..examples/ach_iso20002/scenario/reject                     | Top level folder for the example.                                                                           |
| ..examples/ach_iso20002/scenario/reject/data                | Placeholder for the input test data.                                                                        |
| ..examples/ach_iso20002/scenario/reject/maps                | Placeholder for the maps.                                                                                   |
| ..examples/ach_iso20002/scenario/reject/schemas             | Placeholder for the example schemas.                                                                        |
| ACH Returns to ISO 20022 Bank to Customer Statement example |                                                                                                             |
| ..examples/ach_iso20002/scenario/returns                    | Top level folder for the example.                                                                           |
| ..examples/ach_iso20002/scenario/returns/data               | Placeholder for the input test data.                                                                        |
| ..examples/ach_iso20002/scenario/returns/maps               | Placeholder for the maps.                                                                                   |
| ..examples/ach_iso20002/scenario/returns/schemas            | Placeholder for the example schemas.                                                                        |
| ACH Payment Related Information (PRI) validation example    |                                                                                                             |
| ..examples/ach_pri                                          | Top level folder for the example.                                                                           |
| ..examples/ach_pri/data                                     | Placeholder for the input test data.                                                                        |
| ..examples/ach_pri/maps                                     | Placeholder for the maps.                                                                                   |
| ..examples/ach_pri/schemas                                  | Placeholder for the example schema.                                                                         |
| ..examples/ach_pri/trees                                    | Placeholder for the example type trees.                                                                     |
| ACH Refused Notification of Change Entry example            |                                                                                                             |
| ..examples/ach_rfsc_cor                                     | Top level folder for the example.                                                                           |
| ..examples/ach_rfsc_cor/data                                | Placeholder for the input test data.                                                                        |
| ..examples/ach_rfsc_cor/maps                                | Placeholder for the maps.                                                                                   |
| ACH Returns and Notification of Change (COR) example        |                                                                                                             |

| Directory structure                 | Description                                                                                                                                                                          |
|-------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| ./examples/ach_rtrn_cor             | Top level folder for the example.                                                                                                                                                    |
| ./examples/ach_rtrn_cor/data        | Placeholder for the input test data.                                                                                                                                                 |
| ./examples/ach_rtrn_cor/maps        | Placeholder for the maps.                                                                                                                                                            |
| ACH Returned Entries Report example |                                                                                                                                                                                      |
| ./examples/ach_rtrn_csv             | Top level folder for the example.                                                                                                                                                    |
| ./examples/ach_rtrn_csv/data        | Placeholder for the input test data.                                                                                                                                                 |
| ./examples/ach_rtrn_csv/maps        | Placeholder for the maps.                                                                                                                                                            |
| ./examples/ach_rtrn_csv/trees       | Placeholder for the type tree.                                                                                                                                                       |
| ACH mapping from/to SWIFT example   |                                                                                                                                                                                      |
| ./examples/ach_swift                | Top level folder for the example.                                                                                                                                                    |
| ./examples/ach_swift/data           | Placeholder for the input test data.                                                                                                                                                 |
| ./examples/ach_swift/maps           | Placeholder for the maps.                                                                                                                                                            |
| ./examples/ach_swift/trees          | Placeholder for the type trees.                                                                                                                                                      |
| ACH validation example              |                                                                                                                                                                                      |
| ./examples/ach_validation           | Top level folder for the example.                                                                                                                                                    |
| ./examples/ach_validation/maps      | Placeholder for the maps.                                                                                                                                                            |
| ./examples/ach_validation/data      | Placeholder for the input data.                                                                                                                                                      |
| ./examples/ach_validation/schemas   | Placeholder for the example schemas.                                                                                                                                                 |
| ./examples/ach_validation/trees     | Placeholder for the type trees.                                                                                                                                                      |
| XML to ACH PPD example              |                                                                                                                                                                                      |
| ./examples/ach_xml                  | Top level folder for the example.                                                                                                                                                    |
| ./examples/ach_xml/data             | Placeholder for the input test data.                                                                                                                                                 |
| ./examples/ach_xml/maps             | Placeholder for the maps.                                                                                                                                                            |
| ./examples/ach_xml/schemas          | Placeholder for the example schemas.                                                                                                                                                 |
| ./examples/ach_xml/trees            | Placeholder for the type trees.                                                                                                                                                      |
| Common type trees                   |                                                                                                                                                                                      |
| ./examples/trees                    | Contains these type trees that are used by multiple examples: <ul style="list-style-type: none"> <li>• 5010_820.mtt</li> <li>• ach_error_codes.mtt</li> <li>• utility.mtt</li> </ul> |

## trees directory

The type tree that is contained in the trees directory represents the technical specifications of the ACH rules. These rules are defined in the appendixes of the *Operating Rules of the National Automated Clearing House Association*.

The ach\_vccyy.mtt type trees and the ach\_vccyy\_sfmt.mtt (streaming-format) type trees are contained in this directory.

In the file names mentioned here, ccyy indicates the version of the ACH rules. The ACH technical specifications for the year 2010 are supported by the 2009 type tree. For details, see the release notes.

- **[Support for streaming format messages](#)**

Support is provided for ACH messages without any record terminators. This format is referred to as streaming-format, and is supported by the ach\_vccyy\_sfmc.mtt type trees.

## Support for streaming format messages

Support is provided for ACH messages without any record terminators. This format is referred to as streaming-format, and is supported by the ach\_vccyy\_sfmc.mtt type trees.

The difference between ach\_vccyy.mtt and ach\_vccyy\_sfmc.mtt is support for record terminators. For ACH files with record terminators, it is recommended that you use the ach\_vccyy.mtt type tree. For ACH files without any record terminators, it is recommended that you use the ach\_vccyy\_sfmc.mtt type tree.

A list of the streaming-format type trees that are contained in the trees directory is provided in the release notes.

## NACHA type trees

This documentation describes the type trees that are included with the Pack for Financial Payments NACHA component.

- **[Overview of the NACHA type trees](#)**

The primary type trees that are provided in the Pack for Financial Payments NACHA component provide a full definition of the records and fields that are needed to create or validate an ACH file.

- **[Summary of ACH applications](#)**

A single batch can contain only one ACH application type. Supported ACH applications with brief descriptions are listed here. The *ACH Rulebook* summarizes the application types and groups them by type: consumer, corporate, or other.

- **[Return entries](#)**  
Return entries are generated by an ACH operator when data acceptance rules are violated. They are also generated by an RDFI for any reason with an appropriate return reason code specified by the ACH rules.
- **[Notification of change entries \(COR\)](#)**  
A notification of change entry is created in a Batch with an SEC code of COR.
- **[Acknowledgement records](#)**  
An acknowledgement (ACK or ATX) entry is created by an RDFI to notify ODFI of receipt of CCD or CTX, if such is requested by the CCD or CTX entry.
- **[Analysis errors](#)**  
There are no expected analysis errors for the type trees. They are analyzed for the supported ITX base product platforms, but the .dbe file is not delivered.
- **[Additional validation recommendations](#)**  
Most of the exceptions to validations affect multiple ACH types. If something is specific to one particular application type, that type is noted by its SEC code.

---

## Overview of the NACHA type trees

The primary type trees that are provided in the Pack for Financial Payments NACHA component provide a full definition of the records and fields that are needed to create or validate an ACH file.

An ACH file consists of one or more application batches. Each batch has a header record and batch control record. Each batch can have one or more entry detail records and in some cases, one or more addenda records. The entire file also has a single file header record at the beginning, a file control record and one or more filler records. The tree defines the groups that are needed for each record type, and also the sequencing of batches in the file.

The type names in the type trees are consistent with the terminology used in the NACHA published rules. The root of each tree is called ACH. Each tree has the following categories:

- Element
  - File
  - Group
  - Record - a partitioned group at the same level as the categories
- **[Data elements](#)**  
The types that are defined in this category element comprise the full set of data elements that are needed for any ACH record format.
- **[File category](#)**  
The File category defines the groups that define the full ACH file.
- **[Groups and records](#)**  
In the Group category, groups include all of the subsets and record groups that comprise the ACH file.

---

## Data elements

The types that are defined in this category element comprise the full set of data elements that are needed for any ACH record format.

There are more subcategories, and elements with inherited properties, to set the data definition in a way to allow adequate validation and mapping support when used in the context of a full ACH file.

The type names of data elements are abbreviated versions of the element descriptions. For example, the name of the type that defines the data element Addenda Record Indicator is *Addenda\_Ind*. If you find these, or other names inappropriate for your use, rename the type. Every reference to that type is renamed for you automatically.

Coded values for coded fields in records are implemented in the type tree as restriction lists, and component rules. These coded values are described in detail in the *ACH Rulebook; Appendix 2, Section 2.2*.

---

## File category

The File category defines the groups that define the full ACH file.

The starting point of a full mapping of an ACH file is to select either one of the following from the layout in the type tree structure: ACH/File/ACH/ADV\_Entries or ACH/File/ACH/ALL\_Entries.

---

## Groups and records

In the Group category, groups include all of the subsets and record groups that comprise the ACH file.

These subsets and record groups include the batch types that are needed to represent the supported application types. Many of the batches contain entry detail records with one or more addenda records. Because of this, groups are defined to cover one instance of a full entry in the ACH transaction.

In the type tree, these groups are named so that the validation takes place in the correct order to be able to properly distinguish between similar applications, types, and returns. The ACH rules do not allow for mixed SEC codes, or mixed applications, in a single batch. However, the file can contain multiple batches, each of a different application type.

When a new ACH file is created by use of the type tree, the batch control record for each batch must be mapped with correct counts/total amounts for all included entries.

Finally, the partitioned group Record contains all of the definitions of the individual records that might be used in the ACH file. There are extra partitions for addenda, BatchHeader, EntryDetail, and Returns records.

## Summary of ACH applications

A single batch can contain only one ACH application type. Supported ACH applications with brief descriptions are listed here. The *ACH Rulebook* summarizes the application types and groups them by type: consumer, corporate, or other.

In the following table, application types are listed alphabetically as they appear in the type tree, as groups defined in the partitioned group Batch in the category Group.

Table 1. ACH applications

| Application | Description                                          |
|-------------|------------------------------------------------------|
| ACK         | ACH Payment Acknowledgement (with addenda)           |
| ADV         | Automated Accounting Device                          |
| ARC         | Accounts Receivable Entry                            |
| ATX         | Financial EDI Acknowledgement (with Addenda)         |
| BOC         | Back Office Conversion Entries                       |
| CCD         | Corporate Credit or Debit Entry (with Addenda)       |
| CIE         | Customer-Initiated Entry (with Addenda)              |
| COR         | Notification of Change (with Addenda)                |
| CTX         | Corporate Trade Exchange (with Addenda)              |
| DNE         | Death Notification Entry (with Addenda)              |
| ENR         | Automated Enrollment Entry (with Addenda)            |
| IAT         | International ACH entries (with Addenda)             |
| POP         | Point of Purchase Entry                              |
| POS         | Point of Sale Entry (with Addenda)                   |
| SHR         | Shared Network Transaction (with Addenda)            |
| MTE         | Machine Transfer Entry (with Addenda)                |
| PPD         | Prearranged Payment and Deposit Entry (with Addenda) |
| RCK         | Represented Check Entry                              |
| TEL         | Telephone-Initiated Entry                            |
| TRC         | Check Truncation Entry                               |
| TRX         | Check Truncation Entries Exchange (with Addenda)     |
| WEB         | Internet-Initiated Entry (with Addenda)              |
| XCK         | Destroyed Check Entry                                |

## Return entries

Return entries are generated by an ACH operator when data acceptance rules are violated. They are also generated by an ODFI for any reason with an appropriate return reason code specified by the ACH rules.

The pack includes examples of how to populate a return, dishonored return, and contested dishonored return entries.

After a Return is sent, the ODFI can then dishonor that return, and as one final step, the RDFI can then contest the dishonored return. These have different reason codes and thus are defined separately in the type trees. The names for the groups that define the format of a batch or returns, dishonored returns, and contested dishonored returns are not abbreviated, and thus appear in the type tree as

- Returns
- DishonoredReturns
- ContestedDishonoredReturns

In the partitioned group, Batch Group, there are different formats for the returns that are related to cross border entries. For this reason, the following groups are also defined in the ACH type trees.

## Notification of change entries (COR)

A notification of change entry is created in a Batch with an SEC code of COR.

This type of entry is created by the Receiving Depository Financial Institution (RDFI) to notify the Originating Depository Financial Institution (ODFI) that previously valid information in an entry is now outdated or erroneous, and must be changed. Specifications for creating the Notification of Change records are detailed in the *ACH Rulebook; Appendix 6*. The ODFI can refuse the notification of Change and generate a Refused COR entry with an appropriate reason.

Although the Batch header for a batch that contains COR entries is the same for notification of changes for both consumer and corporate application types, the entry detail records are different. Thus the group that is defined in the type tree for COR contains a partitioned group COR Entry Detail Record to support either type of application.

The Pack for Financial Payments NACHA component comes with examples that show how to populate a Notification of Change and a Refused Notification of Change entry.

These are the names for the groups that define batches or Notification of Change entries and refused Notification of Change entries:

- COR
- RefusedCOR

As with returns, different formats are required for cross border entries. The ACH type trees also contain the following groups:

## Acknowledgement records

An acknowledgement (ACK or ATX) entry is created by an RDFI to notify ODFI of receipt of CCD or CTX, if such is requested by the CCD or CTX entry.

A refused acknowledgement entry is created by the ODFI to refuse an ACK that is incorrectly routed, incorrect, or incomplete.

The Pack for Financial Payments does not generate these automatically, but the type tree provides the appropriate definition to allow you to create your own map for them. These entries are defined by specific SEC codes, ACK and ATX. They are listed in the partitioned group, Batch Group, as the standard application types that are defined by this three letter code.

The NACHA component examples for IBM Sterling B2B Integrator come with an ACH Deenvelope service scenario to generate an ATX message from an Inbound CTX.

## Analysis errors

There are no expected analysis errors for the type trees. They are analyzed for the supported ITX base product platforms, but the .dbe file is not delivered.

For better performance in building any maps that are based on type trees that are delivered with the pack, analyze and save the type tree before you use it in a map. If the type tree is used in any maps, any of the IBM Transformation Extender map execution errors might apply, depending upon your application architecture.

## Additional validation recommendations

Most of the exceptions to validations affect multiple ACH types. If something is specific to one particular application type, that type is noted by its SEC code.

The following table shows the ACH rule, or field that is involved in the exception along with a detailed description.

Table 1. Exceptions to validations.

| ACH rule or field                                                                                                                                                                                                         | Explanation of exception and potential action                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| In all fields that are defined as alphanumeric, in the rules, the characters that are allowed are restricted to a subset of ASCII or EBCDIC.                                                                              | Because of performance concerns, a restriction list is not defined in the type tree to exclude the special ASCII or EBCDIC values. Fields will be left justified, space filled, and defined to the correct fixed size. Mandatory fields are checked to ensure that content is present. Required fields that only include spaces are considered valid by the tree.                                                                                                                                                                                            |
| Total Debit Entry Dollar Amount field and Total Credit Entry Dollar Amount field in Batch Control Record                                                                                                                  | If a message can be either a credit or a debit, the only way to enforce the debit/credit totals is through a map.                                                                                                                                                                                                                                                                                                                                                                                                                                            |
| Check Digit field                                                                                                                                                                                                         | The type tree does not contain a rule to ensure that the value for this field was computed correctly.                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| Effective Entry Date field, Settlement Date field in Batch Header                                                                                                                                                         | Checks against processing turnaround times and other date related rules need to be done as part of the overall implementation.                                                                                                                                                                                                                                                                                                                                                                                                                               |
| Identification number field in ENR                                                                                                                                                                                        | The type tree validation has no means to determine whether this is a Federal Government enrollment. Would need to be implemented in a validation map through lookups, or other means of confirmation.                                                                                                                                                                                                                                                                                                                                                        |
| Payment related information in the addenda records for various application types                                                                                                                                          | The rules call for an asterisk (*) to be the delimiter between data elements, and a backward slash (\), or a tilde (~) to be the terminator between data segments, but this validation must be done in a map. For more information, see the <a href="#">ACH Payment Related Information example</a> . The Payment Related Information field in all of the addenda records must be parsed to validate the data. Similarly, to ensure that the content in this field is ANSI X12 or NACHA, endorsed banking conventions must be validated separately in a map. |
| Rules regarding valid sending point, valid routing numbers                                                                                                                                                                | Other than data format validation, the type tree validation does not provide lookup capabilities against tables of valid ACH participant information.                                                                                                                                                                                                                                                                                                                                                                                                        |
| Rules regarding matching field values (for example, Immediate Origin, File Creation Date, File Creation Time and File ID Modifier) back to original entries or related ACH files in a particular application process flow | The rules must be included in the overall application. In other words, the type tree does not perform records matching within the content of the data, whereas the validation map performs this matching task and reports any duplicates                                                                                                                                                                                                                                                                                                                     |
| Originator City field and State/Province field in the Third IAT addenda record, and the Receiver City field and State/Province field in the Seventh IAT addenda record.                                                   | The State/Province portion of these fields is not mandatory in all cases, but only "if applicable". If the overall implementation is limited to situations where this is applicable, then additional validation could be implemented in a validation map.                                                                                                                                                                                                                                                                                                    |
| Currency Codes                                                                                                                                                                                                            | The type tree does not restrict the value to a valid ISO Currency Code. To make certain the Currency Code is valid, the ISO Currency Code values must be added as restrictions to all Currency Code elements.                                                                                                                                                                                                                                                                                                                                                |
| DFI Account Number field                                                                                                                                                                                                  | The type tree does not provide look up capabilities against validation of the DFI Account number                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
| Rules regarding TRC Entries and TRX Entries                                                                                                                                                                               | TRC Entries and TRX Entries are permitted only to eligible participants. An ODFI may transmit a TRC entry or TRX entry to an RDFI only if the ODFI and RDFI each are participants in the same Check Truncation Program. The type tree does not contain rules to ensure that ODFI or RDFI are both participants in the same Check Truncation Program. If the overall implementation is limited to situations where this is applicable, then additional validation could be implemented in a validation map.                                                   |
| Rules regarding loan accounts                                                                                                                                                                                             | Originators may transmit ACH credit entries to loan accounts belonging to a Receiver. Originators must be aware that, with the exception of reversals to correct erroneous credit, debit entries to loan accounts are prohibited. The type tree does not contain rules that enforce this entry detail rejection criteria.                                                                                                                                                                                                                                    |
| Rules regarding Office of Foreign Assets Control (OFAC) enforced sanctions policies.                                                                                                                                      | The type tree does not have rules to enforce OFAC sanctions policies, such as reporting entities referred to as "Specially Designated Nationals and Blocked Person" acting as ACH participants.                                                                                                                                                                                                                                                                                                                                                              |

## Creating an application specific type tree

These procedures describe how to create NACHA type trees for your specific application.

Using a subset type tree, as described here, can improve mapping run-time performance, as there are fewer application group partitions to test during validation. In addition, the size (disk space required) of the type tree source file can be reduced (minimized). Data validation of a type tree starts at the top, and moves down the list of SEC subtypes, attempting validation of each subtype against the data until a match is found. Use the procedures in this section to "prune" the appropriate version of the Automated Clearing House (ACH) type tree to meet your needs.

Instead of merely deleting the unneeded types directly in the type tree, there is a two-step process that is required to "prune" the tree in order to make it the most efficient:

- First, create a tree by copying the current ach\_vccyy, or ach\_vccyy\_sfmt file and deleting unnecessary application types.  
Note: In the file names above, ccyy indicates the year of the application standards, for example, 2009, or 2015.
- To remove unused components from the type tree that was created in the previous step, create another tree and merge the File type into it.
- **[Copying NACHA type trees](#)**  
To copy standard NACHA type trees:
- **[Removing unnecessary application types](#)**  
In this procedure, only the required application types are retained.
- **[Creating the target type tree](#)**  
The type tree that is created in this procedure is the type tree that contains the industry subsets.
- **[Merging the application type to the new type tree](#)**  
The types that are retained in the type tree are merged into the target type tree to create the industry subsets.

---

## Copying NACHA type trees

To copy standard NACHA type trees:

1. Open the NACHA type tree that you want to copy.
2. From the File menu, choose Save As.
3. Enter a new name in the File name field. For example: myach\_vccyy.mtt, where ccyy is the year of the ACH Standards being copied.
4. Click Ok.

---

## Removing unnecessary application types

In this procedure, only the required application types are retained.

In addition, when you map using the new NACHA type tree, only the necessary batch types appear in the Map Designer. In this example, the assumed application is the WEB application, a consumer application for origination of debit entries via the internet. To remove unnecessary application types:

1. In the new type tree, select Group.
2. Right click on the type and choose the Expand All Subsystems from the context menu.
3. Right-click on the group Batch and choose Select All Subtypes. All NACHA application subtypes are selected (highlighted).
4. For each application type to be retained in the subset tree, locate the subtype for that functional group. For example, locate the type WEB under Batch.
5. Hold down the Ctrl key, and then click each subtype that you want to retain. All the items to be deleted remain selected.
6. After all application subtypes that remain in the subset tree are no longer selected, press the Delete key (or choose Delete from the Type menu).
7. Remove the ADV under the Group category and ADV\_Entries under File category if the ADV\_Batch was deleted in the preceding steps. Save the type tree.

---

## Creating the target type tree

The type tree that is created in this procedure is the type tree that contains the industry subsets.

To create the new type tree, and name the root type:

1. From the File menu, choose the New-Type Tree.
2. In the New Type Tree window, select the parent folder and the file name for the new type tree, and then click Finish.
3. A new type tree is created in the window, with the Root type.
4. In the Properties window, in the Name field, change the name of the root type from Root to ACH.
5. From the File menu, choose Save.

---

## Merging the application type to the new type tree

The types that are retained in the type tree are merged into the target type tree to create the industry subsets.

To merge the application type to the new type tree:

1. Right-click the File category in the type tree that you modified in the [Copying NACHA type trees](#) steps (myach\_vccyy.mtt), then select Merge from the context menu. When the Merge window opens, the Merge Sub-Tree check box is grayed out.
2. Go to the new type tree that you created in [Creating the target type tree](#) and click anywhere in the new type tree window to populate the To field in the Merge window.
3. When the name of the new tree shows in the To Tree box, click Merge, and then click Close.

For scenarios where the application subtype, or subtypes selected are all non-return entries, on the second component of ACH\Group\ALL, edit the rule on Batch(s) by commenting out a specific line in the rule as follows:

```
/*COUNT(EntryDetail Returns Record IN $) */
```

For scenarios where the application subtype, or subtypes selected are all return entries (Contested Dishonored Return, Dishonored Returns, Refused Notification of Change, Returns), on the second component of ACH\Group\ALL , edit rule on Batch(s) by commenting out a specific line in the rule as follows:

```
/*COUNT(EntryDetail Record IN $) */
```

You can now save and begin to use the type tree.

## NACHA component examples

The Pack for Financial Payments NACHA component contains examples to assist you in using the pack. Instructions for using these examples are contained in this documentation. The ACH examples that install with the pack include the following:

- ACH mapping from and to EDI
- ACH mapping from and to SWIFT
- ACH to ISO 20022 Payment Initiation Credit Transfer
- ACH to ISO 20022 Payment Initiation Direct Debit
- ACH Reject to ISO 20022 Transformation
- ACH Returns to ISO 20022 Transformation
- XML to ACH Prearrange Payment and Deposit Entry (PPD)
- ACH Returns and Notification of Change (COR)
- ACH Refused Notification of Change
- ACH Dishonored Return Entry
- ACH Contested Dishonored Return Entry
- ACH Payment Related Information (PRI) validation
- ACH Return Entries Report
- ACH Validation

Other examples are included to be executed within IBM Sterling B2B Integrator ACH service.

The following are example scenarios:

- ACH Deenvelope - Corporate Trade Exchange Entries (CTX)
- ACH Deenvelope - Death Notification Entries (DNE)
- ACH Envelope - Corporate Trade Exchange Entries (CTX)
- ACH Envelope - Death Notification Entries (DNE)

Note: IBM Sterling B2B Integrator examples are available only when the pack is used with the Transformation Extender version 8.4 (and later) base product.

- **[ACH EDI example](#)**

The ACH mapping from and to EDI example represents conversions from and to the Corporate Credit or Debit (CCD), as well as the Corporate Trade Exchange (CTX) ACH entries.

- **[ACH SWIFT example](#)**

The example represents conversions to and from the International ACH Transaction (IAT) and the SWIFT MT103+ Single Customer Credit Transfer.

- **[ACH to ISO 20022 Credit Transfer example](#)**

The ACH to ISO 20022 example demonstrates how to transform an ACH file to an ISO 20022 XML document, as well as how to transform an ISO 20022 XML document to an ACH file.

- **[ACH to ISO 20022 Direct Debit example](#)**

The ACH to ISO 20022 Payment Initiation Direct Debit example demonstrates how to transform an ACH file to an ISO 20022 XML document, as well as how to transform an ISO 20022 document to an ACH file.

- **[ACH ISO 20022 Rejects example](#)**

The ACH ISO 20022 Rejects example demonstrates how to transform an ACH file to an ISO 20022 XML document, as well as how to transform an ISO 20022 XML document to an ACH file.

- **[ACH ISO 20022 Returns example](#)**

The ACH ISO 20022 Returns example demonstrates how to transform an ACH file to an ISO 20022 XML document, as well as how to transform an ISO 20022 XML document to an ACH file.

- **[XML to ACH PPD example](#)**

The example map illustrates how information is mapped from a back-office XYZ Trust XML file into the ACH PPD record layouts.

- **[ACH Returns and Notification of Change \(COR\) example](#)**

The example map illustrates how a return and notification of change is prepared.

- **[ACH Refused Notification of Change Entry example](#)**

The example map shows how a Refused Notification of Change (COR) entry is prepared.

- **[ACH Dishonored Return example](#)**

The ACH Dishonored Return example map shows how a dishonored return entry is prepared.

- **[ACH Contested Dishonored Return Entry example](#)**

The example map shows how an ACH contested dishonored return entry is prepared.

- **[ACH Payment Related Information example](#)**

The ACH Payment Related Information example demonstrates how to validate the content of the Payment Related Information field on the Addenda record, if it conforms to NACHA endorsed banking conventions.

- **[ACH Return Entries Report example](#)**

The example map shows how ACH return entries are extracted from an ACH file. The return entries are reported in categories that are returned due to administrative, unauthorized reasons and all other reasons (except for returned RCK entries).

- **[ACH Validation example](#)**

The ACH Validation example demonstrates how to validate an ACH file if it conforms to NACHA Operating Rules. The base implementations are derived from the ACH type treerules and are not designed to replace the use of the ACH type tree.

- [How to download output files](#)  
Download the output files.
  - [IBM Sterling B2B Integrator examples](#)  
See IBM Sterling B2B Integrator for a complete description of these examples, including deployment instructions.
- 

## ACH EDI example

The ACH mapping from and to EDI example represents conversions from and to the Corporate Credit or Debit (CCD), as well as the Corporate Trade Exchange (CTX) ACH entries.

- [What the example contains](#)  
The ACH EDI example contains these files, schemas, and maps.
  - [How to run the example in Design Studio](#)  
These steps describe how to run the ACH EDI example.
  - [How to run the example in Design Server](#)  
Run instructions for the ACH EDI example.
  - [Scenarios](#)  
The example maps will illustrate the following:
  - [Assumptions](#)  
The following assumptions are made regarding the ACH Mapping from and to EDI example:
- 

## What the example contains

The ACH EDI example contains these files, schemas, and maps.

### Files

These files are contained in the example:

- 820\_for\_ccd.in - input to the x12raccd transformation map which contains remittance advice (820) transactions.
- ccd\_valid.in - input to the ccdx12ra transformation map which contains ACH Corporate Credit (CCD+) entries.
- 820\_for\_ctx.in - input to the x12ractx transformation map which contains remittance advice (820) transactions.
- ctx\_valid.in - input to the ctxx12ra transformation map which contains ACH Corporate Trade Exchange (CTX) entries.

### Schemas

These schemas are contained in the example:

- 5010\_820 - type tree that represents an EDI 820 X12 version 5010X218.
- utility - miscellaneous schema used in sample maps.
- ach\_v2024 - current version of NACHA type tree.

### Maps

These maps are contained in the example:

- addenda - utility map that is called by x12ractx map to populate the ACH Corporate Trade Exchange (CTX) addenda records.
  - ccdx12ra - used to transform a valid ACH Corporate Credit (CCD+) entry to a valid X12 5010X218 820 Payment Advice Order/Remittance.
  - ctxx12ra - used to transform a valid ACH Corporate Trade Exchange (CTX) entry to a valid X12 5010X218 820 Payment Order/Remittance Advice.
  - x12raccd - used to transform a valid X12 5010X218 820 Payment Order/Remittance Advice to a valid ACH Corporate Credit (CCD+) entry.
- 

## How to run the example in Design Studio

These steps describe how to run the ACH EDI example.

### Procedure

1. Use the Design Studio to open the following map source file: ach\_edi.mms
2. Build all the maps in the source file, and then run the following transformation maps:  
Note: Comments which are associated with the rules are based upon situations which are illustrated by the example.
  - x12raccd and ccdx12ra
  - x12ractx and ctxx12ra

# How to run the example in Design Server

Run instructions for the ACH EDI example.

## About this task

These steps describe how to run the ACH EDI example in Design Server.

## Procedure

1. Import the nachaEDI.zip file into the Design Server UI. See the release notes for import instructions.
2. Create a package for the project and build all the maps.
3. Open the nachaEDI project in the Design Server UI.
4. Run the following maps:
  - x12ractx
  - ctxx12ra
  - ccdx12ra
  - x12raccd
5. View the output. Go to the map Diagram-Result, and hover the cursor over Target, then select View Data.
6. To download the output file, see [How to download output files](#).

## Scenarios

The example maps will illustrate the following:

- How a Corporate Credit or Debit (CCD) entry is converted from an EDI X12 820 Payment Order/Remittance Advice, and conversely, how an EDI X12 820 Payment Order Remittance Advice is converted from a CCD entry.
- How a Corporate Trade Exchange (CTX) entry is converted from an EDI X12 820 Payment Order/Remittance Advice and conversely, how an EDI X12 820 Payment Order/Remittance Advice is converted from a CTX entry.

## Assumptions

The following assumptions are made regarding the ACH Mapping from and to EDI example:

- The ccdx12ra example map transforms a valid Corporate Credit or Debit (CCD) entry to a valid EDI X12 5010X218 820 Payment Order/Remittance Advice. For mapping data from the CCD to the EDI X12 820 (Remittance Advice), the CCD plus addenda format is necessary. The Payment Related Information element in the addenda record of a CCD can contain either NACHA endorsed banking conventions or Payment Related EDI X12 Segments. This example has been implemented with the payment related information in the CCD containing EDI X12 RMR Segments. The RMR Segment in the EDI X12 820 is populated with the payment related information from the CCD addenda record.
- The x12raccd example map transforms a valid EDI X12 5010X218 820 Payment Order/Remittance Advice to a valid Corporate Credit or Debit (CCD) entry. Calculations at the Batch Control and File levels are performed. This example maps the TRN Segment in the EDI X12 5010X218 820 to the Payment Related Information element in the CCD Addenda record.
- The x12ractx example map transforms a valid EDI X12 5010X218 820 Payment Order/Remittance Advice to a valid Corporate Trade Exchange (CTX) entry. Calculations at the Batch Control and File levels are performed.
- The ctxx12ra example map transforms a valid Corporate Trade Exchange (CTX) entry to a valid EDI X12 5010X218 820 Payment Order/Remittance Advice.

## ACH SWIFT example

The example represents conversions to and from the International ACH Transaction (IAT) and the SWIFT MT103+ Single Customer Credit Transfer.

- [What the example contains](#)  
The ACH SWIFT example contains these files, schemas, and maps.
- [How to run the example in Design Studio](#)  
Run the map example as follows.
- [How to run the example in Design Server](#)  
Run instructions for the ACH SWIFT example.
- [Scenarios](#)  
The example maps show how an International ACH Transaction (IAT) entry is converted from a SWIFT MT103+ Single customer Credit Transfer.
- [Assumptions](#)  
This example makes the following assumptions:

## What the example contains

The ACH SWIFT example contains these files, schemas, and maps.

## Files

---

These files are contained in the example:

- ach\_iat\_example.in - input to the iatmt103 transformation map which contains a valid International ACH Transaction (IAT) entry.
- mt103\_plus\_example.in - input to the mt103iat transformation map which contains a valid SWIFT MT103 message.

## Schemas

---

These schemas are contained in the example:

- swift\_mt103 - type tree that represents a SWIFT MT103+ message.
- ach\_v2024 - current version of NACHA type tree.

## Maps

---

These maps are contained in the example:

- iatmt103 - used to transform a valid International ACH Transaction (IAT) entry to a valid SWIFT MT103+ Single Customer Credit Transfer.
- mt103iat - used to transform a valid MT103+ Single Customer Credit Transfer to a valid International ACH Transaction (IAT) entry.

## How to run the example in Design Studio

Run the map example as follows.

### Procedure

---

1. Use the Design Studio to open the following map source file: ach\_swift.mms.
2. Build all the maps in the source file and then run the transformation maps iatmt103 and mt103iat.  
Note: Comments that are associated with the rules are based on situations that are illustrated by the example.

## How to run the example in Design Server

Run instructions for the ACH SWIFT example.

### About this task

---

Follow these steps to run the ACH SWIFT example.

### Procedure

---

1. Import the nachaSWIFT.zip file into the Design Server UI. See the release notes for import instructions.
2. Create a package for the project and build all the maps.
3. Open the **nachaSWIFT** project in the Design Server UI.
4. Run the following maps:
  - mt103iat
  - iatmt103
5. View the output. Go to the map Diagram-Result, and hover the cursor over Target, then select View Data.
6. To download the output file, see [How to download output files](#).

## Scenarios

---

The example maps show how an International ACH Transaction (IAT) entry is converted from a SWIFT MT103+ Single customer Credit Transfer.

### About this task

---

The maps also show how a SWIFT MT103+ Single customer Credit transfer is converted to an IAT entry.

## Assumptions

---

This example makes the following assumptions:

- The iatmt103 example map transforms a valid International ACH Transaction (IAT) with a single batch and entry detail records. There are seven addenda records to a SWIFT MT103+ Single Customer Credit Transfer. Information provided on the addenda records is mapped to the closest equivalent SWIFT MT103+ fields.
- The mt103iat example map transforms a valid SWIFT MT103+ Single Customer Credit Transfer to an International ACH Transaction (IAT) entry. Calculations at the Batch Control and File levels are performed.

## ACH to ISO 20022 Credit Transfer example

The ACH to ISO 20022 example demonstrates how to transform an ACH file to an ISO 20022 XML document, as well as how to transform an ISO 20022 XML document to an ACH file.

The transformation is limited to the following Standard Entry Class (SEC) code being transformed into an ISO 20022 Payment Initiation (pain.001.001.03) Credit Transfer file.

- Corporate Credit or Debit (CCD)
- Prearranged Payment and Deposit (PPD)
- Corporate Trade Exchange (CTX)
- International ACH Transaction (IAT)

The implementations are based from the NACHA ISO 20022 Tool for pain.001 (credit transfer) Nov 2016.xlsx document

The example references the *ACH\_ISO\_Mapping\_Assumptions\_Table (Credit Transfer).xlsx* table. These tables install with the nachaISO20022MappingDoc.zip file.

- **What the example contains**  
The ACH ISO 20022 Credit Transfer example contains these files, schemas, and maps.
- **How to run the example in Design Studio**  
These steps describe how to run the ACH to ISO 20022 Credit Transfer example.
- **How to run the example in Design Server**  
Run instructions for the ACH ISO20022 Credit Transfer example.
- **Scenarios**  
The ACH to ISO 20022 Payment Initiation Credit Transfer example maps demonstrate file validation prior to transformation to a pain.001.001.03 document. Predefined rules must be followed to ensure valuation.
- **Assumptions**  
The example maps enforce the pre-conversion rules and, given that no rules are violated, the maps proceed to the transformation phase.

## What the example contains

The ACH ISO 20022 Credit Transfer example contains these files, schemas, and maps.

### Files

These files are contained in the example:

- ach\_ccd\_ct\_input.txt - input to the ccdiso01 map which contains a total of 5 CCD batch entries.
- ach\_ppd\_ct\_input.txt - input to the ppdiso01 map which contains a total of 1 PPD batch entry.
- ach\_ctx\_ct\_input.txt - input to the ctxiso01 map which contains a total of 2 CTX batch entries.
- ach\_iat\_ct\_input.txt - input to the iatiso01 map which contains a total of 2 IAT batch entries.
- iso\_pain001\_to\_ccd\_ct\_input.xml - input to the isoccd01 map which contains a total of 5 Payment Information level records.
- iso\_pain001\_to\_ppd\_ct\_input.xml - input to the isoppd01 map which contains a total of 1 Payment Information level record.
- iso\_pain001\_to\_ctx\_ct\_input.xml - input to the isoctx01 map which contains a total of 2 Payment Information level records.
- iso\_pain001\_to\_iat\_ct\_input.xml - input to the isoiat01 map which contains a total of 2 Payment Information level records.
- ach\_error\_codes.txt - list of error messages used in the pre-conversion rule verification step.

### Schemas

These schemas are contained in the example:

- pre\_conv\_rules- type tree that serves as a placeholder for the pre-conversion rules that are enforced in the transformation maps.
- ach\_error\_codes- metadata that represents the ach\_error\_codes.txt file which lists all error codes.
- ach\_v2024 - current version of NACHA type tree.
- pain.001.001.03.xsd - XML schema that represents the ISO 20022 Payment Initiation document.

### Maps

These maps are contained in the example:

- ccdiso01 - map used to transform an ACH Corporate Credit or Debit (CCD) file to an xml Payment Initiation (pain.001.001.03) Credit Transfer.

- ppdiso01 - map used to transform an ACH Prearranged Payment and Deposit (PPD) file to an xml Payment Initiation (pain.001.001.03) Credit Transfer.
- ctxiso01 - map used to transform an ACH Corporate Trade Exchange (CTX) file to an xml Payment Initiation (pain.001.001.03) Credit Transfer.
- iatiso01 - map used to transform an ACH International ACH Transaction (IAT) file to an xml Payment Initiation (pain.001.001.03) Credit Transfer.
- isoccd01 - map used to transform an xml Payment Initiation (pain.001.001.03) Credit Transfer to an ACH Corporate Credit or Debit (CCD) file.
- isoppd01 - map used to transform an xml Payment Initiation (pain.001.001.03) Credit Transfer to an ACH Prearranged Payment and Deposit (PPD) file.
- isoctx01 - map used to transform an xml Payment Initiation (pain.001.001.03) Credit Transfer to an ACH Corporate Trade Exchange (CTX) file.
- isoiat01 - map used to transform an xml Payment Initiation (pain.001.001.03) Credit Transfer to an ACH International ACH Transaction (IAT) file.

## How to run the example in Design Studio

These steps describe how to run the ACH to ISO 20022 Credit Transfer example.

### About this task

Run the map example as follows:

### Procedure

1. Use the Design Studio to open the following map source files:
  - a. ACH to ISO 20022
    - achiso\_ccd\_ct.mms
    - achiso\_ppd\_ct.mms
    - achiso\_ctx\_ct.mms
    - achiso\_iat\_ct.mms
  - b. ISO 20022 to ACH
    - isoach\_ccd\_ct.mms
    - isoach\_ppd\_ct.mms
    - isoach\_ctx\_ct.mms
    - isoach\_iat\_ct.mms
2. Build all the following maps in the source file:
  - ccdiso01 (transforms CCD to pain.001.001.03)
  - ppdiso01 (transforms PPD to pain.001.001.03)
  - ctxiso01 (transforms CTX to pain.001.001.03)
  - iatiso01 (transforms IAT to pain.001.001.03)
  - isoccd01 (transforms pain.001.001.03 to CCD)
  - isoppd01 (transforms pain.001.001.03 to PPD)
  - isoctx01 (transforms pain.001.001.03 to CTX)
  - isoiat01 (transforms pain.001.001.03 to IAT)
3. Set all compiled maps with a .mmc extension.
4. Define the following directories for those operating systems other than Windows:
  - data
  - maps
  - schemas
5. When the artifacts listed here are copied to operating systems other than Windows, make certain that the FTP settings are set as described sub-steps a and b, below:
  - a. FTP settings must be set to TEXT mode for the following input data:
    - ach\_ccd\_ct\_input.txt
    - ach\_ppd\_ct\_input.txt
    - ach\_ctx\_ct\_input.txt
    - ach\_iat\_ct\_input.txt
    - ach\_error\_codes.txt
  - b. FTP settings must be set to BINARY mode for the following:
    - all compiled maps
    - all xml schema files
    - iso\_pain001\_to\_ccd\_ct\_input.xml
    - iso\_pain001\_to\_ppd\_ct\_input.xml
    - iso\_pain001\_to\_ctx\_ct\_input.xml
    - iso\_pain001\_to\_iat\_ct\_input.xml

Note: MVS users, perform both step 6 and step 7. Non-MVS users go directly to step 7.
6. **MVS only:** Edit the following map card setting of the Schema > Type > Metadata to the //DD:ddname where ddname references the XML schema, pain.001.001.03.xsd:
  - a. Edit the ddname references on output card #2 for the following maps:
    - ccdiso01
    - ppdiso01
    - ctxiso01
    - iatiso01
  - b. Edit the ddname references on input card #1 for the following maps:
    - isoccd01
    - isoppd01
    - isoctx01
    - isoiat01

7. Run the maps listed in step 2 of this procedure.

## How to run the example in Design Server

Run instructions for the ACH ISO20022 Credit Transfer example.

### About this task

These steps describe how to run the ACH ISO20022 Credit Transfer example.

### Procedure

1. Import the nachaISO20022CreditTransfer.zip file into the Design Server UI. See the release notes for import instructions.
2. Create a package for the project and build all the maps.
3. Open the **nachaISO20022CreditTransfer** project in the Design Server UI.
4. Run the following maps:
  - ctxiso01
  - ccdiso01
  - ppdiso01
  - isoccd01
  - isoppd01
  - isoctx01
  - iatiso01
  - isoiat01
5. View the output. Go to the map Diagram-Result, and hover the cursor over Target, then select View Data.
6. To download the output file, see [How to download output files](#).

### Scenarios

The ACH to ISO 20022 Payment Initiation Credit Transfer example maps demonstrate file validation prior to transformation to a pain.001.001.03 document. Predefined rules must be followed to ensure valuation.

The rules are defined in the *ACH\_ISO\_Mapping\_Assumptions\_Table (Credit Transfer).xlsx* table. This table includes the pre-conversion rules needed to validate between ACH to ISO 20022, and ISO 20022 to ACH.

The *ACH\_ISO\_Mapping\_Assumptions\_Table (Credit Transfer).xlsx* table installs with the pack in the following directory:

<install\_dir>/packs/financial\_payments\_vn.n.n.n/nacha/examples/ach\_iso20022/scenario/assumptions

The table is also available on the Support website: <http://www.ibm.com/support/docview.wss?uid=ibm10716007>

### Assumptions

The example maps enforce the pre-conversion rules and, given that no rules are violated, the maps proceed to the transformation phase.

Mapping is done based on the *NACHA ISO 20022 Tool for pain.001 (credit transfer) Nov 2016.xlsx* document.

The *ACH\_ISO\_Mapping\_Assumptions\_Table (Credit Transfer).xlsx* includes some implementation specifics about mapping to and from ACH and ISO pain.001.001.03.

These tables are installed with the nachaISO20022MappingDoc.zip file.

### Expected results

- The ACH to ISO example maps generate an XML document that passes pain.001.001.03 XML schema validation.
- The ISO to ACH example maps generate an ACH file that passes ACH type tree validation.

## ACH to ISO 20022 Direct Debit example

The ACH to ISO 20022 Payment Initiation Direct Debit example demonstrates how to transform an ACH file to an ISO 20022 XML document, as well as how to transform an ISO 20022 document to an ACH file.

The transformation is limited to the following Standard Entry Class (SEC) code being transformed into an ISO 20022 Payment Initiation (pain.008.001.02) Direct Debit file:

- Corporate Credit or Debit (CCD)
- Prearranged Payment and Deposit (PPD)

The implementations are based on the *NACHA ISO 20022 Tool for pain.008 (Direct Debit) Nov 2016.xlsx* document.

The example references the *ACH\_ISO\_Mapping\_Assumptions\_Table (Direct Debit).xlsx* table. These tables install with the nachaISO20022MappingDoc.zip file.

- [What the example contains](#)  
The ACH ISO 20022 Direct Debit example contains these files, schemas, and maps.
- [How to run the example in Design Studio](#)  
These steps describe how to run the ACH to ISO 20022 Payment Initiation Direct Debit example.
- [How to run the example in Design Server](#)  
Run instructions for the ACH ISO20022 Direct Debit example.
- [Scenarios](#)  
The ACH to ISO 20022 Payment Initiation Direct Debit example maps show an ACH file being validated if the file conforms to predefined rules, prior to transformation to a pain.008.001.02 document.
- [Assumptions](#)  
The example maps enforce the pre-conversion rules and if no rules are violated, the maps proceed to the transformation phase.

---

## What the example contains

The ACH ISO 20022 Direct Debit example contains these files, schemas, and maps.

### Files

---

These files are contained in the example:

- ach\_ccd\_dd\_input.txt - input to the ccdiso08 map which contains a total of 5 CCD batch entries.
- ach\_ppd\_dd\_input.txt - input to the ppdiso08 map which contains a total of 1 PPD batch entry.
- iso\_pain008\_to\_ccd\_dd\_input.xml - input to the isoccd08 map which contains a total of 5 Payment Information level records.
- iso\_pain008\_to\_ppd\_dd\_input.xml - input to the isoppd08 map which contains a total of 1 Payment Information level record.
- ach\_error\_codes.txt - list of error messages used in the pre-conversion rule verification step.

### Schemas

---

These schemas are contained in the example:

- pre\_conv\_rules- type tree that serves as place holder for pre-conversion rules enforced in the transformation maps.
- ach\_error\_codes- type tree that represents the ach\_error\_codes.txt file which lists all error codes.
- ach\_v2024 - current version of NACHA type tree.
- pain.008.001.02.xsd - XML schema that represents the ISO 20022 Payment Initiation document.

### Maps

---

These maps are contained in the example:

- ccdiso08 - map used to transform an ACH Corporate Credit or Debit (CCD) file to an xml Payment Initiation (pain.008.001.02) Direct Debit.
- ppdiso08 - map used to transform an ACH Prearranged Payment and Deposit (PPD) file to an xml Payment Initiation (pain.008.001.02) Direct Debit.
- isoccd08 - map used to transform an xml Payment Initiation (pain.008.001.02) Direct Debit to an ACH Corporate Credit or Debit (CCD) file.
- isoppd08 - map used to transform an xml Payment Initiation (pain.008.001.02) Direct Debit to an ACH Prearranged Payment and Deposit (PPD) file.

---

## How to run the example in Design Studio

These steps describe how to run the ACH to ISO 20022 Payment Initiation Direct Debit example.

### About this task

---

Run the map example as follows:

### Procedure

---

1. Use the Design Studio to open the following map source files:
  - a. ACH to ISO 20022
    - achiso\_ccd\_dd.mms
    - achiso\_ppd\_dd.mms
  - b. ISO 20022 to ACH
    - isoach\_ccd\_dd.mms
    - isoach\_ppd\_dd.mms
2. Build all the following maps in the source file:
  - ccdiso08 (transforms CCD to pain.008.001.02)
  - pdiso08 (transforms PPD to pain.008.001.02)

- isoccd08 (transforms pain.008.001.02 to CCD)
  - isoppd08 (transforms pain.008.001.02 to PPD)
3. Set all compiled maps with a .mmc extension.
4. Define the following directories for those operating systems other than Windows:
- data
  - maps
  - schemas
5. When the artifacts listed here are copied to operating systems other than Windows, make certain that the FTP settings are set as described in sub-steps a and b:
- a. FTP settings must be set to TEXT mode for the following input data:
    - ach\_ccd\_dd\_input.txt
    - ach\_ppd\_dd\_input.txt
    - ach\_error\_codes.txt
  - b. FTP settings must be set to BINARY mode for the following:
    - all compiled maps
    - all xml schema files
    - iso\_pain008\_to\_ccd\_dd\_input.xml
    - iso\_pain008\_to\_ppd\_dd\_input.xml
- Note: MVS users, perform both step 6 and step 7. Non-MVS users go directly to step 7.
6. **MVS only:** Edit the following map card setting of the Schema > Type > Metadata to the //DD:ddname where ddname references the XML schema, pain.008.001.02.xsd.
- a. Edit the ddname references on output card #2 (pain.008.001.02.xsd) for the following maps:
    - ccdiso08
    - ppdiso08
  - b. Edit the ddname references on input card #1 (pain.008.001.02.xsd) for the following maps:
    - isoccd08
    - isoppd08
7. Run the maps listed in step 2 of this procedure.

## How to run the example in Design Server

Run instructions for the ACH ISO20022 Direct Debit example.

### About this task

These steps describe how to run the ACH ISO20022 Direct Debit example.

### Procedure

1. Import the nachaISO20022DirectDebit.zip file into the Design Server UI. See the release notes for import instructions.
2. Create a package for the project and build all the maps.
3. Open the **nachaISO20022DirectDebit** project in the Design Server UI.
4. Run the following maps:
  - ccdiso08
  - ppdiso08
  - isoccd08
  - isoppd08
5. View the output. Go to the map Diagram-Result, and hover the cursor over Target, then select View Data.
6. To download the output file, see [How to download output files](#).

## Scenarios

The ACH to ISO 20022 Payment Initiation Direct Debit example maps show an ACH file being validated if the file conforms to predefined rules, prior to transformation to a pain.008.001.02 document.

The rules are defined in the *ACH\_ISO\_Mapping\_Assumptions\_Table (Direct Debit).xlsx* table. This table includes pre-conversion rules needed between ACH to and from ISO pain.008.001.02.

The *ACH\_ISO\_Mapping\_Assumptions\_Table (Direct Debit).xlsx* installs with the pack in the following directory:

<install\_dir>/packs/financial\_payments\_vn.n.n/nacha/examples/ach\_iso20022/scenario/assumptions

The table is also available on the Support website: <http://www.ibm.com/support/docview.wss?uid=ibm10716007>

## Assumptions

The example maps enforce the pre-conversion rules and if no rules are violated, the maps proceed to the transformation phase.

- Mapping is done based on the *NACHA ISO 20022 Tool for pain.008 (Direct Debit) Nov 2016.xlsx* document. These tables install with the nachaISO20022MappingDoc.zip file.
- The *ACH\_ISO\_Mapping\_Assumptions\_Table (Direct Debit).xlsx* document includes implementation specifics on the mapping between ACH, to and from ISO pain.008.

## Expected results

---

- The ACH to ISO example maps generate an XML document which passes pain.008.001.02 XML schema validation.
  - The ISO to ACH example map generates an ACH file which passes ACH type tree validation.
- 

## ACH ISO 20022 Rejects example

The ACH ISO 20022 Rejects example demonstrates how to transform an ACH file to an ISO 20022 XML document, as well as how to transform an ISO 20022 XML document to an ACH file.

The transformation is limited to the following Standard Entry Class (SEC) code being transformed into an ISO 20022 Customer Payment Status Report message (pain.002.001.03) Credit Transfer and Direct Debit file.

- Corporate Credit or Debit (CCD)
- Prearranged Payment and Deposit (PPD)
- Corporate Trade Exchange (CTX)
- ACH International ACH Transaction (IAT)

The implementations are based on the *NACHA ISO 20022 Tool for pain.002.001.03 (Reject) Nov 2016.xlsx* document.

The example references the *ACH\_ISO\_Mapping\_Assumptions\_Table (Reject).xlsx* table. These tables install with the *nachaISO20022MappingDoc.zip* file.

- [\*\*What the example contains\*\*](#)  
The ACH Rejects example contains these files, schemas, and maps.
- [\*\*How to run the example in Design Studio\*\*](#)  
These steps describe how to run the ACH Reject to ISO 20022 transformation example.
- [\*\*How to run the example in Design Server\*\*](#)  
Run instructions for the ACH ISO20022 Rejects example.
- [\*\*Scenarios\*\*](#)  
The ACH to ISO 20022 and ISO 20022 to ACH example maps show ACH and ISO 20022 files being validated if the files conforms to specific rules, prior to transformation to a pain.002 document.
- [\*\*Assumptions\*\*](#)

---

## What the example contains

The ACH Rejects example contains these files, schemas, and maps.

## Files

---

These files are contained in the example:

- ach\_reject\_ccd\_ct\_input.txt - input to the rjtiso01 map which contains a total of 2 Reject CCD batch entries for Credit Transfer.
- ach\_reject\_ccd\_dd\_input.txt - input to the rjtiso01 map which contains a total of 2 Reject CCD batch entries for Direct Debit.
- ach\_reject\_ppd\_ct\_input.txt - input to the rjtiso01 map which contains a total of 1 Reject PPD batch entries for Credit Transfer.
- ach\_reject\_ppd\_dd\_input.txt - input to the rjtiso01 map which contains a total of 1 Reject PPD batch entries for Direct Debit.
- ach\_reject\_ctx\_ct\_input.txt - input to the rjtiso01 map which contains a total of 2 Reject CTX batch entries for Credit Transfer.
- ach\_reject\_ctx\_dd\_input.txt - input to the rjtiso01 map which contains a total of 2 Reject CTX batch entries for Direct Debit.
- ach\_reject\_iat\_ct\_input.txt - input to the rjtiso01 map which contains a total of 1 Reject IAT batch entries for Credit Transfer.
- ach\_reject\_iat\_dd\_input.txt - input to the rjtiso01 map which contains a total of 1 Reject IAT batch entries for Direct Debit.
- iso\_pain002\_to\_reject\_ccd\_ct\_input.xml - input to the isorj01 map which contains a total of 2 OriginalPaymentAndStatus message level records for Credit Transfer.
- iso\_pain002\_to\_reject\_ccd\_dd\_input.xml - input to the isorj01 map which contains a total of 2 OriginalPaymentAndStatus message level records for Direct Debit.
- iso\_pain002\_to\_reject\_ppd\_ct\_input.xml - input to the isorj01 map which contains a total of 1 OriginalPaymentAndStatus message level records for Credit Transfer.
- iso\_pain002\_to\_reject\_ppd\_dd\_input.xml - input to the isorj01 map which contains a total of 1 OriginalPaymentAndStatus message level records for Direct Debit.
- iso\_pain002\_to\_reject\_ctx\_ct\_input.xml - input to the isorj01 map which contains a total of 2 OriginalPaymentAndStatus message level records for Credit Transfer.
- iso\_pain002\_to\_reject\_ctx\_dd\_input.xml - input to the isorj01 map which contains a total of 2 OriginalPaymentAndStatus message level records for Direct Debit.
- iso\_pain002\_to\_reject\_iat\_ct\_input.xml - input to the isorj01 map which contains a total of 1 OriginalPaymentAndStatus message level records for Credit Transfer.
- iso\_pain002\_to\_reject\_iat\_dd\_input.xml - input to the isorj01 map which contains a total of 1 OriginalPaymentAndStatus message level records for Direct Debit.
- ach\_error\_codes.txt - list of error messages used in the pre-conversion rule verification step.
- rtrn\_reason\_codes.txt - Lists of CH return reason code and their equivalent ISO ExternalStatusReason1 Codes.

## Schemas

---

These schemas are contained in the example:

- pain.002.001.03.xsd - XML schema that represents the ISO 20022 Customer Payment Status Report message document.
- pre\_conv\_rules - type tree that serves as place holder for pre-conversion rules enforced in the transformation maps.
- rtrn\_reason\_codes - type tree that represents the rtrn\_reason\_codes.txt file which lists ACH return reason code and their equivalent ISO ExternalStatusReason1 Codes
- ach\_error\_codes - type tree that represents the ach\_error\_codes.txt file which lists all error codes.
- ach\_v2024 - current version of NACHA type tree.

## Maps

---

These maps are contained in the example:

- rjtniso01 - map used to transform an ACH Reject for Corporate Credit or Debit (CCD), Prearranged Payment and Deposit (PPD), ACH International ACH Transaction (IAT), and Corporate Trade Exchange (CTX) file to an xml Customer Payment Status Report message (pain.002.001.03).
- isorjtn01 - map used to transform an xml Customer Payment Status Report message (pain.002.001.03) to an ACH Reject Corporate Credit or Debit (CCD), Prearranged Payment and Deposit (PPD), ACH International ACH Transaction (IAT) and Corporate Trade Exchange (CTX) file.

## How to run the example in Design Studio

---

These steps describe how to run the ACH Reject to ISO 20022 transformation example.

### About this task

---

Run the map example as follows:

### Procedure

---

1. Use the Design Studio to open the following map source file:
  - a. ACH to ISO 20022
    - achiso\_rjct.mms
  - b. ISO 20022 to ACH
    - isoach\_rjct.mms
2. Build all the following maps in the source file:
  - rjtniso01 (transforms Reject to pain.002.001.03)
  - isorjtn01 (transforms pain.002.001.03 to Reject)
3. Set all compiled maps with a .mmc extension.
4. Define the following directories for those operating systems other than Windows:
  - data
  - maps
  - schemas
5. When the artifacts listed here are copied to operating systems other than Windows, make certain that the FTP settings are set as described in sub-steps a and b:
  - a. FTP settings must be set to TEXT mode for the following input data:
    - ach\_reject\_ccd\_ct\_input.txt
    - ach\_reject\_ccd\_dd\_input.txt
    - ach\_reject\_ppd\_ct\_input.txt
    - ach\_reject\_ppd\_dd\_input.txt
    - ach\_reject\_ctx\_ct\_input.txt
    - ach\_reject\_ctx\_dd\_input.txt
    - ach\_reject\_iat\_ct\_input.txt
    - ach\_reject\_iat\_dd\_input.txt
    - rtrn\_reason\_codes.txt
    - ach\_error\_codes.txt
  - b. FTP settings must be set to BINARY mode for the following:
    - all compiled maps
    - all xml schema files
    - iso\_pain002\_to\_reject\_ccd\_ct\_input.xml
    - iso\_pain002\_to\_reject\_ccd\_dd\_input.xml
    - iso\_pain002\_to\_reject\_ppd\_ct\_input.xml
    - iso\_pain002\_to\_reject\_ppd\_dd\_input.xml
    - iso\_pain002\_to\_reject\_ctx\_ct\_input.xml
    - iso\_pain002\_to\_reject\_ctx\_dd\_input.xml
    - iso\_pain002\_to\_reject\_iat\_ct\_input.xml
    - iso\_pain002\_to\_reject\_iat\_dd\_input.xml
- Note: MVS users, perform both step 6 and step 7. Non-MVS users go directly to step 7.
6. **MVS only:** Edit the following map card setting of the Schema > Type > Metadata to the //DD:ddname where ddname references the XML schema, pain.002.001.03.xsd
  - a. Edit the ddname references on output card #2 (pain.002.001.03.xsd) for the following map:
    - rjtniso01

- b. Edit the ddname references on input card #1 (pain.002.001.03.xsd) for the following maps:
- isorjt01
7. Run the maps listed in step 2 of this procedure.

---

## How to run the example in Design Server

Run instructions for the ACH ISO20022 Rejects example.

### About this task

These steps describe how to run the ACH ISO20022 Rejects example.

### Procedure

1. Import the nachaISO20022Rejects.zip file into the Design Server UI. See the release notes for import instructions.
2. Create a package for the project and build all the maps.
3. Open the nachaISO20022Rejects project in the Design Server UI.
4. Run the following maps:
  - isorjt01
  - rjtiso01
5. View the output. Go to the map Diagram-Result, and hover the cursor over Target, then select View Data.
6. To download the output file, see [How to download output files](#).

---

## Scenarios

The ACH to ISO 20022 and ISO 20022 to ACH example maps show ACH and ISO 20022 files being validated if the files conforms to specific rules, prior to transformation to a pain.002 document.

The rules are defined in the *ACH\_ISO\_Mapping\_Assumptions\_Table (Reject).xlsx* table. This table includes pre-conversion rules needed between ACH to and from ISO pain.002.001.03.

The *ACH\_ISO\_Mapping\_Assumptions\_Table (Direct Debit).xlsx* installs with the pack in the following directory:

<install\_dir>/packs/financial\_payments\_vn.n.n.n/nacha/examples/ach\_iso20022/scenario/assumptions

The table is also available on the Support website: <http://www.ibm.com/support/docview.wss?uid=ibm10716007>

---

## Assumptions

- The example maps enforce the pre-conversion rules and if no rules are violated, the maps proceed to the transformation phase.
- Mapping is done based on the *NACHA ISO 20022 Tool for pain.002 (reject) Nov 2016.xlsx* document.
- The *ACH\_ISO\_Mapping\_Assumptions\_Table (Reject).xlsx* includes implementation specifics on the mapping to and from ACH and ISO pain.002.

---

## Expected results

- The ACH to ISO example maps generate an XML document which passes pain.002.001.03 XML schema validation
- The ISO to ACH example maps generate an ACH file which passes ACH type tree validation.

---

## ACH ISO 20022 Returns example

The ACH ISO 20022 Returns example demonstrates how to transform an ACH file to an ISO 20022 XML document, as well as how to transform an ISO 20022 XML document to an ACH file.

The transformation will be limited to the following Standard Entry Class (SEC) code being transformed into an ISO 20022 Bank To Customer Statement message (camt.053.001.02) Credit Transfer and Direct Debit file.

- Corporate Credit or Debit (CCD)
- Prearranged Payment and Deposit (PPD)
- Corporate Trade Exchange (CTX)

The implementations are based on the *NACHA ISO 20022 Tool for camt.053 (returns) Nov 2016.xlsx* document.

The example references the *ACH\_ISO\_Mapping\_Assumptions\_Table (returns).xlsx* table. These tables install with the nachaISO20022MappingDoc.zip file.

- [What the example contains](#)  
The ACH returns example contains these files, schemas, and maps.
- [How to run the example in Design Studio](#)  
These steps describe how to run the ACH Returns to ISO 20022 transformation example.

- [How to run the example in Design Server](#)

Run instructions for the ACH ISO20022 Returns example.

- [Scenarios](#)

- [Assumptions](#)

The ACH Returns to ISO 20022 transformation example maps enforce pre-conversion rules.

---

## What the example contains

The ACH returns example contains these files, schemas, and maps.

### Files

These files are contained in the example:

- ach\_return\_ccd\_ct\_input.txt - input to the rtniso01 map which contains a total of 2 Returns CCD batch entries for Credit Transfer.
- ach\_return\_ccd\_dd\_input.txt - input to the rtniso01 map which contains a total of 2 Returns CCD batch entries for Direct Debit.
- ach\_return\_ppd\_ct\_input.txt - input to the rtniso01 map which contains a total of 1 Returns PPD batch entries for Credit Transfer.
- ach\_return\_ppd\_dd\_input.txt - input to the rtniso01 map which contains a total of 1 Returns PPD batch entries for Direct Debit.
- ach\_return\_ctx\_ct\_input.txt - input to the rtniso01 map which contains a total of 2 Returns CTX batch entries for Credit Transfer.
- ach\_return\_ctx\_dd\_input.txt - input to the rtniso01 map which contains a total of 2 Returns CTX batch entries for Direct Debit.
- iso\_camt053\_to\_returns\_ccd\_ct\_input.xml - input to the isortn01 map which contains a total of 2 Statement message level records for Credit Transfer.
- iso\_camt053\_to\_returns\_ccd\_dd\_input.xml - input to the isortn01 map which contains a total of 2 Statement message level records for Direct Debit.
- iso\_camt053\_to\_returns\_ppd\_ct\_input.xml - input to the isortn01 map which contains a total of 1 Statement message level records for Credit Transfer.
- iso\_camt053\_to\_returns\_ppd\_dd\_input.xml - input to the isortn01 map which contains a total of 1 Statement message level records for Direct Debit.
- iso\_camt053\_to\_returns\_ctx\_ct\_input.xml - input to the isortn01 map which contains a total of 2 Statement message level records for Credit Transfer.
- iso\_camt053\_to\_returns\_ctx\_dd\_input.xml - input to the isortn01 map which contains a total of 2 Statement message level records for Direct Debit.
- ach\_error\_codes.txt - list of error messages used in the pre-conversion rule verification step.
- rtrn\_reason\_codes.txt - lists of CH return reason code and their equivalent ISO ExternalStatusReason1 Codes.

### Schemas

These schemas are contained in the example:

- camt.053.001.02.xsd - XML schema that represents the ISO 20022 Bank To Customer Statement message document.
- pre\_conv\_rules - type tree that serves as place holder for pre-conversion rules enforced in the transformation maps.
- rtrn\_reason\_codes - type tree that represents the rtrn\_reason\_codes.txt file which lists ACH return reason code and their equivalent ISO ExternalStatusReason1 Codes.
- ach\_error\_codes - type tree that represents the ach\_error\_codes.txt file which lists all error codes.
- ach\_v2024 - current version of NACHA type tree.

### Maps

These maps are contained in the example:

- rtniso01 - map used to transform an ACH Returns for Corporate Credit or Debit (CCD), Prearranged Payment and Deposit (PPD) and Corporate Trade Exchange (CTX) file to an XML Bank To Customer Statement message (camt.053.001.02).
  - isortn01 - map used to transform an XML Bank To Customer Statement message (camt.053.001.02) to an ACH Returns Corporate Credit or Debit (CCD), Prearranged Payment and Deposit (PPD) and Corporate Trade Exchange (CTX) file.
- 

## How to run the example in Design Studio

These steps describe how to run the ACH Returns to ISO 20022 transformation example.

### About this task

Run the map example as follows:

### Procedure

1. Use the Design Studio to open the following map source file:
    - a. ACH to ISO 20022
      - achiso\_rtrn.mms
    - b. ISO 20022 to ACH
      - isoach\_rtrn.mms
  2. Build all the following maps in the source file:
    - rtniso01 (transforms Returns to camt.053.001.02)
    - isortn01 (transforms camt.053.001.02 to Returns)
  3. Set all compiled maps with a .mmc extension.
  4. Define the following directories for those operating systems other than Windows:
    - data
    - maps
    - schemas
  5. When the artifacts listed here are copied to operating systems other than Windows, make certain that the FTP settings are set as described in sub-steps a and b:
    - a. FTP settings must be set to TEXT mode for the following input data:
      - ach\_return\_ccd\_ct\_input.txt
      - ach\_return\_ccd\_dd\_input.txt
      - ach\_return\_ppd\_ct\_input.txt
      - ach\_return\_ppd\_dd\_input.txt
      - ach\_return\_ctx\_ct\_input.txt
      - ach\_return\_ctx\_dd\_input.txt
      - rtrn\_reason\_codes.txt
      - ach\_error\_codes.txt
    - b. FTP settings must be set to BINARY mode for the following:
      - all compiled maps
      - all xml schema files
      - iso\_camt053\_to\_returns\_ccd\_ct\_input.xml
      - iso\_camt053\_to\_returns\_ccd\_dd\_input.xml
      - iso\_camt053\_to\_returns\_ppd\_ct\_input.xml
      - iso\_camt053\_to\_returns\_ppd\_dd\_input.xml
      - iso\_camt053\_to\_returns\_ctx\_ct\_input.xml
      - iso\_camt053\_to\_returns\_ctx\_dd\_input.xml
- Note: MVS users, perform step 6 and step 7. Non-MVS users go directly to step 7.
6. **MVS only:** Edit the following map card setting of the Schema > Type > Metadata to the //DD:ddname where ddname reference the XML schema camt.053.001.02.xsd.
- a. Edit the ddname references on output card #2 (camt.053.001.02.xsd) for the following map:
    - rtniso01
  - b. Edit the ddname references on input card #1 (camt.053.001.02.xsd) for the following map:
    - isortn01
7. Run the maps listed in step 2 of this procedure.

## How to run the example in Design Server

Run instructions for the ACH ISO20022 Returns example.

### About this task

These steps describe how to run the ACH ISO20022 Returns example.

### Procedure

1. Import the nachaISO20022Returns.zip file into the Design Server UI. See the release notes for import instructions.
2. Create a package for the project and build all the maps.
3. Open the nachaISO20022Returns project in the Design Server UI.
4. Run the following maps:
  - isortn01
  - rtniso01
5. View the output. Go to the map Diagram-Result, and hover the cursor over Target, then select View Data.
6. To download the output file, see [How to download output files](#).

### Scenarios

The ACH Returns to ISO 20022 and ISO 20022 to ACH Returns example demonstrates file validation prior to transformation to a camt.053.001.02 document. Validation is dependent upon predefined rules being followed.

The *ACH\_ISO\_Mapping\_Assumptions\_Table (Returns).xlsx* table includes the pre-conversion rules needed for validation between ACH and ISO camt.053.

The *ACH\_ISO\_Mapping\_Assumptions\_Table (Returns).xlsx* installs with the pack in the following directory:

The table is also available on the Support website: <http://www.ibm.com/support/docview.wss?uid=ibm10716007>

### Assumptions

The ACH Returns to ISO 20022 transformation example maps enforce pre-conversion rules.

The following are assumed if no rules are violated:

- The maps proceed to the transformation phase.
- Mapping is based on the *NACHA ISO 20022 Tool for camt.053 (returns) Nov 2016.xlsx* document.

The *ACH\_ISO\_Mapping\_Assumptions\_Table (Returns).xlsx* table includes implementation specifics on the mapping between ACH and ISO camt.053.

## Expected results

---

- The ACH Returns to ISO 20022 example maps generate an XML document that passes camt.053.001.02 XML schema validation.
  - The ISO to ACH example maps generate an ACH file that passes ACH schema validation.
- 

## XML to ACH PPD example

The example map illustrates how information is mapped from a back-office XYZ Trust XML file into the ACH PPD record layouts.

- [\*\*What the example contains\*\*](#)  
The XML to ACH example contains these files, schemas, and maps.
  - [\*\*How to run the example in Design Studio\*\*](#)  
Run the map example as follows:
  - [\*\*How to run the example in Design Server\*\*](#)  
Run instructions for the XML to ACH PPD example.
  - [\*\*Scenarios\*\*](#)  
The sample application consists of a payroll application (a batch of consumer credit entries) and a monthly dues application (a batch of consumer debit entries).
  - [\*\*Assumptions\*\*](#)  
The XML to ACH PPD example has these assumptions.
- 

## What the example contains

The XML to ACH example contains these files, schemas, and maps.

### Files

---

These files are contained in the example:

- xyz\_trust\_file.xml - input to the transformation map which contains payroll transactions and membership dues from two originating customers.

### Schemas

---

These schemas are contained in the example:

- xyz.trst.001.xsd - XML schema describing the format of the XML document which holds information required by the sample originating depository institution to format an ACH PPD file.
- xyz.trst.001 - type tree derived from the sample XML schema called xyz.trst.001.xsd.
- ach\_v2024 - current version of NACHA type tree.

### Maps

---

These maps are contained in the example:

- xml\_to\_ppd - used to convert a non-industry specific XML file to an ACH Prearrange Payment and Deposit Entries (PPD) file.
- 

## How to run the example in Design Studio

Run the map example as follows:

### Procedure

---

1. Use the Design Studio to open the following map source file: `xml_to_ppd.mms`
2. Build the map in the source file, and then run the following transformation map:  
Note: Comments which are associated with the rules are based upon situations which are illustrated by the example.
  - `xml_to_ppd`

## How to run the example in Design Server

Run instructions for the XML to ACH PPD example.

## About this task

---

These steps describe how to run the XML to ACH PPD example.

## Procedure

---

1. Import the nachaXML.zip file into the Design Server UI. See the release notes for import instructions.
  2. Create a package for the project and build all the maps.
  3. Open the nachaXML project in the Design Server UI.
  4. Run the following map:
    - `xml_to_ppd`
  5. View the output. Go to the map Diagram-Result, and hover the cursor over Target, then select View Data.
  6. To download the output file, see [How to download output files](#).
- 

## Scenarios

---

The sample application consists of a payroll application (a batch of consumer credit entries) and a monthly dues application (a batch of consumer debit entries).

Based on the sample applications and certain assumptions, this example illustrates how the PPD ACH records are formatted.

---

## Assumptions

---

The XML to ACH PPD example has these assumptions.

Note: The XML to ACH PPD example relates to specific applications outlined in this documentation. The example is not intended to depict situations that are always appropriate.

XYZ Trust originates ACH payments. Presently, it has two originating customers, BEL, Incorporated, and AM Club.

## Application #1

---

BEL, Incorporated pays its employees monthly by direct deposit. On June 25, BEL, Incorporated prepared an ACH file on its PC software package for delivery to XYZ Trust.

The file contains the information for the pay date of June 27. Three individuals are being paid a total of \$90,000. Payroll information for BEL Incorporated is shown here:

|             |                         |
|-------------|-------------------------|
| NAME        | Donn, John              |
| EMPLOYEE ID | 787-87-8787             |
| NET PAY     | \$30,000.00             |
| FI          | QRS S&L                 |
| RTN         | 43655655                |
| ACT. #      | 34-567 890 (checking)   |
|             |                         |
| NAME        | Rees, Jaycee            |
| EMPLOYEE ID | 133-13-3133             |
| NET PAY     | \$30,000.00             |
| FI          | TUV C.U.                |
| RTN         | 43999999                |
| ACT. #      | 10987654343 (savings)   |
|             |                         |
| NAME        | Ainge, Roy              |
| EMPLOYEE ID | 343-34-3434             |
| NET PAY     | \$30,000.00             |
| FI          | WXY Bank                |
| RTN         | 34513093                |
| ACT. #      | 7 78 0-87 69 (checking) |

## Application #2

---

AM Club collects some of its monthly dues by ACH debit. On June 25, AM Club prepares an ACH file on its PC software package and delivers it to XYZ Trust by modem. The file contains three collections for July dues. The entries total \$1,800.00. The ID number for AM Club is its tax ID: 249999991.

Customer information for AM Club is listed here:

|              |              |
|--------------|--------------|
| NAME         | Connor, Beth |
| MEMBERSHIP # | 5130         |
| DUES         | \$600.00     |
| FI           | WXY Bank     |
| RTN          | 34513093     |

|              |                            |
|--------------|----------------------------|
| ACCT. #      | 31872-76 54 8 (checking)   |
| NAME         | Ossmond, Daniel            |
| MEMBERSHIP # | 5199                       |
| DUES         | \$600.00                   |
| FI           | GHI S&L                    |
| RTN          | 29999999                   |
| ACCT. #      | 98 654 434-3911 (checking) |
|              |                            |
| NAME         | Gregg, Samuel              |
| MEMBERSHIP # | 5135                       |
| DUES         | \$600.00                   |
| FI           | LMN C.U.                   |
| RTN          | 20000000                   |
| ACCT. #      | 675718 0-83-55 (checking)  |

The ID number (Routing and ABA Number) for XYZ Trust is 27111111.

XYZ Trust receives the information from its two Originators and prepares a file for delivery to its ACH Operator, LOL services. The Routing Number for LOL Services is 27322225. The file contains one batch of credits and one batch of debits. Each batch carries three PPD entries.

## ACH Returns and Notification of Change (COR) example

The example map illustrates how a return and notification of change is prepared.

The example application shows the preparation of a file which consists of two batches. The first batch carries a single return entry, and the second batch contains a single notification of change entry.

- [What the example contains](#)  
The ACH Return Notification of Change example contains these files, schemas, and maps.
- [How to run the example in Design Studio](#)  
Run the map example as follows.
- [How to run the example in Design Server](#)  
Run instructions for the ACH Return Notification of Change example.
- [Scenarios](#)  
The original entries are received by the RDFI.
- [Assumptions](#)  
These are the assumptions for the ACH Returns and Notifications of Change (COR) example.

## What the example contains

The ACH Return Notification of Change example contains these files, schemas, and maps.

### Files

This file is contained in the example:

- ach\_ppd\_from\_xyz\_trust.in - input to the transformation map which contains payroll transaction with and membership due from two originating customers.

### Schemas

This schema is contained in the example:

- ach\_v2024 - current version of NACHA type tree.

### Maps

This map is contained in the example:

- ach\_return\_cor - map that creates an ACH Return entry and an ACH Notification of Change (COR) entry from an ACH Prearrange Payment and Deposit Entries (PPD) file.

## How to run the example in Design Studio

Run the map example as follows.

### Procedure

1. Use the Design Studio to open the following map source file: ach\_return\_cor.mms.

- Build the map in the source file, and then run the transformation map.
- Comments that are associated with the rules are based on situations that are illustrated by the example.

- ach\_return\_cor

## How to run the example in Design Server

Run instructions for the ACH Return Notification of Change example.

### About this task

These steps describe how to run the ACH Return Notification of Change example.

### Procedure

- Import the nachaReturnNotificationOfChange.zip file into the Design Server UI. See the release notes for import instructions.
- Create a package for the project and build all the maps.
- Open the nachaReturnNotificationOfChange project in the Design Server UI.
- Run the following map:
  - ach\_return\_cor
- View the output. Go to the map Diagram-Result, and hover the cursor over Target, then select View Data.
- To download the output file, see [How to download output files](#).

### Scenarios

The original entries are received by the RDFI.

However, WXY Bank, Routing Number 34513093, cannot post a \$600 entry for Beth Connor, account # 31872-76 54 8. The reason is because the account does not have sufficient funds to cover the debit (return reason code 'R01').

The DEF Bank also prepares a notification of change. The change notifies XYZ Trust, which is the ODFI, that the DFI Account Number for Roy Ainge is incorrect and must be changed. The Original Entry carried a DFI Account Number of 7780-8769, but the correct DFI Account Number is 80-8769.

### Assumptions

These are the assumptions for the ACH Returns and Notifications of Change (COR) example.

Note: The ACH Returns and Notifications of Change (COR) example relates to specific applications outlined in the documentation provided here. The example is not intended to depict situations that are always appropriate.

The WXY Bank received the following PPD entries from XYZ Trust. XYZ trust has started originating ACH payments and it presently has two originating customers: Application #1 and Application #2.

### Application #1

The AM Club collects some of its monthly dues through ACH debit. On June 25, UVW Club prepared an ACH file on its PC software package and delivered it to XYZ Trust by modem.

The file contains one collection for July dues. The entries total \$600.00. AM Club's ID number is its tax ID - 249999991. AM Club's customer information is listed as follows:

|              |                          |
|--------------|--------------------------|
| NAME         | Connor, Beth             |
| MEMBERSHIP # | 5130                     |
| DUES         | \$600.00                 |
| FI           | WXY Bank                 |
| RTN          | 34513093                 |
| ACCT. #      | 31872-76 54 8 (checking) |

### Application #2

BEL, Incorporated pays its employees monthly by direct deposit. On June 25, BEL, Incorporated prepared an ACH file on its PC software package for delivery to XYZ Trust. The file contains the information for the pay date of June 27.

The individual is being paid a total of \$30,000. Payroll information for BEL is listed below.

|              |             |
|--------------|-------------|
| NAME         | Ainge, Roy  |
| MEMBERSHIP # | 343-34-3434 |
| DUES         | \$30,000.00 |
| FI           | WXY Bank    |
| RTN          | 34513093    |

ACCT. # 7 78 0-87 69 (checking)

The ID number for BEL, Incorporated is a user assigned number: 333333331 The ID (Routing and ABA Number) for XYZ Trust is 27111111.

- WXY Bank responds to ACH PPD entries received, and prepares an ACH file with a batch that contains a single RETURN entry due to insufficient funds. It prepares another batch that contains a Notification of Change (COR) entry.
- The ACH file is delivered to the ACH Operator - TTY Services at WXY Bank which has a Routing Number of 34223222.

## ACH Refused Notification of Change Entry example

The example map shows how a Refused Notification of Change (COR) entry is prepared.

The example application shows the preparation of a file which consists of one batch. The batch carries one Refused Notification of Change entry.

- [What the example contains](#)  
The ACH Refused Notification of Change example contains these files, schemas, and maps.
- [How to run the example in Design Studio](#)  
Run the example map as follows.
- [How to run the example in Design Server](#)  
Run instructions for the ACH Refused Notification of Change example.
- [Scenarios](#)  
The Notification of Change (COR) entry is received by XYZ Trust.
- [Assumptions](#)  
These are the assumptions for the ACH Refused Notification of Change Entry example.

## What the example contains

The ACH Refused Notification of Change example contains these files, schemas, and maps.

### Files

This file is contained in the example:

- ach\_cor.in - input to the transformation map which contains a notification about a payroll transaction and corrected account information.

### Schemas

This schema is contained in the example:

- ach\_v2024 - current version of NACHA metadata.

### Maps

This map is contained in the example:

- ach\_refused\_cor - map that creates an ACH Refused Notification of Change entry from an ACH Notification of Change (COR) entry file.

## How to run the example in Design Studio

Run the example map as follows.

### Procedure

1. Use the Design Studio to open the following map source file: ach\_refused\_cor.mms
2. Build the map in the source file, and then run the transformation map.  
Comments that are associated with the rules are based on situations that are illustrated by the example.

- ach\_refused\_cor

## How to run the example in Design Server

Run instructions for the ACH Refused Notification of Change example.

### About this task

These steps describe how to run the ACH Refused Notification of Change example.

### Procedure

- 
1. Import the nachaRefusedNotificationOfChange.zip file into the Design Server UI. See the release notes for import instructions.
  2. Create a package for the project and build all the maps.
  3. Open the nachaRefusedNotificationOfChange project in the Design Server UI.
  4. Run the following map:
    - ach\_refused\_cor
  5. View the output. Go to the map Diagram-Result, and hover the cursor over Target, then select View Data.
  6. To download the output file, see [How to download output files](#).
- 

## Scenarios

The Notification of Change (COR) entry is received by XYZ Trust.

It contains one batch that carries a single COR entry. The COR entry sent by WXY Bank (the RDFI) sends the wrong Original Entry Trace Number. The Trace Number expected was 2711111000003 but it was sent as 27111110000303. XYZ Trust (the ODFI) receives the Notification of Change and cannot identify the original entry because of the error. The ODFI prepares a Refused Notification of Change.

---

## Assumptions

These are the assumptions for the ACH Refused Notification of Change Entry example.

Note: The ACH Refused Notification of Change Entry example relates to specific applications outlined here, and are not intended to depict situations which are always appropriate.

BEL, Incorporated pays its employees monthly by direct deposit. On June 25, BEL, Incorporated prepared an ACH file on its PC software package for delivery to XYZ Trust. The file contains the information for the pay date of June 27. One individual is being paid a total of \$30,000. Payroll information for BEL Incorporated is listed here:

|             |                         |
|-------------|-------------------------|
| NAME        | Ainge, Roy              |
| EMPLOYEE ID | 343-34-3434             |
| NET PAY     | \$30,000.00             |
| FI          | WXY Bank                |
| RTN         | 34513093                |
| ACCT. #     | 7 78 0-87 69 (checking) |

- The ID number for BEL, Corporation is a user assigned number: 333333331
  - The Routing and ABA Number for XYZ Trust is: 27111111. The Routing Number for its ACH Operator, LOL Services is: 27322225.
  - WXY Bank received the entry from XYZ Trust. The Original Entry carried a DFI Account Number of 7780-8769, but the correct DFI Account Number is 80-8769.
  - WXY Bank determines that the Roy Ainge account number was incorrect and prepares an ACH file with a batch that contains a single Notification of Change (COR) entry.
  - However, the Notification of Change contained an incorrect Original Entry Trace Number. The expected number was 2711111000003 but it was sent as 27111110000303. XYZ Trust was unable to reference the original entry with the incorrect Trace Number and responded with an ACH Refused Notification of Change entry.
- 

## ACH Dishonored Return example

The ACH Dishonored Return example map shows how a dishonored return entry is prepared.

The example application shows the preparation of a file which consists of one batch. The batch carries one dishonored return entry.

- [What the example contains](#)  
The ACH Dishonored Return example contains these files, schemas, and maps.
  - [How to run the example in Design Studio](#)  
Run the map example as follows.
  - [How to run the example in Design Server](#)  
Run instructions for the ACH Dishonored Return example.
  - [Scenarios](#)  
The following describes the scenario depicted in the ACH Dishonored Return Entry example.
  - [Assumptions](#)  
These are the assumptions for the ACH Dishonored Return entry example.
- 

## What the example contains

The ACH Dishonored Return example contains these files, schemas, and maps.

## Files

This file is contained in the example:

- ach\_return.in - input to the transformation map which contains membership due transaction that is being returned due to insufficient funds on the account holder.

## Schemas

---

This schema is contained in the example:

- ach\_v2024 - current version of NACHA type tree

## Maps

---

This map is contained in the example:

- ach\_dishonored\_return - map that creates an ACH Dishonored Return entry from an ACH Return entry file.

## How to run the example in Design Studio

---

Run the map example as follows.

### Procedure

---

1. Use the Design Studio to open the following map source file: ach\_dishonored\_return.mms
2. Build the map in the source file, and then run the transformation map.

Note: Comments that are associated with the rules are based upon situations that are illustrated by the example.

- ach\_dishonored\_return

## How to run the example in Design Server

---

Run instructions for the ACH Dishonored Return example.

### About this task

---

These steps describe how to run the ACH Dishonored Return example.

### Procedure

---

1. Import the nachaDishonoredReturn.zip file into the Design Server UI. See the release notes for import instructions.
2. Create a package for the project and build all the maps.
3. Open the nachaDishonoredReturn project in the Design Server UI.
4. Run the following map:
  - ach\_dishonored\_return
5. View the output. Go to the map Diagram-Result, and hover the cursor over Target, then select View Data.
6. To download the output file, see [How to download output files](#).

## Scenarios

---

The following describes the scenario depicted in the ACH Dishonored Return Entry example.

The return entry is received by XYZ Trust, it contains one batch that carries a single return entry. The return had to be made available to the ODFI by opening of business on June 28 in order to be considered on time. For the purpose of this example, assume that the return was on June 27. However, due to ACH Operator hardware problems, XYZ Trust did not receive the return until June 29, instead of June 28. XYZ Trust is unable to post the return to AM Club's account because AM Club has declared bankruptcy. Thus, XYZ Trust suffers a loss and dishonors the return as untimely.

## Assumptions

---

These are the assumptions for the ACH Dishonored Return entry example.

Note: The ACH Dishonored Return entry example relates to specific applications that are outlined in the documentation provided here, and are not intended to depict situations that are always appropriate.

AM Club collects some of its monthly dues through ACH debit. On June 25, AM Club prepared an ACH file on its PC software package and delivered it to XYZ Trust over a modem. The file contains one collection for July dues. The entry total is \$600.00. AM Club's ID number is its tax ID, which is: 249999991

AM Club's customer information is listed below:

|              |              |
|--------------|--------------|
| NAME         | Connor, Beth |
| MEMBERSHIP # | 5130         |
| DUES         | \$600.00     |
| FI           | WXY Bank     |
| RTN          | 34513093     |

ACT. # 31872-76 54 8 (checking)

- XYZ Trust's ID, Routing and ABA Number, is 27111111. Its ACH Operator, LOL services' Routing Number is 27322225.
- WXY Bank received the following PPD entry from XYZ Trust. WXY Bank determined that Beth Connor's account has insufficient funds to cover the membership dues and sent back an ACH return entry on June 27.
- Due to a hardware problem, the ACH Operator was only able to deliver the ACH return to XYZ Trust on June 29 instead of the required June 28 date. At this point AM Club declared bankruptcy. The loss was, therefore, suffered by XYZ Trust. XYZ Trust then responds to the ACH return entry received with a Dishonored Return entry.

## ACH Contested Dishonored Return Entry example

The example map shows how an ACH contested dishonored return entry is prepared.

The example application shows the preparation of a file that consists of one batch. The batch carries one Contested Dishonored Return entry.

- [What the example contains](#)

The ACH Contested Dishonored Return example contains these files, schemas, and maps.

- [How to run the example in Design Studio](#)

Run the map example as follows:

- [How to run the example in Design Server](#)

Run instructions for the ACH Contested Dishonored Return example.

- [Scenarios](#)

The following scenario is depicted in the ACH Contested Dishonored Return Entry example.

- [Assumptions](#)

In the ACH Contested Dishonored Return Entry example, WXY Bank received the following PPD entries from XYZ Trust, which has started to originate ACH payments from customer AM Club.

## What the example contains

The ACH Contested Dishonored Return example contains these files, schemas, and maps.

### Files

This file is contained in the example:

- ach\_dshnrd\_rtrn.in - This file is the input to the transformation map which contains an ACH Dishonored Return entry.

### Schemas

This schema is contained in the example:

- ach\_v2024 - current version of NACHA type tree.

### Maps

This map is contained in the example:

- ach\_cntstd\_dshnrd\_rtrn - This map creates an ACH Contested Dishonored Return entry from an ACH Dishonored Return entry file.

## How to run the example in Design Studio

Run the map example as follows:

### Procedure

1. Use the Design Studio to open the following map source file: ach\_cntstd\_dshnrd\_rtrn.mms
2. Build the map in the source file, and then run the transformation map.

Note: Comments that are associated with the rules are based on situations that are illustrated by the example.

- ach\_cntstd\_dshnrd\_rtrn

## How to run the example in Design Server

Run instructions for the ACH Contested Dishonored Return example.

### About this task

These steps describe how to run the ACH Contested Dishonored Return example.

## Procedure

---

1. Import the nachaContestedDishonoredReturn.zip file into the Design Server UI. See the release notes for import instructions.
  2. Create a package for the project and build all the maps.
  3. Open the nachaContestedDishonoredReturn project in the Design Server UI.
  4. Run the following map:
    - ach\_cntstd\_dshnd\_rtrn
  5. View the output. Go to the map Diagram-Result, and hover the cursor over Target, then select View Data.
  6. To download the output file, see [How to download output files](#).
- 

## Scenarios

The following scenario is depicted in the ACH Contested Dishonored Return Entry example.

The dishonored return entry is received by WXY Bank. It disputes the fact that the original return was untimely. The return had to be made available to the ODFI by the opening of business on June 28 in order to be considered timely. WXY Bank transmitted the return on June 27, but due to hardware problems at the Receiving ACH Operator, the return was not settled until June 29. Thus, XYZ Trust, the originator of the dishonored return, considers the return to be untimely.

---

## Assumptions

In the ACH Contested Dishonored Return Entry example, WXY Bank received the following PPD entries from XYZ Trust, which has started to originate ACH payments from customer AM Club.

Note: The ACH Contested Dishonored Return entry example relates to specific applications that are outlined in the documentation provided here, and is not intended to depict situations that are always appropriate.

## Application 1

---

AM Club collects some of its monthly dues through use of ACH debit. On June 25, AM Club prepared an ACH file on its PC software package and delivered it to XYZ Trust by using a modem. The file contains one collection for July dues. The entries total \$600.00. AM Club's ID number is its tax ID - 249999991.

|              |                          |
|--------------|--------------------------|
| NAME         | Connor, Beth             |
| MEMBERSHIP # | 5130                     |
| DUES         | \$600.00                 |
| FI           | WXY Bank                 |
| RTN          | 34513093                 |
| ACT. #       | 31872-76 54 8 (checking) |

- XYZ Trust's ID (Routing and ABA Number) is 27111111.
  - WXY Bank received the following PPD entry from XYZ Trust. WXY Bank determined that Beth Connor account has insufficient funds to cover the membership dues and sent back an ACH return entry on June 27.
  - Due to a hardware problem, the ACH Operator was only able to deliver the ACH return to XYZ Trust on June 29 instead of the required June 28 date. At this point, AM Club declared bankruptcy. XYZ Trust suffered the loss and responded to the ACH return entry received with a Dishonored Return entry.
  - WXY Bank received the Dishonored Return entry and disputes the fact that the return was untimely, since it transmitted the return on June 27. Because of hardware problems at the Receiving ACH Operator, the return was not settled until June 29.
  - The ACH Contested Dishonored Return file will be delivered to WXY Bank's ACH Operator - TTY Services which has a Routing Number of 34223222.
- 

## ACH Payment Related Information example

The ACH Payment Related Information example demonstrates how to validate the content of the Payment Related Information field on the Addenda record, if it conforms to NACHA endorsed banking conventions.

The conventions are guides to participants involved in electronic payments as to the type of formats, definitions and implementation recommendations to facilitate processing of payment information electronically through the Automated Clearing House (ACH) Network.

- [What the example contains](#)  
The ACH Payment Related Information (PRI) example contains these files, schemas, and maps.
  - [How to run the example in Design Studio](#)  
Run the map example as follows:
  - [How to run the example in Design Server](#)  
Run instructions for the ACH Payment Related Information example.
  - [Scenarios](#)  
The example map shows an ACH file being validated with regard to the formatting of the Payment Related Information field on the Addenda Records if it conforms to the NACHA endorsed banking conventions.
  - [Assumptions](#)  
These are the assumptions for the examples.
- 

## What the example contains

The ACH Payment Related Information (PRI) example contains these files, schemas, and maps.

## Files

---

These files are contained in the example:

- pri.tst.rslt.001.xsd - metadata that represents the XML output reported by the example PRI validation map.
- ach\_pri\_example.in - input to the achpri01\_validate map which contains a total of 20 ACH batch records with various supported Payment Related Information field formats.
- ach\_error\_codes.txt - a resource file that consists of all error codes and descriptions used by the PRI validation map when generating XML output.

## Schemas

---

This schema is contained in the example:

- pri\_tst\_rslt - type tree that represents the place holder used by the example utility maps for each of the PRI validation results.
- pri\_fmt - type tree that represents different payment related information field formats based on NACHA endorsed banking conventions, and ANSI ASC X12 data segments.
- ach\_v2024 - current version of NACHA type tree.
- ach\_error\_codes - type tree that represents the ach\_error\_codes.txt file which lists all error codes.
- pri.tst.rslt.001.xsd - metadata that represents the XML output reported by the example PRI validation map.

## Maps

---

These maps are contained in the example:

- achpri02\_validate\_ack\_atx - utility map called by main map achpri01\_validate to validate the PRI field of ACH Payment Acknowledgement (ACK) and Financial EDI Acknowledgement (ATX) entries.
- achpri05\_validate\_dne - utility map called by main map achpri01\_validate to validate the PRI field of Death Notification Entry (DNE) entries.
- achpri03\_validate\_ccd - utility map called by main map achpri01\_validate to validate the PRI field of Corporate Credit or Debit Entry (CCD) entries.
- achpri06\_validate\_enr - utility map called by main map achpri01\_validate to validate the PRI field of Automated Enrollment Entry (ENR) entries.
- achpri08\_validate\_web - utility map called by main map achpri01\_validate to validate the PRI field of Internet-Initiated or Mobile Entry (WEB) entries.
- achpri07\_validate\_iat - utility map called by main map achpri01\_validate to validate the PRI field of International ACH Entry (IAT) entries.
- achpri04\_validate\_cie\_ppd - utility map called by main map achpri01\_validate to validate the PRI field of Customer Initiated Entry (CIE) and Prearranged Payment and Deposit Entries (PPD) entries.
- achpri01\_validate - the PRI main map used to validate the Addenda Record's PRI field.

## How to run the example in Design Studio

---

Run the map example as follows:

### Procedure

---

1. Use the Design Studio to open the following map source file: pri\_validation.mms.
2. Build all the maps in the source file and then run the map that is called achpri01\_validate.
3. Set all compiled maps with a .mmc extension.
4. Define the following directories for those operating systems other than Windows:
  - data
  - maps
  - schemas
5. When the artifacts listed here are copied to operating systems other than Windows, make certain that the FTP settings are set as described here.
  - a. FTP settings must be set to TEXT mode for the following input data:
    - ach\_pri\_example.in
    - ach\_error\_codes.txt
  - b. FTP settings must be set to BINARY mode for the following:
    - all compiled maps
    - pri.tst.rslt.001.xsd

## How to run the example in Design Server

---

Run instructions for the ACH Payment Related Information example.

## About this task

---

These steps describe how to run the ACH Payment Related Information example.

## Procedure

---

1. Import the nachaPaymentRelatedInfo.zip file into the Design Server UI. See the release notes for import instructions.
  2. Create a package for the project and build all the maps.
  3. Open the nachaPaymentRelatedInfo project in the Design Server UI.
  4. Run the following map:
    - achpri01\_validate
  5. View the output. Go to the map Diagram-Result, and hover the cursor over Target, then select View Data.
  6. To download the output file, see [How to run the example in Design Server](#).
- 

## Scenarios

---

The example map shows an ACH file being validated with regard to the formatting of the Payment Related Information field on the Addenda Records if it conforms to the NACHA endorsed banking conventions.

---

## Assumptions

---

These are the assumptions for the examples.

- The achpri01\_validate example map processes an ACH file with 20 batch records. The example map goes through each batch record. Depending upon the Standard Entry Class (SEC) codes, the example map verifies the format of the Payment Related Information field on the Addenda Record.
- For ACH Payment Acknowledgement (ACK) and Financial EDI Acknowledgement (ATX) entries, the Payment Related Information field must contain an ANSI version release 5010 ASC X12 REF (Reference) data segment.
- For Corporate Credit or Debit Entry (CCD) entries, Payment Related Information field must contain an ANSI ASC X12 data segment, or NACHA endorsed banking conventions (that is Tax Payment, Third-Party Tax Payments, or Child Support).
- For Corporate Credit or Debit Entry (CCD) entries that are used for Healthcare EFT transactions, the Company Entry Description field on the Batch Record has a value that is set to HCCLAIMPMT. Also, the Payment Related Information field must contain an ANSI version release 5010 ASC X12 TRN (Re-association Trace Number) data segment.
- For Corporate Credit or Debit Entry (CCD) entries that are used for Tax Payment, the Payment Related Information field must contain a variant of the ANSI version release 3060 ASC X12 TXP (Tax Payment) data segment. For more information, see the NACHA endorsed, *Tax Payment (TXP) Banking Convention* implementation guide. This implementation guide is available at the following website: <https://www.nacha.org/bankingconvention>
- For Corporate Credit or Debit Entry (CCD) entries that are used for Third-Party Tax Payment, the Payment Related Information field must contain an ASC X12 TPP (Third-Party Tax Payment) data segment. For more information, see the NACHA endorsed, *Third-Party Tax Payment (TPP) Banking Convention* implementation guide. This implementation guide is available on the following website: <https://www.nacha.org/bankingconvention>
- For Corporate Credit or Debit Entry (CCD) entries that are used for Child Support, the Payment Related Information field must contain a variant of ANSI version release 4010 ASC X12 DED (Child Support) data segment. For more information, see the NACHA endorsed, *Child Support (DED) Banking Convention* implementation guide. This implementation guide is available on the following website: <https://www.nacha.org/bankingconvention>
- For Customer Initiated Entry (CIE), Pre-arranged Payment and Deposit Entries (PPD), *Internet-Initiated/Mobile Entries* (WEB debit entries), the Payment Related Information field must contain ANSI ASC X12 data segments.
- For *Internet-Initiated/Mobile Entries* (WEB credit entries), the Payment Related Information field content does not have any formatting requirements.
- For Death Notification Entries (DNE) entries, the Payment Related Information field must contain one of either of the following two NACHA endorsed banking conventions.
  - DATE OF DEATH\*MMDDYY\*CUSTOMERSSN\*#####\*AMOUNT\*\$\$\$\$.cc~
  - DATE OF DEATH\*MMDDYY\*CUSTOMERSSN\*#####\*AMOUNT\*\$\$\$\$.cc~
- In Automated Enrollment Entry (ENR) entries, the Payment Related Information field must contain one of either of the following two NACHA endorsed banking conventions.
  - TRANSACTION CODE (2 CHARS)\*RECEIVING DFI IDENTIFICATION NUMBER (8 CHARS)\*CHECK DIGIT (1 CHAR)\*DFI ACCOUNT NUMBER (MAX 17 CHARS)\*INDIVIDUAL ID NUMBER OR ID NUMBER (9 CHARS)\*INDIVIDUAL SURNAME OR COMPANY NAME(MAX 15 CHARS)\*INDIVIDUAL FIRST NAME OR COMPANY NAME(MAX 7 CHAR)\*REPRESENTATIVE PAYEE INDICATOR/ENROLLEE CLASSIFICATION CODE(1 CHAR)\
  - TRANSACTION CODE (2 CHARS)\*RECEIVING DFI IDENTIFICATION NUMBER (8 CHARS)\*CHECK DIGIT (1 CHAR)\*DFI ACCOUNT NUMBER (MAX 17 CHARS)\*INDIVIDUAL ID NUMBER OR ID NUMBER (9 CHARS)\*INDIVIDUAL SURNAME OR COMPANY NAME(MAX 15 CHARS)\*INDIVIDUAL FIRST NAME OR COMPANY NAME(MAX 7 CHAR)\*REPRESENTATIVE PAYEE INDICATOR/ENROLLEE CLASSIFICATION CODE(1 CHAR)~
- For International ACH Entry (IAT) entries with the Transaction Type Code field within the First IAT Addenda Record. If this field contains ARC, BOC, or RCK, the Payment Related Information field must contain either of the following NACHA endorsed banking conventions:
  - CHECK SERIAL NUMBER
  - CHECK SERIAL NUMBER~
- For International ACH Entry (IAT) entries with the Transaction Type Code field within the First IAT Addenda Record. If this field contains POP, the Payment Related Information field must contain one of either of the following two NACHA endorsed banking conventions.
  - CHECK SERIAL NUMBER (MAX 9 CHARS)\*TERMINAL CITY (MAX 4 CHARS)\*TERMINAL STATE/FOREIGN COUNTRY (2 CHARS)\
  - CHECK SERIAL NUMBER (MAX 9 CHARS)\*TERMINAL CITY (MAX 4 CHARS)\*TERMINAL STATE/FOREIGN COUNTRY (2 CHARS)~
- For International ACH Entry (IAT) entries with the Transaction Type Code field within the First IAT Addenda Record. If this field contains MTE, POS, or SHR, the Payment Related Information field must contain one of the following two NACHA endorsed banking conventions.
  - TERMINAL IDENTIFICATION CODE(MAX 6 CHARS)\*TERMINAL LOCATION (MAX 27 CHARS)\*TERMINAL CITY (MAX 15 CHARS)\*TERMINAL STATE or FOREIGN COUNTRY (2 CHARS)\
  - TERMINAL IDENTIFICATION CODE(MAX 6 CHARS)\*TERMINAL LOCATION (MAX 27 CHARS)\*TERMINAL CITY (MAX 15 CHARS)\*TERMINAL STATE or FOREIGN COUNTRY (2 CHARS)~
- The example map supports only the following SEC codes, ACK, ATX, CCD, CIE, PPD, DNE, ENR, IAT, and WEB entries.
- The example map requires an input ACH file that passes the ACH 2024 type tree validation.
- The achpri01\_validate example map generates an XML output document that consists of the following information about the ACH file.

- File record information
  - Immediate Origin
  - Immediate Origin Name
  - File Creation Date
  - File Creation Time
  - File ID Modifier
  - File Entry / Addenda Count
  - File Total Debit Entry Dollar Amount
  - File Total Credit Entry Dollar Amount
  - File Batch Count
  - File Date / Time Processed
- Batch record information
  - Batch Number
  - Originating DFI Id
  - Company Name
  - Company Id
  - Effective Entry Data
  - Batch Entry/Addenda Count
  - Batch Total Debit Entry Dollar Amount
  - Batch Total Credit entry Dollar Amount
- Entry detail record information
  - Receiving DFI ID
  - Receiving Company Name/Id Number
  - Trace Number
  - ACH (SEC) Entry Code
  - ACH (SEC) Entry Is Supported Flag (true or false)
- Addenda Record Information
  - Entry Detail Sequence Number
  - Addenda Sequence Number
  - Payment Related Information (PRI)
- PRI Validation Information
  - Reason of Failure
  - Status (PASS or FAIL)
- Out of the 20 batch records contained in the input ACH file (ach\_pri\_example.in), the example map detects two batch records with a Payment Related Information field that is not compliant with NACHA endorsed banking conventions. The two batch records are listed here:
  - Batch Number: 4444442, due to a missing TRN data segment on the Payment Related Information field on a Healthcare EFT CCD entry.
  - Batch Number: 4444445, due to a mismatch between the value that is set on the amount field of the Entry Detail Record and the TPP04 amount data element on the Payment Related Information field.

## ACH Return Entries Report example

The example map shows how ACH return entries are extracted from an ACH file. The return entries are reported in categories that are returned due to administrative, unauthorized reasons and all other reasons (except for returned RCK entries).

The example application shows an ACH file input that has two batch entries that contain CCD return entries, and two batch entries that contain BOC return entries.

- [What the example contains](#)  
The ACH Return Entries Report example contains these files, schemas, and maps.
- [How to run the example in Design Studio](#)  
This documentation describes how to run the ACH Return Entries Report example.
- [How to run the example in Design Server](#)  
Run instructions for the ACH Returned Entries Report example.
- [Example scenario](#)  
The example map shows ACH Return entries that are extracted from an ACH file and reported per category.
- [Example assumptions](#)  
The ACH Return Entries Report example contains four applications.

## What the example contains

The ACH Return Entries Report example contains these files, schemas, and maps.

### Files

This file is contained in the example:

- ach\_rtrn\_report\_example.in - input to the map which contains ACH Returns of both corporate and consumer entities.

### Schemas

These schemas are contained in the example:

- ach\_v2024 - current version of NACHA type tree.
- ach\_rtrn\_csv - metadata describing the format of an arbitrary report generated by the map.

## Maps

---

This map is contained in the example:

- ach\_rtrn\_report - map that creates distinct reports for ACH entries returned due to unauthorized, administrative, and other reasons from an ACH Return entry file.
- 

## How to run the example in Design Studio

This documentation describes how to run the ACH Return Entries Report example.

### About this task

---

Follow these procedures to run the map example:

### Procedure

---

1. Use the Design Studio to open the following map source file: ach\_rtrn\_report.mms
2. Build the map in the source file, and then run the transformation map.

Note: Comments that are associated with the rules are based on situations that are illustrated by the example.

- achrtn01\_reports
- 

## How to run the example in Design Server

Run instructions for the ACH Returned Entries Report example.

### About this task

---

These steps describe how to run the ACH Returned Entries Report example.

### Procedure

---

1. Import the nachaReturnedEntriesReport.zip file into the Design Server UI. See the release notes for import instructions.
  2. Create a package for the project and build all the maps.
  3. Open the nachaReturnedEntriesReport project in the Design Server UI.
  4. Run the following map:
    - achrtn01\_reports
  5. View the output. Go to the map Diagram-Result, and hover the cursor over Target, then select View Data.
  6. To download the output file, see [How to download output files](#).
- 

## Example scenario

The example map shows ACH Return entries that are extracted from an ACH file and reported per category.

These categories are reported:

- Administrative (with reason codes R02, R03, R04).
  - Unauthorized (with reason codes R05, R07, R10, R29, R51)
  - Others (all other reason codes except for returned RCK entries, that is, R50, R51, R52, R53). The reports are formatted by using comma-separated values (CSV).
- 

## Example assumptions

The ACH Return Entries Report example contains four applications.

- [Application 1](#)  
COC Export (tax ID: 1006999991) collects corporate receivables from the trading partners that are shown here.
  - [Application 2](#)  
NWNW Import (tax id : 1394440909) collects corporate donations from trading partners.
  - [Application 3](#)  
ACE University (tax ID: 1889383922) collects tuition fees from students and prepares an ACH file.
  - [Application 4](#)  
WOW Gym (tax ID: 1129988333) collects monthly dues from members and prepares an ACH file.
- 

## Application 1

COC Export (tax ID: 1006999991) collects corporate receivables from the trading partners that are shown here.

An ACH file is prepared and sent to COC Export's financial institution, DDD Trust (routing number: 839289911). DDD Trust receives the information and prepares the ACH file for delivery to its ACH Operator (III Services).

|                    |                   |
|--------------------|-------------------|
| NAME:              | Foo Inc.          |
| TRADING PARTNER ID | 475784R           |
| DUES               | \$500.00          |
| FI                 | EEE Bank          |
| RTN                | 14300390          |
| ACCT #             | 49857-80494-33397 |
| NAME               | Mr West LLC       |
| TRADING PARTNER ID | 384107R           |
| DUES               | \$500.00          |
| FI                 | EEE Bank          |
| RTN                | 14300390          |
| ACCT #             | 04942-94873-92843 |

Original entries are received by the RDFI, but EEE Bank cannot post the \$500.00 entry for Foo Inc's account because of return reason code 'R05' (Unauthorized Debit to Consumer Account Using Corporate SEC Code).

EEE Bank also failed to post the \$500.00 for Mr West LLC's account because of return reason code 'R14' (Representative Payee Deceased or Unable to Continue in That Capacity).

---

## Application 2

NWNW Import (tax id : 1394440909) collects corporate donations from trading partners.

An ACH file is prepared and sent to NWNW Import's financial institution, ZZZ Trust (routing number : 012344110). ZZZ Trust receives the information and prepares the ACH file for delivery to its ACH Operator (IV Services).

|                    |                   |
|--------------------|-------------------|
| NAME               | GGG Limited       |
| TRADING PARTNER ID | G933984           |
| DUES               | \$500.00          |
| FI                 | EEE Bank          |
| RTN                | 14300390          |
| ACCT #             | 38738-93838-34747 |
| NAME               | HOGS Meat Inc.    |
| TRADING PARTNER ID | G983988           |
| DUES               | \$500.00          |
| FI                 | EEE Bank          |
| RTN                | 14300390          |
| ACCT #             | 56533-00930-53636 |

Original entries are received by the RDFI, but EEE Bank cannot post the \$500.00 entry for GGG Limited's account because of the return reason code 'R07' (Authorization Revoked by Customer).

EEE Bank also failed to post the \$500.00 to the Hogs Meat Inc. account because of the return reason code 'R03' (No Account/Unable to Locate Account).

---

## Application 3

ACE University (tax ID: 1889383922) collects tuition fees from students and prepares an ACH file.

This ACH file is sent to ACE University's financial institution, SOS Trust (routing number: 039042217). SOS Trust receives the information and prepares the ACH file for delivery to its ACH Operator (VI Services).

|            |                   |
|------------|-------------------|
| NAME       | Fisher, Timothy   |
| STUDENT ID | 15309384-MS       |
| DUES       | \$500.00          |
| FI         | EEE Bank          |
| RTN        | 14300390          |
| ACCT #     | 33339-84844-88859 |
| NAME       | Hooper, Edmond    |
| STUDENT ID | 49349875-CS       |
| DUES       | \$500.00          |
| FI         | EEE Bank          |
| RTN        | 14300390          |
| ACCT #     | 66637-00933-55598 |

Original entries are received by the RDFI, but EEE Bank cannot post the \$500.00 entry to the Timothy Fisher account, because of return reason code 'R10' (Customer Advises Not Authorized, Improper, Ineligible, or part of an Incomplete Transaction).

EEE Bank also failed to post the \$500.00 for Edmond Hooper's account because of return reason code 'R04' (Invalid Account Number)

## Application 4

WOW Gym (tax ID: 1129988333) collects monthly dues from members and prepares an ACH file.

The ACH file is sent to WOW Gym's financial institution, PPP Trust (routing number: 112300478). PPP Trust receives the information and prepares the ACH file for delivery to its ACH Operator (VII Services).

|           |                   |
|-----------|-------------------|
| NAME      | McPearson, Jessie |
| MEMBER ID | M39987432         |
| DUES      | \$500.00          |
| FI        | EEE Bank          |
| RTN       | 14300390          |
| ACCT #    | 63636-18273-03287 |
| NAME      | Oneil, Sharon     |
| MEMBER ID | Y40984303         |
| DUES      | \$500.00          |
| FI        | EEE Bank          |
| RTN       | 14300390          |
| ACCT #    | 87490-29743-84787 |

Original entries are received by the RDFI, but EEE Bank cannot post the \$500.00 entry for Jessie McPearson's account because of the return reason code 'R02' (Account Closed).

EEE Bank also failed to post the \$500.00 for Sharon Oneil's account because of the return reason code 'R01' (Insufficient Funds).

## ACH Validation example

The ACH Validation example demonstrates how to validate an ACH file if it conforms to NACHA Operating Rules. The base implementations are derived from the ACH type treerules and are not designed to replace the use of the ACH type tree.

The ACH validation example can be used to further investigate ACH type tree failures.

- [What the example contains](#)  
The ACH Validation example contains these files, schemas, and maps.
- [How to run the example in Design Studio](#)  
The ACH validation example demonstrates how to validate whether an ACH file conforms to NACHA operating rules.
- [How to run the example in Design Server](#)  
Run instructions for the ACH Validation example.
- [Scenarios](#)  
The ACH validation example demonstrates how to validate whether an ACH file conforms to NACHA operating rules.
- [Assumptions](#)  
These are the assumptions for the validation scenario.

## What the example contains

The ACH Validation example contains these files, schemas, and maps.

### Files

These files are contained in the example:

- ach\_sec\_example.in - input to the achval01 map which contains a total of 13 supported ACH Standard Entry Class codes.
- ach\_config.xml - input to the achval01 map which contains the ACH record/field properties, and attributes to configure validation and reporting.
- ach\_error\_codes.txt - a resource file that consists of all error codes and descriptions used by the maps when generating XML output.

### Schemas

These schemas are contained in the example:

- ach\_v2024 - current version of NACHA type tree.
- ach\_v2024\_sfmt - current version of NACHA sfmt type tree.
- ach\_config.xsd - XML schema that represents the ACH Standard Entry Class records. Also contains user configurable property settings. In this file you should:
  - Treat required fields as optional.
  - Generate compliance report for exceptions only.
- ach\_parse\_info.xsd - XML schema that is used to enforce validation rules on the ACH file.
- ach\_validation\_report.xsd - XML schema that represents the layout of the validation report.
- ach\_generic - metadata that represents the ACH records which are valid, out-of-sequence, and unknown ACH record types.

- ach\_error\_codes - metadata that represents the ach\_error\_codes.txt file which lists all error codes.

## Maps

---

These maps are contained in the example:

- achval01 - the main map used to validate an ACH File.
- achval02 - utility map called by main map which handles unknown ACH record types, and ACH records that are out-of-sequence.
- achval03 - utility map called by main map to generate a report of ACH records and fields which have validation exceptions.
- achval04 - utility map called by main map to generate a report of duplicate ACH records.

---

## How to run the example in Design Studio

The ACH validation example demonstrates how to validate whether an ACH file conforms to NACHA operating rules.

### About this task

---

The base implementations are based on the ACH type tree rules and are not designed to replace the use of the ACH type tree. This ACH validation example can be used to further investigate any ACH type tree failures.

### Procedure

---

1. Use the Design Studio to open the following map source file: ach\_validation.mms
2. Build all maps that are contained in the source file. Run the achval01 map.
3. Set all compiled maps with the .mmc extension.
4. Define the following directories for operating systems other than Windows:
  - data
  - maps
  - schemas
5. When the artifacts listed here are copied to operating systems other than Windows, make certain that the FTP settings are set as described in a and b:
  - a. Set FTP settings to TEXT mode for the following input data:
    - ach\_sec\_example.in
    - ach\_error\_codes.txt
  - b. Set FTP settings to BINARY mode for the following items:
    - All compiled maps
    - All XML schema files
    - ach\_config.xml
6. For MVS, edit the following map card setting of the Schema->Type->Metadata to //DD:ddname where ddname references the XML schema ach\_config.xsd
  - a. achval01asf
    - input card #2 (ach\_config.xsd)
    - output card #1 (ach\_parse\_info.xsd)
  - b. achval02
    - output card #1 (ach\_parse\_info.xsd)
  - c. achval03
    - input card #1 (ach\_validation\_report.xsd)
    - output card #1 (ach\_validation\_report.xsd)

---

## How to run the example in Design Server

Run instructions for the ACH Validation example.

### About this task

---

These steps describe how to run the ACH Validation example.

### Procedure

---

1. Import the nachaValidation.zip file into the Design Server UI. See the release notes for import instructions.
2. Create a package for the project and build all the maps.
3. Open the nachaValidation project in the Design Server UI.
4. Run the following map:
  - achval01
5. View the output. Go to the map Diagram-Result, and hover the cursor over Target, then select View Data.
6. To download the output file, see [How to download output files](#).

---

## Scenarios

The ACH validation example demonstrates how to validate whether an ACH file conforms to NACHA operating rules.

## Assumptions

These are the assumptions for the validation scenario.

- The achval01 example map processes an ACH file with 13 batch records. The example map goes through each ACH record and verifies the format of the File Header, Batch Header, Entry Detail, Addenda, Batch Control and File Control record formats.

The input file contains the following:

- Standard Entry Class (SEC)
- CCD (Corporate Credit or Debit)
- CIE (Customer Initiated Entry)
- CTX (Corporate Trade Exchange)
- MTE (Machine Transfer Entry)
- POS (Point of Sale)
- PPD (Pre-arrange Payment and Deposit Entry)
- RCK (Re-presented Check Entry)
- SHR (Shared Network Transaction)
- TEL (Telephone-Initiated Entry)
- WEB (Internet-Initiated/Mobile Entry)

- The configuration file ach\_config.xml can be edited for these actions:

- Treat the required field as optional. Default is 'YES.'
- Generate a report for ACH fields that have exceptions. Default is 'YES.'
- Enforce validation on an ACH field. Default is 'true.'

Note: Note: Keeping the rest of the settings is recommended, since they are used internally by the maps to process ACH records as defined in the NACHA rule book.

- The achval01 example map generates an XML output document that consists of the following information about the ACH file:

1. ACH record info

- name
- value
- line number

2. ACH field info

- name
- value

3. ACH configuration info

- column number
- length
- attribute type
- validation flag
- requirement attribute

4. Exception info

- description of reason of failure
- status (Pass or Fail)

Note: Of the 13 batch records contained in the input ACH file (ach\_sec\_example.in), the example map detects the last batch record with a WEB SEC code which is noncompliant with NACHA Operating rules. The invalid batch record is listed here:

- Batch Number: 2222800, due to a mismatch of the Company Identification field between the value set on the Batch Header record and Batch Control record.

## How to download output files

Download the output files.

### About this task

To download the output file, go to **Files** in the Design Server UI and download the desired file using these steps:

### Procedure

1. Click the output Card Settings you want to download.
2. From the Command text box, copy the name of the file that you want to download. Make certain that you copy only the file name.  
For example, if the Command text box displays:

```
data/ach_contested_dishonored_return.out
```

copy only:

```
ach_contested_dishonored_return.out
```

3. From Files, click the New icon to open the New File dialog box.
4. Paste the file name in the Name field.
5. Click the Folder field drop down and select data.
6. Click Ok.
7. Build and run the map.
8. Go to Files and select Download to view the data.

# IBM Sterling B2B Integrator examples

See IBM Sterling B2B Integrator for a complete description of these examples, including deployment instructions.

## Related concepts

- [IBM Sterling B2B Integrator](#)

## IBM Sterling B2B Integrator

In IBM Sterling B2B Integrator, a service is a resource that you can configure to carry out an activity.

A typical activity scenario is one where ACH files must be sent to a trading partner. In preparation for this, the data must be enveloped in order to provide identifying batch and interchange data. The IBM Sterling B2B Integrator business process calls the ACH Envelope process to perform enveloping services, while the ACH Deenvelope process provides de-enveloping services.

A document envelope consists of control information that enables organizations to effectively exchange documents. The document envelope takes the original document, assigns a control number, and packages header and trailer information with it prior to submitting it to a trading partner. Creating document envelopes are necessary if you want to do electronic exchange of data with your trading partners.

The pack examples for IBM Sterling B2B Integrator demonstrate the following:

- Creates fully enveloped ACH outbound messages.
- Processes inbound ACH messages.
- Uses Transformation Extender maps for data transformation.

Transformation Extender maps can run within the IBM Sterling B2B Integrator services on some, but not all, versions of the Transformation Extender base product. See the product release notes for those versions of the base product that support IBM Sterling B2B Integrator.

- [Directory structure of the IBM Sterling B2B Integrator examples](#)

The examples for use of IBM Sterling B2B Integrator with the NACHA component include transformation maps, corresponding type trees, test data, and exported IBM Sterling B2B Integrator artifacts.

- [ACH Deenvelope CTX example](#)

The inbound\_ach\_ctx.mms file is the map source for this example.

- [ACH Envelope CTX example](#)

The outbound\_ach\_ctx.mms file is the map source for this example.

- [ACH Deenvelope DNE example](#)

The inbound\_ach\_dne.mms file is the map source for this example.

- [ACH Envelope DNE example](#)

The outbound\_ach\_dne.mms file is the map source for this example.

- [Deploying and executing the IBM Sterling B2B Integrator examples](#)

These procedures describe how to run the ACH Deenvelope and ACH Envelope services examples from Transformation Extender, and from IBM Sterling B2B Integrator.

- [Design considerations of the examples](#)

The documentation provided here describes design features and concepts that were taken into account when building the example transformation maps invoked by the ACH Deenvelope and ACH Envelope services

## Directory structure of the IBM Sterling B2B Integrator examples

The examples for use of IBM Sterling B2B Integrator with the NACHA component include transformation maps, corresponding type trees, test data, and exported IBM Sterling B2B Integrator artifacts.

The IBM Sterling B2B Integrator artifacts include trading partner envelopes, business processes and schemas that are relevant when executing the Transformation Extender map within IBM Sterling B2B Integrator.

The table shows the directory structure used for processing ACH CTX data using both IBM Sterling B2B Integrator ACH Deenvelope and ACH Envelope services.

Table 1. Directory structure for the ACH Deenvelope service examples

| Directory structure                             | Description                                                      |
|-------------------------------------------------|------------------------------------------------------------------|
| ./examples/sterling/services/ach_deenvelope/ctx | Contains the ACH Deenvelope service CTX example.                 |
| ./data                                          | Placeholder for input test data.                                 |
| ./maps                                          | Placeholder for the transformation maps.                         |
| ./platform_support                              | Placeholder for IBM Sterling B2B Integrator import(.xml) files.  |
| ./examples/sterling/services/ach_deenvelope/dne | Contains the ACH Deenvelope service DNE example.                 |
| ./data                                          | Placeholder for input test data.                                 |
| ./maps                                          | Placeholder for the transformation maps.                         |
| ./platform_support                              | Placeholder for IBM Sterling B2B Integrator import (.xml) files. |

Table 2. Directory structure for the ACH Envelope service examples

| Directory structure                          | Description                                    |
|----------------------------------------------|------------------------------------------------|
| /examples/sterling/services/ach_envelope/ctx | Contains the ACH Envelope service CTX example. |
| ./data                                       | Placeholder for input test data.               |

| Directory structure                          | Description                                                      |
|----------------------------------------------|------------------------------------------------------------------|
| ..maps                                       | Placeholder for the transformation maps.                         |
| ..platform_support                           | Placeholder for IBM Sterling B2B Integrator import (.xml) files. |
| /examples/sterling/services/ach_envelope/dne | Contains the ACH Envelope service DNE example.                   |
| ..data                                       | Placeholder for the input test data.                             |
| ..maps                                       | Placeholder for the transformation maps.                         |
| ..platform_support                           | Placeholder for IBM Sterling B2B Integrator import (.xml) files. |

Table 3. CTX and DNE trees and schemas

| Directory structure                 | Description                                                                                           |
|-------------------------------------|-------------------------------------------------------------------------------------------------------|
| /examples/sterling/services/schemas | Placeholder for the schemas used to represent IBM Sterling B2B Integrator proprietary ACH xml format. |
| /examples/sterling/services/trees   | Placeholder for the type trees used in the examples.                                                  |

## ACH Deenvelope CTX example

The inbound\_ach\_ctx.mms file is the map source for this example.

The map source installs into the following directory location:

```
<install_dir>/packs/financial_payments_vn.n.n.n/nacha/examples/sterling/services/ach_deenvelope/ctx/maps
```

The following maps are contained in this map source:

- x12\_ra\_to\_xml\_ctx - This map transforms an X12 5010X218 820 Payment/Order Remittance Advice (RA) file to the IBM Sterling B2B Integrator XML format that represents CTX record format. This XML format is based from custom IBM Sterling B2B Integrator transformation maps.
- x12\_ra\_to\_flat\_rpt - This map is a post-processing map that transforms an X12 5010X218 820 Payment Order/Remittance Advice (RA) file to an arbitrary back office flat file which corresponds to a payment order report.
- x12\_ra\_to\_atx\_pri - This map extracts, from the X12 5010X218 820 Payment Order/Remittance Advice (RA) file, the main REF segment and populates this as the Payment Related Information of the addenda record of the ATX acknowledgement entry.
- xml\_ctx\_to\_csv\_ctx - This map converts an ACH CTX message in XML format into a comma-separated value file format. This XML format is based from the ACH service and is included as part of the process data of the IBM Sterling B2B Integrator.

XML schemas are generated to represent the IBM Sterling B2B Integrator ACH XML format.

Note: The NACHA component of the Pack for Financial Payments is delivered with XML schemas to represent the IBM Sterling B2B Integrator proprietary XML format. See [ACH service XML schemas](#).

## ACH Envelope CTX example

The outbound\_ach\_ctx.mms file is the map source for this example.

The maps source installs into the following directory location:

```
<install_dir>/packs/financial_payments_vn.n.n.n/nacha/examples/sterling/services/ach_envelope/ctx/maps
```

The following map is contained in this map source:

- x12\_ra\_to\_xml\_ctx - This map transforms an X12 5010X218 820 Payment Order/Remittance Advice (RA) file to the IBM Sterling B2B Integrator proprietary XML format that represents CTX record format. This XML format is based on custom IBM Sterling B2B Integrator maps.

Note: This map is the same as the map used in the ACH Deenvelope example, but the purpose of its use in the ACH envelope and deenvelope services are different. This map is used for validation in the ACH Deenvelope service, while for the ACH Envelope service, it is used for transformation.

## ACH Deenvelope DNE example

The inbound\_ach\_dne.mms file is the map source for this example.

The map source installs into the following directory location:

```
<install_dir>/packs/financial_payments_vn.n.n.n/nacha/examples/sterling/services/ach_deenvelope/dne/maps
```

The following files are contained in this map source:

- inbound\_dne\_pri\_to\_xml - This map is used to transform payment related information (PRI) from an ACH DNE addenda record into an XML format. The output of this map will be appended to the IBM Sterling B2B Integrator proprietary XML format of the ACH message, as translated payment related information of the addenda records.
- xml\_dne\_to\_dne\_flat - This map is used to transform proprietary, IBM Sterling B2B Integrator XML format, to an arbitrary back office DNE flat file with fixed length positional elements, that corresponds to an ACH DNE addenda record format.
- create\_dne\_pri - This miscellaneous map is used to generate payment information for a ACH DNE addenda record.
- validate\_dne\_pri - This is a miscellaneous map that is used to validate a file consisting of payment related information of an ACH DNE addenda record.

## ACH Envelope DNE example

The outbound\_ach\_dne.mms file is the map source for this example.

The map source installs into the following directory location:

<install\_dir>/packs/financial\_payments\_vn.n.n.n/nacha/examples/sterling/services/ach\_envelope/dne/maps

The following maps are contained in this map source:

- **outbound\_dne\_entryDtls** - This map is used to extract an arbitrary back office ACH DNE flat file with fixed length positional elements that corresponds to an ACH DNE entry detail record format. The back-office DNE record excludes the following fields: **record type code**, **AddendaRecordIndicator**, and **TraceNumber**. These fields are populated by the ACH Envelope service.
- **outbound\_dne\_addenda** - This map is used to extract an arbitrary back office ACH DNE flat file with fixed length positional elements that corresponds to an ACH DNE addenda record format. The back-office DNE record excludes the following fields: **record type code**, **addenda sequence number**, and **entry detail sequence number**. These fields are populated by the ACH Envelope service.
- **create\_dne\_flat** - This is a miscellaneous map that is used to generate an arbitrary back office DNE flat file with fixed length positional elements.
- **validate\_dne\_flat** - This is a miscellaneous map that is used to validate an arbitrary back office DNE flat file with fixed length positional elements.

## Deploying and executing the IBM Sterling B2B Integrator examples

These procedures describe how to run the ACH Deenvelope and ACH Envelope services examples from Transformation Extender, and from IBM Sterling B2B Integrator.

The example maps call the IBM Sterling B2B Integrator adapter to write values to IBM Sterling B2B Integrator tables. Because of this, the map fails if you attempt to run the map outside of IBM Sterling B2B Integrator. You can, however, test the map from Design Studio by using the Run on Sterling B2B Integrator option.

Design Studio preferences must be configured to connect to the IBM Sterling B2B Integrator. See the Transformation Extender Map Designer documentation for more information.

- [Deploying the example maps to IBM Sterling B2B Integrator](#)

Maps are deployed using either Design Studio, or through use of the IBM Sterling B2B Integrator.

- [Running the ACH service examples](#)

The documentation provided here describes how to run the ACH Deenvelope service and the ACH Envelope service examples.

- [Expected results](#)

The documentation provided here describes the results returned from the ACH Deenvelope CTX and DNE examples, and the ACH Envelope CTX and DNE examples.

## Deploying the example maps to IBM Sterling B2B Integrator

Maps are deployed using either Design Studio, or through use of the IBM Sterling B2B Integrator.

- [Deploying the map using the Design Studio](#)

This documentation describes how to deploy the map to IBM Sterling B2B Integrator using Design Studio.

- [Deploying the map using IBM Sterling B2B Integrator](#)

The following procedures describe an alternate method of checking in the map by using the IBM Sterling B2B Integrator dashboard.

## Deploying the map using the Design Studio

This documentation describes how to deploy the map to IBM Sterling B2B Integrator using Design Studio.

### Procedure

1. Configure the Design Studio preferences before you attempt to connect to the IBM Sterling B2B Integrator server. See the Design Studio documentation for more information.
2. Open the source map in Design Studio.
3. Right click on the map, then choose Deploy to Sterling B2B Integrator. Enter your ID and password if you are prompted.

## Deploying the map using IBM Sterling B2B Integrator

The following procedures describe an alternate method of checking in the map by using the IBM Sterling B2B Integrator dashboard.

### Procedure

1. Compile the source maps for the correct platform on which the IBM Sterling B2B Integrator is installed.
2. From the IBM Sterling B2B Integrator dashboard, go to Deployment > Maps, then select the Check in new Map from Map Editor option.  
If the map has already been deployed to the IBM Sterling B2B Integrator, you can update it by first searching for, and locating the map. You can then use the Map Source Manager to check out the map and check in the new version.

## Running the ACH service examples

The documentation provided here describes how to run the ACH Deenvelope service and the ACH Envelope service examples.

To run the examples, you must first perform the following activities:

- Import IBM Sterling B2B Integrator artifacts
- Deploy the map to IBM Sterling B2B Integrator
- **[Importing IBM Sterling B2B Integrator artifacts for ACH service examples](#)**  
From the IBM Sterling B2B Integrator dashboard, import one of the following files from the platform\_support directory of the example.
- **[Running the ACH service examples from the Transformation Extender](#)**  
The following procedures describe how to run the ACH Deenvelope and ACH Envelope examples from the Transformation Extender.
- **[Running the ACH service examples from IBM Sterling B2B Integrator](#)**  
The following procedures describe how to run the ACH Deenvelope and ACH Envelope examples using the IBM Sterling B2B Integrator dashboard.

## Importing IBM Sterling B2B Integrator artifacts for ACH service examples

From the IBM Sterling B2B Integrator dashboard, import one of the following files from the platform\_support directory of the example.

### About this task

- If you are running the CTX Deenvelope example, import: ExportACHctxDeenvSIOBJECTS.xml
- If you are running the CTX Envelope example, import: ExportACHctxEnvSIOBJECTS.xml
- If you are running the DNE Deenvelope example, import: ExportACHdneDeenvSIOBJECTS.xml
- If you are running the DNE Envelope example, import: ExportACHdneEnvSIOBJECTS.xml

### Procedure

1. Log in to the IBM Sterling B2B Integrator dashboard.
2. From the Administration menu, expand the Deployment and Resource Manager sections, and then select the Import/Export option.
3. Choose the Import Resources option.
  - a. For the ACH Deenvelope services example for CTX, select: ExportACHctxDeenvSIOBJECTS.xml
  - b. For the ACH Envelope services example for CTX, select: ExportACHctxEnvSIOBJECTS.xml
  - c. For the ACH Deenvelope services example for DNE, select: ExportACHdneDeenvSIOBJECTS.xml
  - d. For the ACH Envelope services example for DNE, select: ExportACHdneEnvSIOBJECTS.xml
4. Enter "password" as the Passphrase, and then check the Import All Resources check box.
5. Click Next.
6. In the Create Resource Tag window, leave the Tag Name and Tag Description entries blank and then click Next.
7. In the Update Objects window, select Yes to update objects if they exist in the system, and then click Next.
8. In the Confirm window, click Finish.
9. View the Import and Performance reports. When finished, click Return.

## Running the ACH service examples from the Transformation Extender

The following procedures describe how to run the ACH Deenvelope and ACH Envelope examples from the Transformation Extender.

### Procedure

1. Use the Design Studio to open the source map for the example that you want to run:
  - a. If you are running the ACH Deenvelope service CTX example, open the inbound\_ach\_ctx.mms source map and then open these maps:
    - x12\_ra\_to\_atx\_pri
    - x12\_ra\_to\_flat\_rpt
    - x12\_ra\_to\_xml\_ctx
    - xml\_ctx\_to\_csv\_ctx
  - b. If you are running the ACH Envelope service CTX example, open the outbound\_ach\_ctx.mms source map and then open x12\_ra\_to\_xml\_ctx.
  - c. If you are running the ACH Deenvelope service DNE example, open the inbound\_ach\_dne.mms source map and then open these maps:
    - inbound\_dne\_pri\_to\_xml
    - xml\_dne\_to\_dne\_flat
  - d. If you are running the ACH Envelope service DNE example, open the outbound\_ach\_dne.mms source map and then open these maps:
    - outbound\_dne\_addenda
    - outbound\_dne\_entryDtls
2. Right click on the map, and then choose Run on IBM Sterling B2B Integrator. Enter the ID and password if you are prompted.  
The processing completes.
3. The following output files are created in each corresponding ACH service example scenario data directory:
  - a. If you are running the ACH Deenvelope service CTX example, these output files are created:
    - ach\_atx\_pri.out - (from the x12\_ra\_to\_atx\_pri map)
    - pymt\_ordr\_rpt.out - (from the x12\_ra\_to\_flat\_rpt map)
    - ach\_ctx\_out.xml (from the x12\_ra\_to\_xml\_ctx map)
    - ctx\_xml\_to\_csv.out (from the xml\_ctx\_to\_csv\_ctx map)
  - b. If you are running the ACH Envelope service CTX example, this output file is created:
    - ach\_ctx\_out.xml - (from the x12\_ra\_to\_xml\_ctx map)
  - c. If you are running the ACH Deenvelope service DNE example, these output files are created:
    - pri\_out.xml (from the inbound\_dne\_pri\_to\_xml map)
    - ach\_dne\_flat.out (from the xml\_dne\_to\_dne\_flat map)

d. If you are running the ACH Envelope service DNE example, these output files are created:

- ach\_dne\_entry\_dtls.out (from outbound\_dne\_entryDtls map)
- ach\_dne\_addenda.out (from outbound\_dne\_addenda map)

## Results

---

Other files might be generated in the maps directory: *map\_name\_MapTestRequest.msg*, and *map\_name\_MapResponse.msg*. These files include the results returned by the web service that runs the map, including the output data and other status information.

## Running the ACH service examples from IBM Sterling B2B Integrator

The following procedures describe how to run the ACH Deenvelope and ACH Envelope examples using the IBM Sterling B2B Integrator dashboard.

## Procedure

---

1. Import IBM Sterling B2B Integrator artifacts as described in [Importing IBM Sterling B2B Integrator artifacts for ACH service examples](#).
2. Deploy the example maps as described in [Deploying the example maps to IBM Sterling B2B Integrator](#).
3. From the Administration menu, go to the Business Process Manager page.
4. Search for a Business Process:
  - a. If you are running the IBM Sterling B2B Integrator ACH Deenvelope service CTX example, search for the EDIDeveloppe business process.
  - b. If you are running the IBM Sterling B2B Integrator ACH Envelope service CTX example, search for the ACH\_CTX\_EnvelopeUsingWTX business process.
  - c. If you are running the IBM Sterling B2B Integrator ACH Deenvelope service DNE example, search for the EDIDeveloppe business process.
  - d. If you are running the IBM Sterling B2B Integrator ACH Envelope service DNE example, search for the ACH\_DNE\_EnvelopeUsingWTX business process.
5. Select the execution manager for the Business Process, then select the Execute option.
6. On the Local Desktop filename, browse, and then select, the appropriate input file from the data folder.
  - If you are running the ACH Deenvelope service CTX example, select the ach\_ctx\_inbound.inp file.
  - If you are running the ACH Envelope service CTX example, select the x12\_ra\_to\_ctx.inp file.
  - If you are running the ACH Deenvelope service DNE example, select the ach\_dne\_inbound.inp file.
  - If you are running the ACH Envelope service DNE example, select the ach\_dne\_flat.inp file.  
Note: If the IBM Sterling B2B Integrator server is operating on UNIX, use ach\_dne\_flat\_LNX.inp as the input file.
7. Select the Go option.
8. The Execute Business Process results show a status of Success for all steps in the Business Process. You can view more details, such as Status Report, document, and Instance Data by going to Business Process > Monitor > Current Process. The translated output of the examples can be viewed by going to: Business > Monitor > Current Documents. The document information shows the translated output of the generic de-envelope and envelope service. The Document Instance Data shows the Process data values that are set by the service, including the traceNo values set by the map.

## Expected results

---

The documentation provided here describes the results returned from the ACH Deenvelope CTX and DNE examples, and the ACH Envelope CTX and DNE examples.

- [\*\*CTX Deenvelope example expected results\*\*](#)

Expected results for the CTX Deenvelope example are provide here.

- [\*\*DNE Deenvelope example expected results\*\*](#)

Expected results for the DNE Deenvelope example are provided here.

- [\*\*CTX Envelope example expected results\*\*](#)

Expected results for the CTX Envelope example are provided here.

- [\*\*DNE Envelope example expected results\*\*](#)

Expected results for the DNE Envelope example are provided here.

## CTX Deenvelope example expected results

---

Expected results for the CTX Deenvelope example are provide here.

- When analyzing the 5010\_820.mtt type tree, Warnings are not expected. Errors are not expected.
- The xml\_ctx\_to\_csv\_ctx translation map creates a comma separated file based on the ACH CTX message.
- The x12\_ra\_to\_flat\_rpt translation map creates a back office payment order report file based on the ACH CTX message.
- The x12\_ra\_to\_xml\_ctx transformation map creates an XML equivalent of the X12 EDI 820 used by the ACH service for validation.
- The service creates an acknowledgement for the CTX message, an ACH ATX message. The payment related information of the ATX message is based on the result of the x12\_ra\_to\_atx\_pri translation map .
- The entry for trace number, traceNo, exists in both the ProcessData and Correlation tables.

## DNE Deenvelope example expected results

---

Expected results for the DNE Deenvelope example are provided here.

- When analyzing the ach\_pyamtRltdInfo.mtt type tree, Warnings are not expected. Errors are not expected.
- The xml\_dne\_to\_dne\_flat translation map will extract one Addenda record and one Entry detail record.

- The inbound\_dne\_pri\_to\_xml translation map will transform the DNE Payment Related information into XML format. This XML payment related information will be used by the service to complete its internal XML representation of the DNE message.
  - The entry for trace number, traceNo, exists in both the ProcessData and Correlation tables.
- 

## CTX Envelope example expected results

Expected results for the CTX Envelope example are provided here.

- When analyzing the 5010\_820.mtt type tree, Warnings are not expected. Errors are not expected.
  - The x12\_ra\_to\_xml\_ctx transformation map creates an XML equivalent of the X12 EDI 820 used by the ACH service for building an ACH CTX message.
  - The service creates an ACH CTX message with Addenda records based from the input X12 EDI 820 data.
  - The entry for trace number, traceNo, exists in both the ProcessData and Correlation tables.
- 

## DNE Envelope example expected results

Expected results for the DNE Envelope example are provided here.

- When analyzing the ach\_dne\_flat.mtt type tree, Warnings are not expected. Errors are not expected.
  - The outbound\_dne\_entryDtls translation map extracts entry detail records from a back office file which holds both addenda and entry detail information.
  - The outbound\_dne\_addenda translation map extracts addenda records from a back office file which holds both addenda and entry detail information.
  - The service creates an ACH DNE message consisting of three entry detail records with three corresponding addenda records.
  - The entry for trace number, traceNo, exists in both the ProcessData and Correlation tables.
- 

## Design considerations of the examples

The documentation provided here describes design features and concepts that were taken into account when building the example transformation maps invoked by the ACH Deenvelope and ACH Envelope services

- **[IBM Sterling B2B Integrator data harness](#)**  
Some Transformation Extender maps share information with the IBM Sterling B2B Integrator processes.
  - **[Thread safety](#)**  
A thread-safe map can run in a multi-threaded environment.
  - **[Specific map and type tree settings](#)**  
These are the map and type tree settings used with IBM Sterling B2B Integrator.
  - **[Custom type trees](#)**  
The custom type trees described here are provided with the ACH Deenvelope and ACH Envelope examples.
  - **[ACH service XML schemas](#)**  
The ACH Deenvelope service has features that transform ACH messages into intermediary XML formats. The Pack for Financial Payments NACHA component provides XML schemas that describe content and constraints on the IBM Sterling B2B Integrator proprietary XML documents.
- 

## IBM Sterling B2B Integrator data harness

Some Transformation Extender maps share information with the IBM Sterling B2B Integrator processes.

For example, the maps can perform one of the following activities:

- Store information about the data, for example, a document ID, or other key field, that can be used by IBM Sterling B2B Integrator to track the data.
- Get information from IBM Sterling B2B Integrator tables, such as trading partner information, or code list table data. The maps read and write this information to IBM Sterling B2B Integrator database tables using the IBM Sterling B2B Integrator adapter. This is referred to as the IBM Sterling B2B Integrator Data Harness which provides access to these tables.

The IBM Sterling B2B Integrator adapter is used to get and to set certain values that are maintained in the IBM Sterling B2B Integrator adapter database tables.

- The setProcessData call is used to set values such as reference ids in the processData table. This value can be used by other services in the Business Process flow.
- The setCorrelation call is used to set the values such as reference ids in the correlation data table. This value can be used to help search for documents processed by the map.

The ACH Deenvelope service and ACH Envelope service example maps use PUT calls to the IBM Sterling B2B Integrator adapter in order to set the TraceNo with the following values in the IBM Sterling B2B Integrator database called ProcessData and Correlation tables:

- For CTX scenarios, and for DNE inbound scenario - Originating DFI Identification
- For DNE outbound scenario - Receiving DFI Identification

For more information about the IBM Sterling B2B Integrator Data Harness, see the Transformation Extender IBM Sterling B2B Integrator documentation.

---

## Thread safety

A thread-safe map can run in a multi-threaded environment.

One of the attributes a map needs in order to run in a multi-threaded environment is that when multiple occurrences of the map are instantiated, none of the resources used by an individual instance of the map are used by another instance of the map.

For example, the map's work files, audit logs, backup files, and trace logs are individually named, that is, they are unique within each instance of the map.

For more information, about thread safety, including how you can verify that the map meets at least the basic thread-safe checks, see the Transformation Extender Map Designer documentation.

## Specific map and type tree settings

These are the map and type tree settings used with IBM Sterling B2B Integrator.

- **Burst**  
The *burst* concept deals with the scope of the data that is processed in one run of the map. ACH Envelope service example maps are set to execute in burst mode.
- **Audit settings**  
Certain map settings allow the Transformation Extender map to generate data audit information that can be used by the Generic service in business process reports.
- **Restart**  
These are the Restart flag settings.

## Burst

The *burst* concept deals with the scope of the data that is processed in one run of the map. ACH Envelope service example maps are set to execute in burst mode.

The ACH Envelope service runs a portion of the input with each iteration of the map. This is expected behavior in several IBM Sterling B2B Integrator services.

In the ACH Envelope service example, in the map input card, the FetchAs property is set to Burst, with a FetchUnit of 1. This allows the Transformation Extender map to generate data audit information that can be used by the ACH.

## Audit settings

Certain map settings allow the Transformation Extender map to generate data audit information that can be used by the Generic service in business process reports.

In the map settings, the MapAudit switch is set to ON, the BurstAudit and SummaryAudit settings are set to Always, and the AuditLocation is set to Memory.

- **ACH Deenvelope service data audit settings**  
In the ACH Deenvelope service examples, the Data Audit Settings for the maps in the CTX and DNE examples are set as listed here.
- **ACH Envelope service data audit settings**  
These are the data audit settings for the maps in the ACH Envelope examples.

## ACH Deenvelope service data audit settings

In the ACH Deenvelope service examples, the Data Audit Settings for the maps in the CTX and DNE examples are set as listed here.

The following are the data audit settings for the x12\_ra\_to\_flat\_rpt, the x12\_ra\_to\_flat\_rpt, and the x12\_ra\_to\_xml\_ctx maps in the ACH Deenvelope CTX example.

|           |                                                                                                                         |
|-----------|-------------------------------------------------------------------------------------------------------------------------|
| Audit     | Partner X12 Inbound Interchange<br>Inbound Partner Funct'l Group ANSI<br>Transaction #820 Inbound Partner Set V5010X218 |
| Track     | Occurrence                                                                                                              |
| Details   | Occurrence                                                                                                              |
| Item Data | Occurrence                                                                                                              |

The following are the data audit settings for the xml\_ctx\_to\_csv\_ctx map in the ACH Deenvelope CTX example.

|           |                                          |
|-----------|------------------------------------------|
| Audit     | Audit all:ExtractedFields:global:xml_ctx |
| Track     | Occurrence                               |
| Details   | Occurrence                               |
| Item Data | Occurrence                               |

The following are the data audit settings for the xml\_dne\_to\_dne\_flat map in the ACH Deenvelope DNE example.

|           |                                      |
|-----------|--------------------------------------|
| Audit     | Audit sequence:OUTPUT:global:xml_dne |
| Track     | Occurrence                           |
| Details   | Occurrence                           |
| Item Data | Occurrence                           |

The following are the data audit settings for the inbound\_dne\_pri\_to\_xml map in the ACH Deenvelope DNE example.

|           |                        |
|-----------|------------------------|
| Audit     | PRI Record:ach_dne_pri |
| Track     | Occurrence             |
| Details   | Occurrence             |
| Item Data | Occurrence             |

## ACH Envelope service data audit settings

These are the data audit settings for the maps in the ACH Envelope examples.

The following are the data audit settings for the x12\_ra\_to\_xml\_ctx map in the ACH Envelope CTX example.

|           |                                                                                                                    |
|-----------|--------------------------------------------------------------------------------------------------------------------|
| Audit     | Transaction #820 Inbound Partner Set 5010X218<br>Partner X12 Inbound Interchange Inbound Partner Funct'lGroup ANSI |
| Track     | Occurrence                                                                                                         |
| Details   | Occurrence                                                                                                         |
| Item Data | Occurrence                                                                                                         |

The following are the data audit settings for the outbound\_dne\_entryDtls map and the outbound\_dne\_addenda map in the ACH Envelope DNE example.

|           |                      |
|-----------|----------------------|
| Audit     | group batch:flat_DNE |
| Track     | Occurrence           |
| Details   | Occurrence           |
| Item Data | Occurrence           |

## Restart

These are the Restart flag settings.

- [ACH Deenvelope service](#)  
The Restart flag on the 5010\_820.mtt and ach\_pyamtRltInfo.mtt type trees are specified on the following components.
- [ACH Envelope service](#)  
The Restart flags on the 5010\_820.mtt and the ach\_dne\_flat.mtt type trees are specified on the following components.

## ACH Deenvelope service

The Restart flag on the 5010\_820.mtt and ach\_pyamtRltInfo.mtt type trees are specified on the following components.

- CTX example - The Restart flag on the 5010\_820.mtt type tree is specified on the following components:
  - Partner X12 Inbound Interchange (s)
  - Transaction #820 Inbound Partner Set 5010X218 (s)
  - Inbound Partner Funct'lGroup ANSI (0:99999)
- DNE example- The Restart flag on the ach\_pyamtRltInfo.mtt type tree is specified on the following component:
  - PRI Record (1:s)

## ACH Envelope service

The Restart flags on the 5010\_820.mtt and the ach\_dne\_flat.mtt type trees are specified on the following components.

- CTX example - The Restart flag on the 5010\_820.mtt type tree are specified on the following components
  - Partner X12 Inbound Interchange (s)
  - Transaction #820 Inbound Partner Set 5010X218 (s)
  - Inbound partner Funct'lGroup ANSI (0:99999)
- DNE example - The Restart flag on the ach\_dne\_flat.mtt type tree is specified on the following component:
  - group batch (1:s)

## Custom type trees

The custom type trees described here are provided with the ACH Deenvelope and ACH Envelope examples.

- [5010\\_820.mtt type tree](#)  
On both the ACH Deenvelope service and the ACH Envelope service CTX examples, the service extracts the EDI X12 820 from the CTX Addenda records and inputs it to the transformation maps configured within the service.
- [ach\\_pyamtRltInfo.mtt type tree](#)  
On the ACH Deenvelope service DNE example, the service extracts the Payment Related Information from the DNE message and inputs it to a transformation map that is configured within the service.
- [ach\\_dne\\_flat.mtt type tree](#)  
On the ACH Envelope service DNE example, the service has a back-office input file which has two record formats, entry details and addenda records.
- [pyamt\\_ordr\\_rpt.mtt](#)  
On the ACH Deenvelope service CTX example, this type tree is used by the service post-processing map, x12\_ra\_to\_flat\_rpt , when transforming the X12 820 into a back office payment order report format.
- [ach\\_ctx\\_csv.mtt](#)  
After the ACH Deenvelope service CTX example completes, the service generates a IBM Sterling B2B Integrator process data which contains an XML representation

of the CTX message.

## 5010\_820.mtt type tree

On both the ACH Deenvelope service and the ACH Envelope service CTX examples, the service extracts the EDI X12 820 from the CTX Addenda records and inputs it to the transformation maps configured within the service.

The NACHA component of the Pack for Financial Payments ships a type tree that represents the EDI ANSI 5010X218 X12 820 message syntax. This type tree is used by the map x12\_ra\_to\_xml\_ctx to validate the X12 820 and transform it to IBM Sterling B2B Integrator proprietary XML format. This type tree is used by the ACH Deenvelope service post processing map x12\_ra\_to\_flat\_rpt to transform the X12 820 to a back office payment order report. This type tree is used by the ACH Deenvelope ACK Addenda Record map x12\_ra\_to\_atx\_pri which extracts the REF segment and is used by the service to generate the Payment Related Information on the acknowledgement ATX message.

## ach\_pymtRltdInfo.mtt type tree

On the ACH Deenvelope service DNE example, the service extracts the Payment Related Information from the DNE message and inputs it to a transformation map that is configured within the service.

The NACHA component of the Pack for Financial Payments includes a type tree that represents the 80 character Payment Related Information syntax. This type tree is used by the map inbound\_dne\_pri\_to\_xml which converts the PRI into XML. The XML formatted PRI is embedded by the service to the IBM Sterling B2B Integrator proprietary XML representation of the DNE message. The same type tree is used in the ACH Deenvelope service CTX example, ACK Addenda Record map x12\_ra\_to\_atx\_pri when populating the contents of the PRI of the ATX message.

## ach\_dne\_flat.mtt type tree

On the ACH Envelope service DNE example, the service has a back-office input file which has two record formats, entry details and addenda records.

The Pack for Financial Payments NACHA component ships a type tree that represents the 77 character Entry Detail Record and 80 character Addenda record syntax. This type tree is used by the service Entry Detail map outbound\_dne\_entryDtls to extract the entry detail record. The type tree is used by the service Addenda map outbound\_dne\_addenda to extract the addenda record. Output from both maps is used by the service to populate a DNE message. The same type tree is used in the ACH Deenvelope service DNE example, post-processing map xml\_dne\_to\_dne\_flat when transforming IBM Sterling B2B Integrator proprietary XML format to a back office format.

## pymt\_ordr\_rpt.mtt

On the ACH Deenvelope service CTX example, this type tree is used by the service post-processing map, x12\_ra\_to\_flat\_rpt , when transforming the X12 820 into a back office payment order report format.

## ach\_ctx\_csv.mtt

After the ACH Deenvelope service CTX example completes, the service generates a IBM Sterling B2B Integrator process data which contains an XML representation of the CTX message.

This type tree is used by an example map called xml\_ctx\_to\_csv\_ctx to transform the XML file extracted from IBM Sterling B2B Integrator process data which represents a CTX message into a comma-separated value file format.

## ACH service XML schemas

The ACH Deenvelope service has features that transform ACH messages into intermediary XML formats. The Pack for Financial Payments NACHA component provides XML schemas that describe content and constraints on the IBM Sterling B2B Integrator proprietary XML documents.

The XML schemas allow Transformation Extender maps that are configured within ACH service to work directly with the service XML representation of ACH messages.

These XML schemas are provided.

- ach.ack.001.xsd
- ach.adv.001.xsd
- ach.arc.001.xsd
- ach.atx.001.xsd
- ach.boc.001.xsd
- ach.ccd.001.xsd
- ach.cie.001.xsd
- ach.cmn.001.xsd ( common base types shared by the different XML schemas)
- ach.cntstd.dshnrd.rtrn.001.xsd
- ach.cor.001.xsd

- ach.ctx.001.xsd
- ach.ctx.002.xsd
- ach.dne.001.xsd
- ach.dshnrd.rtrn.001.xsd
- ach.enr.001.xsd
- ach.iat.001.xsd
- ach.iat.cor.001.xsd
- ach.iat.rtrn.001.xsd
- ach.mte.001.xsd
- ach.pop.001.xsd
- ach.pos.001.xsd
- ach.ppd.001.xsd
- ach.rck.001.xsd
- ach.rfsd.ack.001.xsd
- ach.rfsd.atx.001.xsd
- ach.rfsd.cor.001.xsd
- ach.rtrn.001.xsd
- ach.shr.001.xsd
- ach.tel.001.xsd
- ach.trc.001.xsd
- ach trx.001.xsd
- ach.web.001.xsd
- ach.xck.001.xsd

For examples of how to use the XML schema, see the ACH Deenvelope service CTX and DNE examples.

## Overview of the SEPA component

The SEPA component of the Pack for Financial Payments supports validation of the Single European Payments Area (SEPA) scheme datasets, transformation samples between SEPA, and individual domestic formats.

The SEPA component can link your current payments systems and operations in their current mode to SEPA validation, providing processing and enrichment for your back office and intermediary systems.

This documentation provides introductory and reference material needed to implement the SEPA component of the Pack for Financial Payments as part of an overall SEPA solution. This documentation contains background on SEPA, including the following topics:

- A description of how the SEPA component fits into your SEPA processing for credit transfer and direct debit transactions
- Configuration of the SEPA component, including specifics of deploying the component on z/OS
- A description of the SEPA message validation framework
- A discussion of the domestic format converters.

For a description of installation and uninstallation procedures for this product, refer to the release notes.

- **[Principal components](#)**

The SEPA component of the Pack for Financial Payments contains the following principal components:

- **[The SEPA initiative](#)**

In today's financial marketplace, customers are demanding improved services and want a prompt payment processing system that is consistent both within their home country, and in other European markets.

## Principal components

The SEPA component of the Pack for Financial Payments contains the following principal components:

- ISO 20022 (UNIFI) message schemas for credit transfer and direct debit (.xsd files) and the type trees for credit transfer and direct debit formats.
- Additional validation maps used to validate the general usage rules as defined in the EPC rule books.
- Example converter maps for the following domestic credit transfer and direct debit messages:
  - United Kingdom - BACS, see [Example for the UK](#).
  - Germany - DTAS, see [Example for Germany](#).
  - France - MINOS and CFONB, see [Example for France](#).
  - Italy - SIA Rete Interbacaria (RNI), see [Example for Italy](#).
- Type trees for the SWIFTNet FIN MT 1nn message, as well as the transformation maps used to convert between SEPA formats and the SWIFTNet FIN MT101 MT102, MT102+, MT103 and MT103+ messages.
- z/OS example with sample JCL for a SEPA pass through map. See [Running the SEPA component on z/OS](#).
- Reference data templates (bic.xml, and currencycodedecimals.xml files).
- An example validation map that uses the European Payments Council (EPC) schemas for the relevant payments clearing and settlement messages that use native schema mapping. These maps can be used as a replacement for the main validation maps to improve performance, since these maps use the EPC schema subset of the ISO20022 schema.

## The SEPA initiative

In today's financial marketplace, customers are demanding improved services and want a prompt payment processing system that is consistent both within their home country, and in other European markets.

Each European country has developed its own banking and payment system in accordance with local market requirements. The result has been a varying level of payment service for customers. This is particularly noticeable to citizens who cross borders regularly to shop and holiday within Europe.

To address this problem, the EPC, in 2002, proposed a single payment process. At that time, 42 banks, and three European Credit Sector Associations (ECSAs), came together to release a proposal formally launching the SEPA concept.

The definition of SEPA was approved by the EPC in December of 2004 and states that:

*"SEPA will be the area where citizens, companies and other economic actors will be able to make and receive payments in euro, within Europe (currently defined as consisting of the 28 European Union (EU) member states plus Iceland, Norway, Liechtenstein and Switzerland), whether between or within national boundaries under the same basic conditions, rights and obligations, regardless of their location."*

- **[SEPA message standards](#)**

This section describes SEPA message standards and provides an overview of the UNIFI ISO 20022 (UNIFI) standard upon which the SEPA standard is based.

- **[EPC rulebook validation of SEPA messages](#)**

The EPC publishes various documents related to SEPA, including a data model reference, rulebooks, and implementation guides for both credit transfers and direct debits.

---

## SEPA message standards

This section describes SEPA message standards and provides an overview of the UNIFI ISO 20022 (UNIFI) standard upon which the SEPA standard is based.

Note: The terms "UNIFI" and "ISO 20022" are used interchangeably throughout this and other SEPA documentation. Both refer to the ISO 20022 Universal Financial Industry message scheme.

This platform was proposed by the International Standards Organization (ISO) to be used to develop all financial industry messages. It must be understood that UNIFI by itself does not describe messages. Rather, it can be thought of as a recipe to be used to develop financial industry message standards. The main ingredients of this recipe are a development methodology, a registration process, and a central repository.

- **[Message overview](#)**

The core messages defined for use by the EPC comprise a subset of the ISO 20022 Payments Clearing and Settlement (pacs.nnn.nnn.nn), Cash Management (camt.nnn.nnn.nn), and Payments Initiation (pain.nnn.nnn.nn) business areas.

- **[About the UNIFI message standard](#)**

---

## Message overview

The core messages defined for use by the EPC comprise a subset of the ISO 20022 Payments Clearing and Settlement (pacs.nnn.nnn.nn), Cash Management (camt.nnn.nnn.nn), and Payments Initiation (pain.nnn.nnn.nn) business areas.

The **pacs** and **camt** messages are for use in the PSP-to-PSP processing space, and have been designated mandatory by the EPC and they must be used by all SEPA participants. The **pain** messages are recommended for use in the customer-to-PSP space.

It should be noted that, within SEPA, there are numerous terms that can be used interchangeably. In addition to the terms UNIFI and ISO 20022 XML which are nearly synonymous, a SEPA message can be referred to either by its UNIFI message number, or by a dataset identified in the EPC rulebook. For example, the following terms mean the same:

- FI to FI Customer Credit Transfer
- pacs.008.001.08
- Inter-PSP Payment Dataset – DS-02

- **[Credit Transfer messages](#)**

Credit transfer SEPA messages include Inter-PSP (PSP-to-PSP) and Customer to PSP messages.

- **[Credit Transfer - Inter-PSP messages](#)**

SEPA Credit Transfer Inter-PSP messages are defined in the following table.

- **[Credit Transfer - Customer-to-PSP messages](#)**

SEPA Credit Transfer, Customer-to-PSP messages, are defined in the following table:

---

## Credit Transfer messages

Credit transfer SEPA messages include Inter-PSP (PSP-to-PSP) and Customer to PSP messages.

- **[Credit Transfer - Inter-PSP messages](#)**

SEPA Credit Transfer Inter-PSP messages are defined in the following table.

- **[Credit Transfer - Customer-to-PSP messages](#)**

SEPA Credit Transfer, Customer-to-PSP messages, are defined in the following table:

---

## Credit Transfer - Inter-PSP messages

SEPA Credit Transfer Inter-PSP messages are defined in the following table.

| Message ID      | Description              | Dataset designation                                         | Purpose                                                                                                      |
|-----------------|--------------------------|-------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------|
| camt.027.001.07 | Use of Claim Non-Receipt | DS-09 - Inter-PSP SCT Inquiry Dataset for Claim Non-Receipt | Transports the Claim Non-Receipt from originator PSP to beneficiary PSP, directly, or through intermediaries |

| <b>Message ID</b> | <b>Description</b>                              | <b>Dataset designation</b>                                                              | <b>Purpose</b>                                                                                                                                                                                                                                                                                                                                                                                                                                          |
|-------------------|-------------------------------------------------|-----------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| camt.029.001.09   | Use of Resolution of Investigation              | DS-06 - Inter-PSP Negative Answer to a Recall of SEPA Credit Transfer Dataset           | In the context of a Recall of an SCT, this message is used to transmit a negative answer to a Recall of Credit Transfer message (FI-to-FI Payment Cancellation Request V01, camt.056.001.08), or a Request for Status Update on a Recall of Credit Transfer message (FI-to-FI Payment Status Request, pacs.028.001.03).                                                                                                                                 |
| camt.029.001.09   | Use of Resolution of Investigation              | DS-08 - Inter-PSP Negative Response to the Request for Recall by the Originator Dataset | The message transmits a negative answer to one of the following: <ul style="list-style-type: none"> <li>Request for Recall by the Originator message (FI-to-FI Payment Cancellation Request V01, camt.056.001.08).</li> <li>Request for Status Update on a Request for Recall by the Originator message (FI-to-FI Payment Status Request, pacs.028.001.03).</li> </ul>                                                                                  |
| camt.029.001.09   | Use of Resolution of Investigation              | DS-10 - Inter-PSP Negative Response to Claim Non-Receipt                                | This message is used to transmit a negative response to one of the following: <ul style="list-style-type: none"> <li>Claim Non-Receipt message (Claim Non-Receipt, camt.027.001.07)</li> <li>Request for Status Update on a Claim Non-Receipt message (FI-to-FI Payment Status Request, pacs.028.001.03)</li> </ul>                                                                                                                                     |
| camt.029.001.09   | Use of Resolution of Investigation              | DS-10 - Inter-PSP Positive Response to Claim Non-Receipt                                | This message is used to transmit a positive response to the following: <ul style="list-style-type: none"> <li>Claim Non-Receipt message (Claim Non-Receipt, camt.027.001.07)</li> <li>Request for Status Update on a Claim Non-Receipt message (FI-to-FI Payment Status Request, pacs.028.001.03)</li> </ul>                                                                                                                                            |
| camt.029.001.09   | Use of Resolution of Investigation              | DS-10 - Inter-PSP Negative Response to Claim for Value Date Correction                  | This message is used to transmit a negative response to the following: <ul style="list-style-type: none"> <li>Claim for Value Date Correction message (Request To Modify Payment, camt.087.001.06)</li> <li>Request for Status Update on a Claim for Value Date Correction message (FI-to-FI Payment Status Request, pacs.028.001.03)</li> </ul>                                                                                                        |
| camt.029.001.09   | Use of Resolution of Investigation              | DS-10 - Inter-PSP Confirmed Positive Response to Claim for Value Date Correction        | Transports a positive response to a Claim for Value Date Correction message (Request to Modify Payment, camt.087.001.06).                                                                                                                                                                                                                                                                                                                               |
| camt.029.001.09   | Use of FI To FI Payment Status Request          |                                                                                         | A pacs.028.001.03 message can be sent to the Beneficiary PSP in case the Originator PSP did not receive a response to an SCT Inquiry (Claim Non-Receipt or Claim for Value Date Correction).                                                                                                                                                                                                                                                            |
| camt.056.001.08   | Use of FI-to-FI Payment Cancellation Request    | DS-05 - Inter-PSP Recall of SEPA Credit Transfer Dataset                                | Transports the request to cancel a pacs.008.001.08, or to cancel specified transactions from the Originator PSP to the next PSP in the chain, or to the CSM, or from the CSM to the next PSP in the chain.                                                                                                                                                                                                                                              |
| camt.056.001.08   | Use of FI-to-FI Payment Cancellation Request    | DS-07 - Request for Recall by the Originator Dataset                                    | The message is used to submit the request made by the Originator to cancel a pacs.008.001.08 for other reasons than those allowed in a Recall.                                                                                                                                                                                                                                                                                                          |
| camt.087.001.06   | Use of Request to Modify Payment                | DS-09 -Inter-PSP SCT Inquiry Dataset for Claim for Value Date Correction                | This message is used to transport the Claim for Value Date Correction from the Originator PSP to the Beneficiary PSP, directly or through intermediaries.                                                                                                                                                                                                                                                                                               |
| pacs.002.001.10   | Use of the FI to FI Payment Status Report       | DS-03 - Inter-PSP Reject SEPA Credit Transfer Dataset                                   | Transports the credit transfer reject instruction between PSPs directly or through intermediaries. Caters to single and bulk instructions.                                                                                                                                                                                                                                                                                                              |
| pacs.004.001.09   | Payment Return                                  | DS-03 - Inter-PSP Return Credit Transfer Dataset                                        | Transports the credit transfer return instructions between PSPs, directly, or through intermediaries. Caters to single and bulk instructions.                                                                                                                                                                                                                                                                                                           |
| pacs.004.001.09   | Use of inter-PSP return credit transfer message | DS-06 - Inter-PSP Positive Answer to a Recall of SEPA Credit Transfer Dataset           | The positive answer to a Recall of Credit Transfer message (camt.056.001.08); or Request for Status Update on a Recall of Credit Transfer message (pacs.028.001.03).                                                                                                                                                                                                                                                                                    |
| pacs.004.001.09   | Use of inter-PSP return credit transfer message | DS-08 - Inter-PSP Positive Response to the Request for Recall by the Originator Dataset | In the context of a Request for Recall by the Originator of an SCT, this message is used to transmit a positive answer to the following: <ul style="list-style-type: none"> <li>Request for Recall by the Originator message (FI-to-FI Payment Cancellation Request V01, camt.056.001.08).</li> <li>Request for Status Update on a Request for Recall by the Originator message (FI-to-FI Payment Status Request, pacs.028.001.03).</li> </ul>          |
| pacs.008.001.08   | Use of FI to FI Customer Credit Transfer        | DS-02 - Inter-PSP Payment Dataset                                                       | Transports payment instructions from originator PSP to beneficiary PSP directly, or through intermediaries. Caters to single and bulk instructions.                                                                                                                                                                                                                                                                                                     |
| pacs.008.001.08   | Use of FI to FI Customer Credit Transfer        |                                                                                         | The ERI Option supports the transmission of one occurrence of 140 characters of Unstructured Remittance Information (AT-61) and up to 999 occurrences of Structured Remittance Information (AT-62) within a single SEPA Credit Transfer Instruction to allow the swift settlement of several payment obligations for the Originator.                                                                                                                    |
| pacs.008.001.08   | Use of FI to FI Customer Credit Transfer        | DS-11 - Inter-PSP Fee and/or Compensation Payment Dataset                               | The message is used to transport the instruction related to the payment of: <ul style="list-style-type: none"> <li>An inter-PSP fee for handling the SCT inquiry in case of a positive response to an SCT inquiry for the reasons 'Claim of Non-Receipt' and 'Claim for Value Date Correction', and/or</li> <li>Interest compensation resulting from a positive response to an SCT inquiry for the reason 'Claim for Value Date Correction'.</li> </ul> |

| Message ID      | Description                                | Dataset designation                                                                                           | Purpose                                                                                                                                                                                                                                                                                                                                                                                                                                    |
|-----------------|--------------------------------------------|---------------------------------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| pacs.028.001.03 | Use of the FI to FI Payment Status Request |                                                                                                               | <p>This message is sent to the Beneficiary PSP in case the Originator PSP did not receive an answer to a Recall, where the following applies:</p> <ul style="list-style-type: none"> <li>Index 3.6 'Original Instruction Identification' must be populated with the 'Cancellation Identification' (AT-R7) of the related camt.056 transaction information</li> </ul> <p>Note: The message caters for a single or group status request.</p> |
| pacs.028.001.03 | Use of FI To FI Payment Status Request     |                                                                                                               | <p>This message is sent to the Beneficiary PSP in case the Originator PSP did not receive an answer to a Request for Recall by the Originator, where the following applies:</p> <ul style="list-style-type: none"> <li>Index 3.6 'Original Instruction Identification' must be populated with the 'Cancellation Identification' (AT-51) of the related camt.056 transaction information.</li> </ul>                                        |
| pacs.028.001.03 | Use of Resolution of Investigation         | DS-10 - Inter-PSP Positive Response to Claim for Value Date Correction with request for interest compensation | <p>This message is used to transmit a positive response to the following:</p> <ul style="list-style-type: none"> <li>Claim for Value Date Correction message (Request To Modify Payment, camt.087.001.06)</li> <li>Request for Status Update on a Claim for Value Date Correction message (FI - to-FI Payment Status Request, pacs.028.001.03)</li> </ul>                                                                                  |

## Credit Transfer - Customer-to-PSP messages

SEPA Credit Transfer, Customer-to-PSP messages, are defined in the following table:

| Message ID      | Description                                    | Dataset designation                                                                                                                                                             | Purpose                                                                                                                                                                                                                                                                                                                                                                                                                             |
|-----------------|------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| pain.001.001.09 | Use of the Customer Credit Transfer Initiation | DS-01 - Customer to PSP Credit Transfer Information                                                                                                                             | Initiation Customer PSP. Used to transport Customer to PSP credit transfer information sent by the Originator PSP. Caters to single and bulk instructions.                                                                                                                                                                                                                                                                          |
| pain.002.001.10 | Reject - Customer Payment Status Report        | based on DS-03 - PSP to Customer Reject Credit Transfer Dataset                                                                                                                 | The code 'RJCT' (Rejected) must be used in 'Group Status', or 'Payment Information Status' or 'Transaction Status', to transport the Credit Transfer Reject instruction between the PSPs and their remitting customers.<br>Note: The SEPA Implementation Guidelines are not sufficient to fully implement this message and require additional information under bilateral agreement between the customer and the PSP.               |
| pain.001.001.09 | Use of the Customer Credit Transfer Initiation | DS-01 - Transfer Back of Received Credit Transfer without Originator IBAN - based on a Customer to PSP Credit Transfer Information                                              | The message is used when the Beneficiary of an earlier executed SEPA Credit Transfer Transaction wishes to transfer back (in full or partly) funds to the Originator of this transaction but the Beneficiary does not have the IBAN of the account of the Originator.<br>As a consequence, the Beneficiary can provide in the SEPA Credit Transfer Transaction, an alternative identifier to this attribute to the Beneficiary PSP. |
| pain.001.001.09 | Use of the Customer Credit Transfer Initiation | DS-01 - Customer to PSP Credit Transfer Information with use of the Extended Remittance Information option as per Annex V 'Extended Remittance Information' of the SCT rulebook | Annex V "Extended Remittance Information" (ERI) to the SCT Rulebook (document EPC 152-18) describes this optional feature of the SCT scheme that can only be used between PSPs which have adhered to the option.                                                                                                                                                                                                                    |

## Direct debit messages

Direct debit messages include Inter-PSP (PSP-to-PSP), Customer to PSP, and E-Mandate messages.

- [\*\*Direct Debit - Inter-PSP messages\*\*](#)  
SEPA Direct Debit Inter-PSP messages include messages for both B2B and Core.
- [\*\*Direct Debit - Customer-to-PSP messages\*\*](#)  
SEPA Direct Debit, Customer-to-PSP messages include messages for both B2B and Core.
- [\*\*Direct Debit - E-Mandate messages\*\*](#)  
SEPA Direct Debit, E-Mandate messages include messages for both B2B and Core.

## Direct Debit - Inter-PSP messages

SEPA Direct Debit Inter-PSP messages include messages for both B2B and Core.

- [\*\*Direct Debit - Inter-PSP B2B messages\*\*](#)  
The SEPA Direct Debit, Inter-PSP, Business-to-Business (B2B), messages are defined in this table.
- [\*\*Direct Debit - Inter-PSP Core messages\*\*](#)  
The SEPA, Direct Debit Inter-PSP Core messages are defined in this table.

## Direct Debit - Inter-PSP B2B messages

The SEPA Direct Debit, Inter-PSP, Business-to-Business (B2B), messages are defined in this table.

| Message ID      | Description                    | Dataset designation                                        | Purpose                                                                                                                                                                              |
|-----------------|--------------------------------|------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| pacs.002.001.10 | Payment Status Report - Reject | DS-05 Inter-PSP Direct Debit Reject Dataset                | Transports the B2B direct debit reject instruction between PSPs, directly or thru intermediaries. Caters to single and bulk instructions.                                            |
| pacs.003.001.08 | FI to FI Customer Direct Debit | DS-04 Inter-PSP Collection                                 | Transports the B2B direct debit collection instruction from the Creditor to the Debtor PSP, directly or thru intermediaries. Caters to single and bulk instructions.                 |
| pacs.004.001.09 | Payment Return                 | DS-05 Inter-PSP Direct Debit Return/Refund of a Collection | Transports the B2B direct debit return instruction from debtor PSP to creditor PSP, directly or through intermediaries. Caters to single and bulk instructions                       |
| pacs.007.001.09 | Payment Reversal               | DS-07 Inter-PSP Reversal Instruction for a Collection      | Transports the inter-PSP reversal instruction for a collection from the creditor PSP to the debtor PSP, directly, or through intermediaries. Caters to single and bulk instructions. |

## Direct Debit - Inter-PSP Core messages

The SEPA, Direct Debit Inter-PSP Core messages are defined in this table.

| Message ID      | Description                             | Dataset designation                                          | Purpose                                                                                                                                                                                                      |
|-----------------|-----------------------------------------|--------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| pacs.002.001.10 | FI to FI Payment Status Report - Reject | DS-05 - Inter-PSP Direct Debit Reject Dataset                | Transports the Direct Debit Rejection instruction between PSPs, directly or through intermediaries. The message caters for bulk and single reject instructions.                                              |
| pacs.003.001.08 | FI to FI Customer Direct Debit          | DS-04 - Inter-PSP Collection                                 | Transport the Direct Debit Collection instruction from the Creditor PSP to the Debtor PSP, directly or through intermediaries. The message caters for bulk and single direct debit instructions.             |
| pacs.004.001.09 | Return or Refund - Payment Return       | DS-05 - Inter-PSP Direct Debit Return/Refund of a Collection | Transports the direct debit return/refund instruction from the Debtor PSP to the Creditor PSP, directly or through intermediaries.                                                                           |
| pacs.007.001.09 | Payment Reversal                        | DS-07 - Inter-PSP Reversal Instruction for a Collection      | Transports the Inter-PSP Reversal Instruction for a Collection sent by the Creditor PSP to the Debtor PSP, directly or through intermediaries. The message caters for bulk and single reversal instructions. |

## Direct Debit - Customer-to-PSP messages

SEPA Direct Debit, Customer-to-PSP messages include messages for both B2B and Core.

- [Direct Debit - Customer-to-PSP B2B messages](#)  
The SEPA, Direct Debit, Customer-to-PSP, Business-to-Business (B2B), messages are defined in the following table.
- [Direct Debit - Customer-to-PSP core messages](#)  
The SEPA, Direct Debit, Customer-to-PSP, core messages are defined in this table.

## Direct Debit - Customer-to-PSP B2B messages

The SEPA, Direct Debit, Customer-to-PSP, Business-to-Business (B2B), messages are defined in the following table.

| Message ID      | Description                             | Dataset designation                                                             | Purpose                                                                                                                                            |
|-----------------|-----------------------------------------|---------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------|
| pain.002.001.10 | Customer Payment Status Report - Reject | PSP to Customer Direct Debit Reject Dataset (Based on DS-05)                    | Transports B2B direct debit reject instructions between PSPs and remitting customers. Caters to single and bulk instructions.                      |
| pain.007.001.09 | Customer Payment Reversal               | Customer to PSP Reversal Instruction for a Collection (Based on DS07 and DS-03) | Transports the customer to PSP reversal instruction for a collection from the creditor to the creditor PSP. Caters to single and bulk instructions |
| pain.008.001.08 | Customer Direct Debit Initiation        | DS-03 Customer to PSP Direct Debit Collection Dataset                           | Transports the customer to PSP reversal instruction for a collection from the creditor to the creditor PSP. Caters to single and bulk instructions |

## Direct Debit - Customer-to-PSP core messages

The SEPA, Direct Debit, Customer-to-PSP, core messages are defined in this table.

| Message ID      | Description                              | Dataset designation                                                              | Purpose                                                                                                                                                                      |
|-----------------|------------------------------------------|----------------------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| pain.002.001.10 | Payment Status Report                    | DS-05 - PSP to Customer Direct Debit Reject Dataset                              | Transports the Direct Debit Reject instruction between the PSP and its remitting customer. The message caters for bulk and single reject instructions.                       |
| pain.007.001.09 | Customer to PSP Payment Reversal         | Customer to PSP Reversal Instruction for a Collection (Based on DS07 and DS-03). | Transports the Customer to PSP Reversal Instruction for a Collection sent by the Creditor to the Creditor PSP. The message caters for bulk and single reversal instructions. |
| pain.008.001.08 | Use of Customer Direct Debit Initiation. | DS-03 - Customer to PSP Direct Debit Collection Dataset.                         | Transports the Customer to PSP Reversal Instruction for a Collection sent by the Creditor to the Creditor PSP. The message caters for bulk and single reversal instructions. |

## Direct Debit - E-Mandate messages

SEPA Direct Debit, E-Mandate messages include messages for both B2B and Core.

- **Direct Debit - E-Mandate B2B messages**  
The SEPA, Direct Debit, E-Mandate, B2B messages are defined in this table.
- **Direct Debit - E-Mandate Core messages**  
The SEPA, Direct Debit, E-Mandate, Core messages are defined in this table.

## Direct Debit - E-Mandate B2B messages

The SEPA, Direct Debit, E-Mandate, B2B messages are defined in this table.

| Message ID          | Description                  | Dataset designation                                    | Purpose                                                                                                                                                                                                                                                  |
|---------------------|------------------------------|--------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| pain.009.001.0<br>6 | Mandate Initiation Request   | DS-12 - E-Mandate Proposal/Request Message - Issuing   | Transports the request to use the validation service from the Creditor to the Debtor PSP. The message caters for a single request.                                                                                                                       |
| pain.010.001.0<br>6 | Mandate Amendment Request    | DS-12 - E-Mandate Proposal/Request Message - Amendment | Transports the amendment of a mandate sent by the Creditor to the Debtor PSP. The message caters for a single amendment request.                                                                                                                         |
| pain.011.001.0<br>6 | Mandate Cancellation Request | E-Mandate Proposal/Request Message – Cancellation      | Transports the cancellation request of a mandate sent by the Creditor to the Debtor PSP. The message caters for a single cancellation request.                                                                                                           |
| pain.012.001.0<br>6 | Mandate Acceptance Report    | DS-13 - E-Mandate Validation Message                   | Transports the confirmation of the acceptance or rejection of a mandate request (initiation, amendment or cancellation) sent by the Debtor PSP to the Creditor. The message caters for a single confirmation of rejection of a specific Mandate Request. |

## Direct Debit - E-Mandate Core messages

The SEPA, Direct Debit, E-Mandate, Core messages are defined in this table.

| Message ID          | Description                  | Dataset designation                                       | Purpose                                                                                                                                                                                                                                                 |
|---------------------|------------------------------|-----------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| pain.009.001.0<br>6 | Mandate Initiation Request   | DS-12 - E-Mandate Proposal/Request Message – Issuing      | Transports the request to use the validation service from the Creditor to the Debtor PSP. The message caters for a single request                                                                                                                       |
| pain.010.001.0<br>6 | Mandate Amendment Request    | DS-12 - E-Mandate Proposal/Request Message – Amendment    | Transports the amendment of a mandate sent by the Creditor to the Debtor PSP. The message caters for a single amendment request.                                                                                                                        |
| pain.011.001.0<br>6 | Mandate Cancellation Request | DS-12 - E-Mandate Proposal/Request Message – Cancellation | Transports the cancellation request of a mandate sent by the Creditor to the Debtor PSP. The message caters for a single cancellation request.                                                                                                          |
| pain.012.001.0<br>6 | Mandate Acceptance Report    | DS-13 - E-Mandate Validation Message                      | Transports the confirmation of the acceptance or rejection of a mandate request (initiation, amendment or cancellation) sent by the Debtor PSP to the Creditor. The message caters for a single confirmation of rejection of a specific Mandate Request |

## About the UNIFI message standard

At the beginning of the decade, the general interest in Internet Protocol (IP) and XML was perceived as a unique opportunity for the industry to move to a common XML-based language. There was a problem, however, because XML is not a language at all, but rather a meta-language which everyone can use to define their own dialect. As a result, several standardization initiatives grew from the common XML concept, each with messages using their own dialect, or adaptation of XML. This led to not only an increased risk of yet more 'languages' being introduced, but a waste of resources in the form of duplicated effort, and a further risk of divergence when more than one initiative is brought to focus on the same business area.

To solve this problem, ISO proposed a single standardization approach, or "recipe" which includes a common development methodology, a common process, and a common repository, that would be used by all financial standards initiatives. This recipe, called UNIFI or ISO 20022 is the message standard around which the SEPA component is developed.

For a complete description of ISO 20022 message flow, refer to the International Standardization Organization, ISO 20022 UNIversal Financial Industry message scheme. This information is available at:

[www.iso20022.org/](http://www.iso20022.org/)

## EPC rulebook validation of SEPA messages

The EPC publishes various documents related to SEPA, including a data model reference, rulebooks, and implementation guides for both credit transfers and direct debits.

The SEPA data model describes how three layers are recognized:

- A business process layer.
- A logical data layer to define the datasets and their attributes.
- The physical layer which specifies the actual document formats and messages.

The rulebooks contain the business requirements and rules for the operation of the SEPA schemes.

The implementation guides contain detailed specifications on the physical layer of the data model, and thus define SEPA-specific validation that is applied to ISO 20022 (UNIFI) messages to ensure compliance with the SEPA.

The SEPA component of the Pack for Financial Payments includes maps to apply these validation rules to the various ISO 20022 SEPA messages so that customers can validate their messages to ensure compliance. These messages are listed in the [Credit Transfer messages](#) and in the [Direct debit messages](#).

The maps provide a check-digit validation of the following fields:

- IBAN
- Identification of the Creditor (AT-02) - direct debit only

They also conduct validation checks against XML lookup files to verify the following items:

- Bank Identification Code (BIC)
- Currency codes
- Country codes
- [Business to Business direct debit implementation](#)  
The original SEPA scheme is the SEPA core direct debit scheme.
- [External Purpose Code List](#)  
The SEPA component of the Pack for Financial Payments contains an External Purpose Code List. This code list contains values that define the reason for payment transactions between customers.

## Business to Business direct debit implementation

The original SEPA scheme is the SEPA core direct debit scheme.

There is one difference in the actual data for the two schemas:

Payment Information – Local Instrument – The code is CORE for the Core Direct Debit Schema and B2B for the Business to Business schema.

## External Purpose Code List

The SEPA component of the Pack for Financial Payments contains an External Purpose Code List. This code list contains values that define the reason for payment transactions between customers.

This code list was developed within the framework of some of the ISO 20022 Payment messages. The code values can be used in specific elements of the Payments messages, and are required in order to validate all Customer to PSP messages. Although Pub code values can be used in specific elements of the Payments messages, the External Purpose Code List is not part of the ISO 20022 standard, as it is not a standalone ISO standard such as BIC, or Currency Codes. To access the current code values, go to the ISO 20022 website, [www.iso20022.org](http://www.iso20022.org).

## Configuration

The documentation provided here describes SEPA system configuration.

This includes a detailed description of the major components that comprise the SEPA component for the Pack for Financial Payments, followed by detailed instructions on how to run the SEPA component on z/OS®. The z/OS section includes sample batch scripts and JCL.

- [Major components of the SEPA component](#)
- [Running the SEPA component on z/OS](#)
- [Working with the default project](#)

The SEPA component is supplied with a default Transformation Extender project to help you configure the component into the ECLIPSE-based Design Studio.

## Major components of the SEPA component

When you install the Transformation Extender Pack for Financial Payments SEPA component, the following subdirectories are created under the <install\_dir>/packs/financial\_payments\_vn.n.n.n/sepa directory: examples, mapsandschemas,res, and type\_trees.

- [Examples](#)  
The examples directory contains examples for converting between SEPA formats and various national formats, as well as other parts of the SEPA component of the Pack for Financial Payments.
- [mapsandschemas](#)  
The **mapsandschemas** directory contains validation framework maps, schemas, and reference data files and maps.
- [res](#)  
The res directory contains files that can be used to display SEPA validation and error messages in alternate languages.

- [type trees](#)

The type\_trees directory contains type trees for the validation framework and corresponding BICPlusIBAN directory structures.

---

## Examples

The examples directory contains examples for converting between SEPA formats and various national formats, as well as other parts of the SEPA component of the Pack for Financial Payments.

These are the formats for the countries that are supported by examples in the SEPA component:

- Germany, DTAUS – see /examples/germany
- France, MINOS, CFONB – see /examples/france
- United Kingdom, BACS – see /examples/uk
- Italy, RNI – see /examples/italy

In addition to the format subdirectories, the examples subdirectory also contains the following subdirectories:

- passthrough subdirectory – see /examples/passthrough. Contains an example for validating a message against a schema in the z/OS® environment along with a sample of the JCL required to run it.
- pain\_pacs subdirectory – see /examples/pain\_pacs. Contains example maps used to create inter-PSP messages from customer-to-PSP messages and PSP-to-customer messages.
- sample\_jcl subdirectory – see /examples/sample\_jcl. Contains an example that shows how to run the SEPA component validation maps on z/OS.
- swiftfin subdirectory – see /examples/swiftfin. Contains an example that describes the SEPA component format converter for the SWIFTNet FIN format.
- epc\_mapsandschemas subdirectory. Contains an example validation map source that uses the EPC published schemas for the relevant payments clearing and settlement messages by using native schema mapping. These maps can be used as a replacement for the main validation maps to improve performance, since these maps use the EPC schema subset of the ISO20022 schema.
- bundle/unbundle - contains examples that demonstrate working with aggregate (bundle) SEPA data.

Note the following points about the examples:

- They are supplied as examples only, and are not to be considered a full set of all possible conversions. The domestic type trees that are supplied with the SEPA component support basic syntactic validation, but do not provide for full validation.
- The SEPA type trees that are used in the converters are the ISO20022 type trees. The transformation rules are best estimates. Pay particular attention to BIC and IBAN conversions, since these conversions might require the use of extra reference data that only you can access.

Instructions for running the examples are provided in the [SEPA component examples](#) documentation.

---

## mapsandschemas

The **mapsandschemas** directory contains validation framework maps, schemas, and reference data files and maps.

The components of the **mapsandschemas** subdirectory are described in this list:

- xliff.dtd file – Not included as part of the pack, and must be copied from ITX install directory into the **mapsandschemas** directory.
- epcl.txt file – contains the latest External Purpose Code List that is used to validate all Customer to PSP messages.
- eeaSEPACTryCd.txt file - contains the list of countries and territories which are part of the jurisdictional scope of the SEPA Schemes and their ISO country codes.
- errc.txt file - contains the latest External Reversal Reason Code that is used to validate all Customer to PSP Payment Reversal messages.
- **Validation framework maps** - perform validation of the SEPA message by checking for: 1) Basic UNIFI compliance and 2) EPC-mandated usage rules for Credit Transfer and Direct Debit. The main map for the validation framework (sepavalid) resides under the sepa\_validation.mms. All of the validation maps reside within this source file, so they can all be built at the same time, by using the Build All option in the Map Designer.
- **ISO 20022 Schemas** - All needed schemas are provided for the validation of all the different message types and levels (UNIFI, SEPA, CORE, and AOS). Any new or modified schema must be located here. Also, there is a SEPA generic schema (sepagen.xsd) that contains a list of all messages that are supported by the validation framework maps.
- sepa\_library.mms file – includes additional utility maps to work with the **BICPlusIBAN** directory.
- **Dependent EPC** - All dependent EPC are validated based on the Local Instrument Code.
- **Reference data files and maps** - The SEPA component comes with XML templates used by the validation framework for validating BICs, currency code, country code and allowed decimal places for currency amounts information. The SEPA component also contains maps to populate these templates with live data supplied by SWIFT (sepa\_library.mms).
- [BICPlusIBAN Utility](#)

---

## BICPlusIBAN Utility

The SWIFT BICPlusIBAN is a collection of institution identifiers from International, National directory providers and Financial institutions. This directory was created to improve straight-through processing (STP) by identifying correspondents and counterparties accurately. Additional utility maps are included in the sepa\_library.mms file to work with the **BICPlusIBAN** directory.

---

## res

The res directory contains files that can be used to display SEPA validation and error messages in alternate languages.

To display messages in one of the supported languages, refer to the following procedures:

1. Rename the existing sepamsgs.xml file located in the **mapsandschemas** directory to sepamsgs\_en.xml.
2. Copy the sepamsgs\_<language>.xml file for the desired language from the **res** directory to the **mapsandschemas** directory, naming the file to sepamsgs.xml.
3. Update the type tree errorxml.mtt located in the **type\_trees** directory. Change the property '**Data Language**' from '**Native**' to '**UTF-8**' for the following items:
  - Error/fields/action
  - Error/fields/explanation
  - Error/fields/msgText
4. Analyze and save the type tree.
5. Update the map **seplib05\_extract\_error\_msg** on sepa\_validation.mms located in the **mapsandschemas** directory. On output card#1 update the map rules on the following to replace TEXT function with CTEXT function using '**UTF-8**' as character set.
  - msgText
  - explanation fields
  - action fields

Re-build all the maps under sepa\_validation.mms

The following language files are available:

- sepamsgs\_de.xml - German
- sepamsgs\_es.xml - Spanish
- sepamsgs\_fr.xml - French
- sepamsgs\_it.xml - Italian
- sepamsgs\_ja.xml - Japanese
- sepamsgs\_ko.xml - Korean
- sepamsgs\_pt.xml - Portuguese
- sepamsgs\_zh-CN.xml - Simplified Chinese
- sepamsgs\_zh-TW.xml - Traditional Chinese

---

## type\_trees

The type\_trees directory contains type trees for the validation framework and corresponding BICPlusIBAN directory structures.

All schemas required for the validation framework have a corresponding tree in this subdirectory. These schema type trees are created using the type tree schema importer. A tree for the SEPA generic schema (sepagen.mtt) is also provided.

These type trees were imported using both the Xerces and the Classic importer.

---

## Running the SEPA component on z/OS

Before deploying the SEPA component of the Transformation Extender Pack for Financial Payments on the z/OS® platform, make note of the following factors:

- The SEPA framework consists of a main map and several executable run maps that use several schemas to validate UNIFI and SEPA.  
Note: EBA functionality is deprecated in this version of the Pack for Financial Payments.
- All compiled maps and schemas, as well as input XML files, must be ported in binary to the z/OS platform. The schemas and maps should reside in the same location, like a library dataset.
- The xliff.dtd from the base product directory must be ported in binary to the z/OS environment.
- All of the type trees in the type\_trees subdirectory that are related to a schema or DTD must be (manually) modified to include the DDNAME which will point to the corresponding schema or DTD.
- After modification of the type trees, the maps have to be built and then ported to z/OS.
- Every map, schema, and file that is ported to z/OS must have a unique 8-character DDNAME in the JCL.
- **Preparing work for execution on z/OS**  
In order to view the XML error log produced on z/OS, you can use 'ICONV', if available, or you can modify the JCL for the map run command to add '/VX15 name of third output card for the sepalid map'

---

## Preparing work for execution on z/OS

In order to view the XML error log produced on z/OS, you can use 'ICONV', if available, or you can modify the JCL for the map run command to add '/VX15 name of third output card for the sepalid map'

You might want to create a list of the maps, files, and schemas that will reside on z/OS. In the list, indicate the original file name, the type of file, the name that the file on z/OS, and the DDNAME. The following table shows how a typical list can be constructed. For a more complete description of all of the maps, files and schemas being deployed on z/OS, refer to the example FTP script and the example JCL contained in the examples/sample\_jcl directory.

| Original file name  | Type | File name on z/OS     | DDNAME   |
|---------------------|------|-----------------------|----------|
| pacs.002.001.10.xsd | map  | sepa.maplib(SE1B002C) | SE1B002C |
| xliff.dtd           | dtd  | xliff.dtd             | XLIFF    |

| Original file name       | Type       | File name on z/OS | DDNAME   |
|--------------------------|------------|-------------------|----------|
| currencycodedecimals.xml | INPUT CARD | currency.xml      | CURRENCY |

## Working with the default project

The SEPA component is supplied with a default Transformation Extender project to help you configure the component into the ECLIPSE-based Design Studio.

Perform the following steps to import the default project into your Transformation Extender Design Studio workspace:

1. From the Transformation Extender Design Studio menu bar, select File > Import. The Import window opens.
2. From the General folder, select Existing Projects into Workspace. Choose Next.
3. In the Import window, choose the Select root directory option, then use the Browse option to navigate to <install\_dir>/ packs.
4. From the list of existing projects, select financial\_payments\_vn.n.n.n/sepa, where n.n.n.n represents the current version of the Pack for Financial Payments. Select Ok.
5. Ensure that the Copy project to workspace check box is not checked. Choose Finish.

## Validation overview

One of the primary objects of the SEPA component is the validation framework map.

The validation framework map is designed to simplify the calling of validation steps that are provided by IBM and the optional integration of custom validation steps. This map orchestrates the validation flow and enables the logging to debug validation flow.

This validation framework map will take in any UNIFI 20022 formatted XML message and validate the set of rules published by the EPC. Default setting for UNIFI validation is OFF, that is '0'. To enable UNIFI validation, change input card 3 ValidateUNIFI on property SourceRule > GET > Source > Data from '0' to '1'. The validation framework is completely operated by the call of one map, **sepvalid**, which is discussed in detail in the following section.

It should be noted that no matter what the SEPA message type, the validation framework map passes correct data.

In order for a SEPA message to pass through the validation framework without error, the following conditions must be met:

- the XML must be well-formed
- all published applicable UNIFI rules must be followed
- all published applicable SEPA IGs and rulebooks must be enforced
- XML must validate against the published UNIFI/SEPA schemas

Optionally, the validation framework allows for validation of a message against an original SEPA message, where appropriate. The SEPA implementation guides have rules that require the original payment information to be validated in the current data document. Specifically, the contents of the Original Transaction Reference must match the original transaction information. For example the **pacs004** return refund and **pacs002** reject, for both credit transfer transactions and direct debit collections. The **pacs007** direct debit reversal also has this requirement. For all of these cases, the original message is the **pacs008** FI to FI customer transfer, or the **pacs003** FI to FI customer direct debit.

- **sepvalid**

This validation framework will take in any UNIFI 20022 formatted XML message and validate the set of rules published by the EPC. Default setting for UNIFI validation is OFF, that is '0'. To enable UNIFI validation, change input card 3 ValidateUNIFI on property SourceRule > GET > Source > Data from '0' to '1'.

- **SEPA validation customization**

Within the SEPA validation framework maps, you can customize validation for the following:

- **Unenforced usage rules**

The Validation Framework does not enforce usage rules that involve uniqueness over time, because the SEPA component does not provide a full end-to-end view of all the messages handled.

## sepvalid

This validation framework will take in any UNIFI 20022 formatted XML message and validate the set of rules published by the EPC. Default setting for UNIFI validation is OFF, that is '0'. To enable UNIFI validation, change input card 3 ValidateUNIFI on property SourceRule > GET > Source > Data from '0' to '1'.

The framework is completely operated by the call of one map, **sepvalid**. No matter what SEPA message type, this map will pass the data through the validation framework. The framework has the ability to determine if the data is ISO 20022 UNIFI schema data, or if the data is part of the EPC Technical Validation Set, which has two sets of schemas, CORE schemas, which allow only the fields that are required by SEPA, and the CORE plus AOS, which allow additional elements. Custom schemas can also be added.

The main map that is called is **sepvalid**. Three inputs are allowed:

- The first input will be the SEPA message that needs to be validated.
- The second input will be the original message. If no original data file is provided, then the matching rules in the SEPA implementation guide will not be processed.
- The third input will be to enable or disable the UNIFI rule validation. Default setting for UNIFI validation is OFF, that is '0'. To enable UNIFI validation, change input card 3 ValidateUNIFI on property SourceRule > GET > Source > Data from '0' to '1'.

There are 3 Output cards for the **sepvalid** map. Cards one and two do not produce output files, and card three produces the **seperror.xml** file that allows you to see all of the validation errors and framework messages produced from the data file. If debugging of the process is required, change the output for card 1 from SINK to File and assign it a file name. This debug file will then serve as a debugging tool, as it contains steps that document where in the process it has been, and what was performed.

## SEPA validation customization

Within the SEPA validation framework maps, you can customize validation for the following:

- Add additional error reporting, or add checks that are outside of the scope of the SEPA or UNIFI schemas
- Add validation for a custom schema
- Change individual validation rules, which is especially useful for those rules in the customer-to-PSP messages that are only "recommended" by SEPA

## Unenforced usage rules

The Validation Framework does not enforce usage rules that involve uniqueness over time, because the SEPA component does not provide a full end-to-end view of all the messages handled.

Rules that imply that a field can only be used if there is agreement between originator and beneficiary are also not enforced. The following SEPA rules are involved:

| SEPA message  | Element                               | Rule                                                                                                                                                                              |
|---------------|---------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| pacs.008.(CT) | Transaction Identification            | Must contain a reference that is meaningful to the Originators PSP and is unique over time.                                                                                       |
| pacs.008.(CT) | Remittance Information - Unstructured | Unstructured may carry structured remittance information, as agreed between the Originator and the Beneficiary.                                                                   |
| pacs.002(CT)  | Original Transaction Identification   | Must contain a reference that is unique over time.                                                                                                                                |
| pacs.004(CT)  | End To End Identification             | A customer reference that must be passed on in the end-to-end payment chain. If no reference was given, 'NOTPROVIDED' must be used.                                               |
| pain.001(CT)  | Remittance Information - Unstructured | Unstructured may carry structured remittance information, as agreed between the Originator and the Beneficiary.                                                                   |
| pacs.003.(DD) | Transaction Identification            | Must contain a reference that is meaningful to the Creditors PSP and is unique over time.                                                                                         |
| pacs.003.(DD) | Remittance Information - Unstructured | Unstructured may carry structured remittance information, as agreed between the Creditor and the Debtor.                                                                          |
| pacs.003(DD)  | End-To-End Identification             | A customer reference that must be passed on in the end-to-end payment chain. If no reference was given, 'NOTPROVIDED' must be used.                                               |
| pacs.004.(DD) | Transaction Identification            | Must contain a reference that is meaningful to the Creditors PSP and is unique over time.                                                                                         |
| pacs.004(DD)  | Original Transaction Identification   | Must contain a reference that is meaningful to the Creditor's PSP and is unique over time.                                                                                        |
| pacs.007.(DD) | Original Transaction Identification   | Must contain a reference that is meaningful to the Creditors PSP and is unique over time.                                                                                         |
| pain.008(DD)  | Remittance Information - Unstructured | Unstructured may carry structured remittance information, as agreed between the Creditor and the Debtor.                                                                          |
| camt.029(CT)  | Additional Information                | Assumption is Additional Information (AddtlInf) coming in camt.029 is based on original message camt.056. Cross verification to original/associated camt.056 cannot be performed. |

## Validation details

The following topics contain a detailed discussion of the SEPA component validation processes. It provides:

- a description of the maps inside the validation framework
  - a detailed description of validation framework map naming conventions
  - instructions on customizing validation steps
  - a discussion of how failures and unexpected results should be handled.
- [\*\*Maps inside the validation framework\*\*](#)  
One purpose of the validation framework is to simplify the calling of the IBM-provided validation steps in the SEPA component of the Pack for Financial Payments.
- [\*\*Validation framework map naming conventions\*\*](#)  
To navigate easily through the maps in the validation framework, it is important that you understand the map naming conventions.
- [\*\*How to run SEPA Compliance in Design Studio\*\*](#)
- [\*\*How to run the SEPA Compliance in Design Server\*\*](#)
- [\*\*Types of customization for SEPA validation\*\*](#)  
This section describes how to add a new schema or validation step, modify a validation step, and modify or add validation rules.
- [\*\*Handling failures and unexpected results\*\*](#)  
Between the validation framework built-in debugging tools, and general ITX debugging techniques, it is possible to obtain information about map and data format failures, as well as and unexpected results.
- [\*\*BIC plus IBAN validation and lookup\*\*](#)  
Proper handling of bank identification with either the Bank Identification Code (BIC), or International Bank Account Number (IBAN) is required throughout the SEPA process, a process that includes validation, domestic format conversion, and client data mapping..

## Maps inside the validation framework

One purpose of the validation framework is to simplify the calling of the IBM-provided validation steps in the SEPA component of the Pack for Financial Payments.

In the world of SEPA/XML, the information needed to determine the cause of invalid data may not always be straightforward, and at times, it might require several levels of checking in order to achieve meaningful error reporting.

- [Stepped map process](#)

## Stepped map process

The stepped map process validates the data against different criteria. For example, the first step may be to validate against a schema and the second step may be to validate against some additional rules. Each map step reports any errors that are found. If the data is good, according to all the map steps performed, then no output is produced by the validation framework. If the data fails a map step, one or more errors will be produced from that step. It is important to understand the independent nature of the map steps, and that each step is designed to look for certain failures. In some cases, data may fail on multiple mapsteps; a failure of an enumeration value in a schema may also present itself as a failure in the map rules.

For example, to process a **pacs\_008\_001\_08** message, the message is validated against the following:

- **pacs\_008\_001\_08 schema**
- SEPA rules not in the schema
- UNIFI rules (Optional feature)

## Validation framework map naming conventions

To navigate easily through the maps in the validation framework, it is important that you understand the map naming conventions.

The naming convention used in the validation framework maps is a combination of an eight character coded reference, plus a string that matches the SEPA schema naming convention. When looking at the validation framework map names, notice that periods are replaced by underscores (`_`).

The first eight characters are significant, as they must be unique for mainframe processing. The naming scheme for the first eight characters in the validation framework map names is as follows:

**"se" + map level + PSP/customer + message + ct/dd**

This is illustrated in the following example. Data conforming to **pacs\_002\_001\_10 FIToFIPmtStsRpt** schema to be validated against the Credit Transfer Implementation Guide would first pass through the level one map named:

**se1b002c\_FIToFIPmtStsRpt**

**"se" + map level="1" + PSP/customer="PSP"="b" + message = "002" + ct/dd="ct"="c"**

The map name is shown below with the first eight characters indicated:



Characters 1 through 8 are defined as follows:

- **Characters 1 and 2** – These characters always begin with the literal **se**.
- **Character 3** – This character indicates the validation framework mapping level. It may be a **0**, or a **1**. There are two validation framework mapping levels:
  - **Level-zero maps** – At the highest level are framework level maps, called level-zero maps. A level-zero map always contains a **0**.
  - **Level-one maps** – At the next level of detail are validation level maps, or level-one maps. These maps perform specific validation for the messages. A level-one map always contains a **1**.
- **Character 4** – This will be character **b** to indicate the message is PSP-to-PSP. Character **c** to indicate the message is customer-to-PSP and character **e** to indicate the message is an e-Mandate.
- **Character 5, 6 and 7** – These characters indicate the UNIFI message name.
- **Character 8** – This character indicates whether the message is used for a credit transfer, or for a direct debit. Credit transfers are indicated when a **c** appears in this character position. A direct debit is indicated when a **d** appears in character position 8. Characters **e**, **f**, **g**, **h**, **i**, **j** and **m** are variations of map names.

The remainder of the map name (i.e., `_FIToFIPmtStsRpt`) is a direct transformation of the primary schema name.

## How to run SEPA Compliance in Design Studio

These steps describe how to run the SEPA Compliance component in Design Studio.

1. Copy xliff.dtd file from ITX install directory into the mapsandschemas directory.
2. Use the Design Studio to open and build the maps on the following map source files.
  - sepa\_error\_parser.mms
  - sepa\_library.mms
  - sepa\_validation.mms
3. Run the following map from sepa\_validation.mms map source file.
  - sepalid

# How to run the SEPA Compliance in Design Server

These steps describe how to run the SEPA Compliance component in Design Server.

1. Import the sepaCompliance.zip file into the Design Server UI. See the release notes for import instructions.
  2. Create a package for the project and build all the maps.
  3. Open the sepaCompliance project in the Design Server UI.
  4. Run the sepalid
  5. View the output. Go to the map Diagram-Result, and hover the cursor over Target, then select **View Data**.
  6. To download the output file, see [How to download output files](#).
- [How to download output files](#)  
After you build/run maps, download the output files.

## How to download output files

After you build/run maps, download the output files.

### About this task

To download the output file, go to **Files** in the Design Server UI and download the desired file using these steps:

### Procedure

1. Click the output Card Settings you want to download.
2. From the Command text box, copy the name of the file that you want to download. Make certain that you copy only the file name.  
For example, if the Command text box displays:  
  
data/pain008d.xml  
  
copy only:  
  
pain008d.xml
3. From Files, click the New icon to open the New File dialog box.
4. Paste the file name in the Name field.
5. Click the Folder field drop down and select data.
6. Click Ok.
7. Build and re-run the map.
8. Go to Files and select Download to view the data.

## Types of customization for SEPA validation

This section describes how to add a new schema or validation step, modify a validation step, and modify or add validation rules.

Note: Customization requires modification of the SEPA component validation maps. Make a copy of the validation maps prior to customization to avoid the need to reinstall the SEPA component of the Pack for Financial Payments, if you need to go back to the original maps.

- [Adding new schemas](#)  
If you are not using the published ISO 20022 UNIFI schemas or the EPC revised schemas published for SEPA, you will have to add any customized schemas to the framework.
- [Adding new validation steps](#)  
If you are using customized SEPA schemas, you must customize the framework, and possibly the validation maps, that the validation framework calls.
- [Modifying validation steps](#)  
These steps describe how to modify validation steps.
- [Modify or add validation rules](#)  
In many cases, you will have to either add additional validation rules, or perhaps in the case of the **pain.xx** message types, where all of the recommended rules in the SEPA implementation guidelines are included, you may need to remove some rules.

## Adding new schemas

If you are not using the published ISO 20022 UNIFI schemas or the EPC revised schemas published for SEPA, you will have to add any customized schemas to the framework.

This might be necessary if you are implementing recommended and mandatory fields in the customer-to bank messages. It might also be appropriate if the message is being sent to another member of an AOS community, and the schema has been modified to more specifically define those message element fields that are permitted by SEPA. This process requires tree and map modifications.

Follow these procedures to add a new schema:

1. Add the new schema name to the seagen.xsd schema located in the mapsandschemas directory, and save the file. This is a simple schema, and contains a choice group with the list of schemas currently used by SEPA. A new element can be inserted as follows:

```
<xss:element name="custom_schema_name" type="xs:string">
```

2. Open the Type Designer and modify the existing seagen file by importing the seagen.xsd schema that you edited in step 1 as a Classic type tree.

3. Place the modified type tree in the type\_trees directory.

4. Check to make certain that your customized schema is located in the mapsandschemas directory.

Note: You must edit seagen after it is created to remove any restriction lists that may have been created by the import process. At this point in the validation, we need to treat the bulk of the instance document as a blob of data (disregarding XML tags), until it is passed to the lower level maps that perform the validation

The previous steps will allow an instance document based on your new schema to be processed by the validation framework. However, to perform specific validation, you must add a new validation step for this schema. See [Adding new validation steps](#).

## Adding new validation steps

If you are using customized SEPA schemas, you must customize the framework, and possibly the validation maps, that the validation framework calls.

Here you will find a description of how to add new schemas or customize existing ones. Even if you are using only the existing SEPA schemas, you may want to add a new map step. One example could be that your overall implementation requires you to process SEPA messages that only contain a certain number of payment transactions, which is less than the number allowed by SEPA.

Creating a new map step involves updating the validation framework maps to:

- Optionally extract a subset of information from the message to make downstream mapping more straightforward, with the use of a NVP WorkTemp. See [About NVP WorkTemps\(s\)](#).
- Set up the new step as a call to a functional map.
- [About NVP WorkTemps\(s\)](#)  
The NVP WorkTemp(s) are used to extract information from the data, and from previous steps, without the need for excess hierarchical functional map calling structure.
- [Add a custom schema](#)

## About NVP WorkTemps(s)

The NVP WorkTemp(s) are used to extract information from the data, and from previous steps, without the need for excess hierarchical functional map calling structure.

In other words, there are some things that you cannot easily do all on one map rule. NVP WorkTemp(s) are found at both the **sevalid** map level and the zero level maps (for example, **se0b002d\_FIToFIPmtStsRpt**). As a general rule, if the information being extracted in a NVP WorkTemp(s) is useful to most of the messages, it should be extracted at the sevalid map level and passed into the zero level maps as parameters. In the case of adding new custom mapsteps, if you are adding multiple steps, and this information will be used in all the validation maps for those steps, extract at the **sevalid** map level. For new steps, however, many of the anticipated new pieces of information needed are expected to pertain just to that new mapstep. In this case the NVP WorkTemp(s) in the new step should be used.

## Add a custom schema

The process for adding a custom schema uses the **sevalid** map. This process adds a new run map call to the existing validation maps, or could call a new or modified validation map. Follow the procedures below to add a validation step.

1. Open **sepa\_validation** in the map designer, then select output card 1 of the **sevalid** map.
2. If desired, create a new NVP WorkTemp under the Global WorkArea. This can then be passed to any of the new validation steps created.
3. Add an index to the FileType(s) group. To do this, first go to the bottom of the **sevalid** map. Right click on the **FileType(s)** and add an index. This index will have a corresponding call to a level zero (se0) functional map that will call the validation level one (se1) run map for validation. The corresponding level zero map can also contain framework rules.
4. In the FileTypes element group, complete all elements as needed for the new FileType. The following table provides a description of the table groups and associated elements.
5. Create the required functional maps and run maps to perform the validation required. You may want to copy other maps within the validation framework as a starting point.
6. If you want to use an NVP WorkTemp at the functional map level, these are found in the output card as an indexed group under the Step WorkArea element.

| Group                         | Action                                                                                                                                                                                                                                          |
|-------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| StepDescription Element Group | Add a description of the step.                                                                                                                                                                                                                  |
| StepWorkArea Element Group    | Create indexes as needed in the <b>NVP WorkTemp</b> .                                                                                                                                                                                           |
| Operation Group               | <b>RUNCall</b> Element = Used for running map commands.<br><b>PUTCall</b> Element = Used for putting files or queues etc., using an ITX adapter.<br><b>DBCall</b> Element = Used for connecting to an external database for framework activity. |
| Step Results Element Group    | Used for reporting step results.                                                                                                                                                                                                                |

## Modifying validation steps

These steps describe how to modify validation steps.

1. Navigate to the desired functional map in **sevalid** that needs to be updated. Notice that the functional map names begin with **se0**. The framework maps all have this naming convention, while the validation run maps all start with **se1**. These maps are also referred to as level zero and level one maps.  
Note: The 8 character prefix name for all of the maps must be consistent in order to facilitate deployment to z/OS.
2. Add an index to the **STEP(s)** element. The **STEP (s)** element is always at the bottom of the output card in the functional map.
3. In the **STEP [n]** element, complete the following groups and associated group elements as described in the table in the **Adding a validation step** section.

## Modify or add validation rules

In many cases, you will have to either add additional validation rules, or perhaps in the case of the **pain.xx** message types, where all of the recommended rules in the SEPA implementation guidelines are included, you may need to remove some rules.

Refer to the following section **Guidelines for modifying or adding validation rules** when modifying or adding validation rules. To add error messages, refer to the **Adding error messages** section.

- [Guidelines for modifying or adding validation rules](#)  
Observe these guidelines when modifying or adding validation rules.
- [Adding error messages](#)  
All of the rules call the **seplib05\_extract\_error\_msg** functional map. This map is used for the framework and the validation maps to create error messages in the **seperror.xml** output file.

## Guidelines for modifying or adding validation rules

Observe these guidelines when modifying or adding validation rules.

- All of the rules are in the **se1\*** run maps.
- Rules are added by creating a 'Rule' index.
- Rules are organized in the order that they appear in the SEPA Implementation Guides.
- Comments in each existing rule refer to the correct index in the SEPA Implementation Guides.

## Adding error messages

All of the rules call the **seplib05\_extract\_error\_msg** functional map. This map is used for the framework and the validation maps to create error messages in the **seperror.xml** output file.

Two values are echoed. The first value in this rule, for example **SEP0019E**, corresponds to the error number in the **sepamsgs.xml** file that contains all of the error messages used in the framework and validation maps. The second value is passed into the {0} variable in the **sepamsgs.xml**, and can be any text string data.

When you want to add additional error messages to the **sepamsgs.xml**, make certain that you add them in the same format as the existing messages. Each message will have 3 trans-unit-ids:

- Error Message (EM)
- Extended Error (EE)
- Error Action (EA)

String values from the map are passed into the EM id using the {0} variable. Any additions to this file must be added in ascending trans-unit id order.

## Handling failures and unexpected results

Between the validation framework built-in debugging tools, and general ITX debugging techniques, it is possible to obtain information about map and data format failures, as well as and unexpected results.

The first step that should be taken if any unexpected results are encountered, is to consult output card 3 (**seperror.xml** by default). This output contains an XML structure that describes any framework or validation errors in end-user terms. The next section describes output card 3 validation errors. If output card 3 is checked, and the reason for the failure is not apparent, consult output card 1. Refer to the Output card 1 section for details of this procedure.

For a description of the SEPA error parser and instructions on how to run the **sepa\_error\_parser** utility map, see [SEPA schema error parser \(Deprecated\)](#).

- [Checking output card 1](#)
- [Checking output card 3](#)
- [SEPA schema error parser \(Deprecated\)](#)

## Checking output card 1

If the reason for the failure is still not apparent after consulting output card 3, check output card one. First, create this file, if calling **sevalid** from outside of Map Designer, or in the Map Designer change the output card settings from **sink** to **file** and specify an output file location (a suggested name is **out.xml**). By default, this card is suppressed (card setting of **sink**) from creation. This XML instance document will contain a log of all processing during the run of the **sevalid** map. It will also list all of the temporary variables used and show the various map steps performed and their results.

You will be able to use this output to track the steps of the framework and determine on which map the failure is occurring, or the unexpected results are being introduced.

## Checking output card 3

Output card 3 shows the name of the file being validated, the name of the (optional) original file to be validated, the start and end date/times and any errors. If there are no errors reported by the validation framework for a map run then this output will not be created.

- [Example of output card 3 validation errors](#)

## Example of output card 3 validation errors

Output card 3 validation errors are illustrated in the following example:

```
<MapInfo>
<FileToValidate>/opt/data/hipfiles/5bd883af2ab79c00a1168c0b/5c07ea2c2ab79c00272650f0/maps/./data/pacs812c.xml</FileToValidate>
<OriginalFile>/opt/data/hipfiles/5bd883af2ab79c00a1168c0b/5c07ea2c2ab79c00272650f0/maps/./data/sep000.xml</OriginalFile>
 <Start>20181206154248</Start>
 <End>20181206154248</End>
 <Validation>
 <Description refId="PmtInfId:SUB - EndToEndId:SUB">
 <Message
id="_home_markdown_jenkins_workspace_Transform_in_SSVSD8_11.0.1_com.ibm.help.spe_fsp_pk.doc_sepa_901_concepts_c_pack_sepa_outpu
t_card_3_example_SEP0155E">Payment Type Information must be present in the 'FIToFICstmrCdtTrf/GrpHdr/PmtTpInf |'
FIToFICstmrCdtTrf/CdtTrfTxInf{1}/PmtTpInf', but not both</Message>
 <Explanation>Payment Type Information must be present at either the Payment Information level or in the
Credit Transfer Transaction Information level, but not both</Explanation>
 <Action>Ensure the Payment Type Information is present either in Payment Information level or the
Credit Transfer Transaction Information level, but not in both levels</Action>
 </Description>
 </Validation>
</MapInfo>
```

## SEPA schema error parser (Deprecated)

If you encounter a SEPA schema validation error when SEPA validation is performed using the maps in sepa\_validation, you are presented with a SEP0300E or SEP0301 error message in the error log. There are many elements in these schemas that SEPA does not use. In this event, only the SEP0300E or SEP0301E errors will appear in the error log. To help to determine what these errors are, you can use the sepa\_error\_parser utility map.

The sepa\_error\_parser runs the SEPA data file through schema validation with the TX trace option turned on. In order to expedite diagnosis of errors, this map will create a subset data file containing only the bad transactions. The XML trace log will be parsed, which will then be used by this framework to retrieve the error data information for the input data. The list of data errors will be presented as the output to the framework, which will show the line numbers of the data that is in error, as well as the element that failed validation. After the framework completes the process, refer to the subset bad data file which will be presented as card 4 in the utility map (errordat.xml) and the error log to correct schema errors in the data.

- [Running the error parser](#)

## Running the error parser

Follow these instructions to run the **sepa\_error\_parser** map:

1. Pass your SEPA data file into the **errparse** run map with an override to input card 1.
2. Retrieve the error log, which will be output card 3. The contents will refer to the subset data file that was used to create the error log. Use this subset data file to locate and correct the schema errors.

Note: If you are a z/OS user, you must modify the **sepa\_error\_parser.jcl** located in the **mapsandschemas** directory. You can then deploy to your system.

## BIC plus IBAN validation and lookup

Proper handling of bank identification with either the Bank Identification Code (BIC), or International Bank Account Number (IBAN) is required throughout the SEPA process, a process that includes validation, domestic format conversion, and client data mapping..

Although the IBAM has the same function in multiple countries, the components that comprise the IBAN can vary in length, or in other requirements. This increases the complexity involved in SEPA implementations.

- [The BICPlusIBAN directory](#)

In January 2008, SWIFT enhanced the BIC Database Plus directory to include IBAN information and published a revised directory structure with additional fields. The BIC Database Plus directory is now called the BICPlusIBAN directory.

- [The BICPlusIBAN utility](#)

The BICPlusIBAN utility provides a way to look-up BIC or IBAN information in the published directory. This utility can be used within domestic converter maps based on the samples provided in the pack, or implemented in an application map separate from individual message processing.

- [Turning BIC and IBAN checking on and off](#)

You are able to disable both the BIC and the IBAN checking from within the SEPA validation framework.

## The BICPlusIBAN directory

In January 2008, SWIFT enhanced the BIC Database Plus directory to include IBAN information and published a revised directory structure with additional fields. The BIC Database Plus directory is now called the BICPlusIBAN directory.

The BICPlusIBAN directory provides data for financial institutions in all SEPA countries. Through this directory, all BICs and IBANs are systematically checked against the latest versions of both the ISO BIC directory and the ISO IBAN registry, for which SWIFT is the registration authority. SWIFT validates the data with financial institutions, national banks and banking institutions. Additionally, cross checks against data from multiple sources improve the quality and consistency.

The BICPlusIBAN directory helps you to perform the following tasks:

- increase STP rates
- lower the cost associated with handling rejected messages
- comply with SEPA mandate requiring the use of IBAN and BIC
- provide better service to your clients by repairing incomplete messages

## The BICPlusIBAN utility

The BICPlusIBAN utility provides a way to look-up BIC or IBAN information in the published directory. This utility can be used within domestic converter maps based on the samples provided in the pack, or implemented in an application map separate from individual message processing.

For example, the utility could be used within an application map to prepare, in advance, a subset of the BIC-IBAN information if not all of the countries or institutions represented in the directory were required in a particular SEPA implementation.

For any IBAN specified in a payment, directed to any SEPA country, by using the utility along with the BICPlusIBAN directory you are able to lookup the associated BIC to be used in the SEPA payment. The lookup function also returns those BICs that cannot be simply derived from the national bank code included in the IBAN. You can also validate the BICs, and the IBANs and their relationships in your outgoing SEPA payments. The utility is based on the specifications provided by SWIFT to accommodate the varying lengths and other requirements of the fields within the IBAN. Conversely, you can also use the utility to do a BIC lookup based on a particular IBAN.

- [BICPlusIBAN support](#)

These are the BICPlusIBAN files in the SEPA component, with a description, related type trees, and scenarios illustrating which maps to use.

## BICPlusIBAN support

These are the BICPlusIBAN files in the SEPA component, with a description, related type trees, and scenarios illustrating which maps to use.

## BICPlusIBAN files

BICPlusIBAN file name	Description	Type trees
bi.txt	BI (BICPlusIBAN) file, gives information about financial institutions and their IBAN-related data	bi
is.txt	IS (IBAN structure) file, gives information about IBAN structure applicable in the countries	is
ct.txt	CT (Countries) file, gives information about ISO countries	ct
cu.txt	CU (Currencies) file, gives information about ISO currencies	cu

Note: The files defined above are examples used to show the capabilities of the BICPlusIBAN utility maps. For complete and current versions of the BICPlusIBAN files, download the files from SWIFT.

## Scenarios

The table below presents various scenarios and shows which map to use.

Map name	Scenario
seplib07_derive_bic_from_iban	IBAN is present but BIC is missing in a SEPA payment instruction.
seplib08_validate_bank_id	Ordering customer has constructed the IBAN, and the Bank ID contained in the IBAN needs to be validated.
seplib09_validate_bic	Ordering customer attempted to derive the BIC itself from financial institution's name and address and needs to validate that the BIC is a valid BIC.
seplib10_validate_bic_ibancomb	Verify if the BIC and IBAN belong to one and the same institution
seplib11_validate_ct	Lookup country codes.
seplib12_validate_cu	Lookup currency codes.

## Technical specification for BICPlusIBAN

A technical specification called TECH\_SPEC\_BI.pdf can be downloaded from SWIFT. The information in this specification is the basis of the functions of the BICPlusIBAN utility maps.

## Turning BIC and IBAN checking on and off

You are able to disable both the BIC and the IBAN checking from within the SEPA validation framework.

The following sections describe how to disable these validations.

- [Disabling BIC validations](#)

Perform the following steps to disable BIC validations.

- [Disabling IBAN validations](#)

Perform the following steps to disable IBAN validations,

## Disabling BIC validations

Perform the following steps to disable BIC validations.

### Procedure

1. Open the sepa\_validation source file and locate the following f functional maps:
  - f\_bic\_searchup
  - f\_bic\_searchup\_unifi
2. Change the map rule to =NONE.
3. Comment out the remainder of the rule as noted below:

### Example

```
=NONE
/*f_bic_level2
 ("SEP0002E", EITHER (SEARCHUP (subtype Attr:AttrList:ElemDecl BIC:BICCodes Element:Global:BicList, id
 Attr:AttrList:ElemDecl BIC:BICCodes Element:Global:BicList, CTEXT (BIC, "NATIVE")), "n/a"),
 CTEXT (BIC, "NATIVE") ,
 BicList ,
 path,
 refId)*/
```

## Disabling IBAN validations

Perform the following steps to disable IBAN validations,

### Procedure

1. Open the sepa\_validation source file and locate the f\_format\_IBAN functional map.
2. Change the map rule to =NONE.
3. Comment out the remainder of the rule as noted below:

### Example

```
= NONE
/*f_sub_letter(MID (IBAN, 5, SIZE(IBAN)) + MID (IBAN , 1, 4),
ErrMsg,
path,
CTEXT (OriginalIBAN , "NATIVE"),
RefIds)*/
```

## Domestic format support

The Transformation Extender SEPA component provides domestic support for the following countries.

- the United Kingdom

- Germany

- Italy

- France

- [Domestic format for the United Kingdom](#)

This section describes the BACS domestic format converter for the United Kingdom (U.K.).

- [Domestic format for Germany](#)

The DTAUS data format was defined within the overall online banking standard that was announced in 1995 by the German banks represented by their associations participating in the Central Banking Committee (ZKA - Zentraler Kreditausschuss).

- [Domestic format for France](#)

There are two domestic format converters for France:

- [\*\*Domestic format for Italy\*\*](#)

The domestic format example for Italy includes maps, trees, and data used to convert SEPA data to SIA Rete Interbacaria (RNI) Italian domestic format, and conversion from the domestic formats to SEPA.

- [\*\*SEPA format converters for SWIFTNet FIN\*\*](#)

This documentation describes the SEPA format converter for the SWIFTNet FIN format.

---

## Domestic format for the United Kingdom

This section describes the BACS domestic format converter for the United Kingdom (U.K.).

Note: This format is available for purchase from APACS at [www.apacs.org.uk](http://www.apacs.org.uk), and references Standard 18 - 1 October 2002.pdf BACS Interchange Standards (v18/9 dated 1 October 2002)

The BACS standard contains two formats for data records: BACS input and BACS output. Both of these formats have the same basic field structure of 100 bytes, but the BACS output format is extended by additional fields. The BACS standard also defines Volume and File and User Header records which vary in size. The BACS input and output formats are described in as follows:

- **BACS input** - This format is used by Banks, and their customers, to send payment data to BACS by electronic transfer, or other means. After initial validation, the data is forwarded to the relevant bank(s) using BACS output format. The BACS input format can be either 100 or 106 bytes. The additional 6 bytes are used to specify individual processing dates within BACS. For full details on this process, refer to the BACS User Manual.
- **BACS output** - The BACS output is always 120 bytes. The additional 20 bytes contain fields added after validation by BACS: Error Code, BACS User Number and BACS Reference (unique reference for each payment; used by BACS for query purposes).

The SEPA type tree and converter maps are based upon the output format. If conversion to and from the input format is required, it can be easily achieved since the base 100-byte format is identical, although the extra data provided by the BACS output format is useful in providing additional unique identifiers for each transaction.

There are several types of data records in BACS format. These data records are identified by Transaction Code (TX Code) and vary depending upon whether the sender is a Bank or a Bank's Customer. The SEPA type tree contains a single file description containing all possible record types.

- [\*\*Domestic format example for the United Kingdom\*\*](#)

Example files that describe the United Kingdom domestic format are located in the <install\_dir>/packs/financial\_payments\_vn.n.n.n/sepa/examples/uk directory.

---

## Domestic format example for the United Kingdom

Example files that describe the United Kingdom domestic format are located in the <install\_dir>/packs/financial\_payments\_vn.n.n.n/sepa/examples/uk directory.

See the [Example for the UK](#) documentation for more information.

---

## Domestic format for Germany

The DTAUS data format was defined within the overall online banking standard that was announced in 1995 by the German banks represented by their associations participating in the Central Banking Committee (ZKA - Zentraler Kreditausschuss).

The full standard is the "Homebanking Computer Interface (HBCI)". This format is used for both interbank and customer-bank payments processing in Germany.

In the DTAUS format, a physical file contains:

- a single file header also called Record A
- one or more payment exchange messages called Record C
- a single file trailer called Record E

There is a single format used for both direct debit or credit transfer, identified by Record A (File Header) Field 3 (Identifier). However, a logical file must contain only credits or only direct debits (no mixed payment records). Record A contains the sender and receiver of the file, and is a fixed length of 128 bytes. Each Record C contains details of the orders to be executed (credits or debits) and contains a constant and a variable length section. Record E is used for performing checks.

- [\*\*Domestic format example for Germany\*\*](#)

Example files that describe the domestic format for Germany are located in the <install\_dir>/packs/financial\_payments\_vn.n.n.n/sepa/examples/germany directory.

---

## Domestic format example for Germany

Example files that describe the domestic format for Germany are located in the <install\_dir>/packs/financial\_payments\_vn.n.n.n/sepa/examples/germany directory.

See the [Example for Germany](#) documentation for more information.

---

## Domestic format for France

There are two domestic format converters for France:

- MINOS

- CFONB
  - [\*\*CFONB domestic format converter for France\*\*](#)  
The CFONB organization defines additional formats used in payments processing for the Customer-Bank processes.
  - [\*\*MINOS format converter for France\*\*](#)  
The MINOS domestic format converter for France is defined in the Handbook of SIT Interbank Transaction Standards, and is administered by GSIT, the French Interbank Teleclearing System.
  - [\*\*Domestic format example for France\*\*](#)  
For the SEPA converter examples for France, in which two type trees are provided.
- 

## CFONB domestic format converter for France

The CFONB organization defines additional formats used in payments processing for the Customer-Bank processes.

This format, used in the examples for France, is described in *Remises informatisees d'ordres de paiement International, au format 320 caractères*. This document is available through a catalogue on the web site: [www.revue-banque.fr/editionCatalogue.do?shortcut=edition\\_catalogue](http://www.revue-banque.fr/editionCatalogue.do?shortcut=edition_catalogue). This format will be replaced by an XML based format on which the French banking community works currently, and by the SEPA PAIN (payments initiation) format in the case of SEPA payment operations.

It should be noted that the CFONB format is close to the MINOS format, but each bank has the freedom to adapt the format to its specific needs.

- [\*\*Payment message record types\*\*](#)  
Each CFONB320\_PI payment message is identified by the value of PI in positions 3 to 4 of each record and is composed of the following record types.
- 

## Payment message record types

Each CFONB320\_PI payment message is identified by the value of PI in positions 3 to 4 of each record and is composed of the following record types.

- Mandatory header line identified by the first two characters of: 03.
- Group(s) of detail records consisting of the following:
  - Mandatory detail line identified by the first two characters of: 04.
  - Optional (dependent) line for details related to the Beneficiary Bank. The first two characters are: 05.
  - Optional line for details related to the Intermediate Bank. The first two characters are: 06.
  - Optional line for additional information related to the transfer. The first two characters are: 07.
- Mandatory total line. The first two characters are: 08.

## MINOS format converter for France

The MINOS domestic format converter for France is defined in the Handbook of SIT Interbank Transaction Standards, and is administered by GSIT, the French Interbank Teleclearing System.

This handbook contains the structure of the transactions used for interbank payment information. This document is available to order from the web site: [www.gsit.fr/Gb/information/minos.htm](http://www.gsit.fr/Gb/information/minos.htm).

For the MINOS format, there are various payment types handled through the current system, but only a very few correspond to the payments processing defined by the SEPA rules and datasets. The first payment types of interest are the Virements (sweeps, credit transfers), which are the most common kind of operations handled through the interbanking in France. These map to SEPA Credit Transfer processing. The credit transfer is identified by the 'code opération' in the SIT part of the record = 120. The second payment type of interest is the Prelevement (direct debit) which corresponds to SEPA Direct Debit processing. Different types of direct debits exist, however, the direct debit example included with the SEPA component of the Transformation Extender Pack for Financial Payments uses the message with a 'code operation' = 180 ('Prélèvements 4 jours').

MINOS distinguishes two message directions:

- 'Aller' – from the Bank to the Clearing House (ACH)
- 'Retour' – from the Clearing House to the Bank

Each of these directions can handle different types of operations (Credit transfer, Direct Debit)

'Remise aller' is a payment file containing the following:

- A header record
- 1-N payment operations of a same category and for the same settlement date:
  - Same category means: same type of operation (credit/debit), same currency, same urgency, same end of day (time period)
- A summary record containing control information:
  - Number of operations
  - Total amount
  - Signature

'Remise retour' (remittance ACH to bank): a payment file containing:

- A header record
- 1-N payment messages to be posted in the same basket, composed of:
  - A header record containing the installation number of the issuer
  - 1-N operations issued by the same sender
  - A summary record (number of operations, total amount)
- A summary record containing control information

- Number of messages
  - Total amount
  - Signatures
- 

## Domestic format example for France

For the SEPA converter examples for France, in which two type trees are provided.

- CFONB320, which contains the type tree definitions for the formats that can be defined by “Remises Informatisées D’Ordres de Paiement International” (CFONB320\_PI), and “Remises Informatisées D’Ordres de Virement National France” (CFONB320\_VF). Only the CFONB\_PI format is used in the example maps.
- MINOS, which contains the type tree definitions for many of the MINOS formats

Example files that describe the domestic format for France are located in the <install\_dir>/packs/financial\_payments\_vn.n.n.n/sepa/examples/france directory.

See the [Example for France](#) documentation for more information.

---

## Domestic format for Italy

The domestic format example for Italy includes maps, trees, and data used to convert SEPA data to SIA Rete Interbacaria (RNI) Italian domestic format, and conversion from the domestic formats to SEPA.

RNI Is a proprietary message format from SIA Interbank Company for Automation (Società Interbancaria per l'Automazione). These messages are used to carry out banking operations such as the following:

- Exchange accounting information
- Determining the balance of each participant
- Procedures for truncated cheques
- Electronic collection orders (RIBAs)
- Direct debits
- Retail credit transfers

These procedures are managed by service providers, or *centri applicativi*, which are bank-owned software companies that carry out a number of activities on the banks' behalf in connection with the exchange of accounting information on interbank payments.

- [Domestic format example for Italy](#)

Example files that describe the domestic format for Italy are located in the <install\_dir>/packs/financial\_payments\_vn.n.n.n/sepa/examples/italy directory.

---

## Domestic format example for Italy

Example files that describe the domestic format for Italy are located in the <install\_dir>/packs/financial\_payments\_vn.n.n.n/sepa/examples/italy directory.

See the [Example for Italy](#) documentation for more information.

---

## SEPA format converters for SWIFTNet FIN

This documentation describes the SEPA format converter for the SWIFTNet FIN format.

Mapping guidelines exist for the SWIFTNet FIN MT 101, MT 102, MT 102+, MT 103, MT 103+message.

- [SWIFTNet FIN format example](#)

Example files that describe the SWIFTNet FIN format are located in the <install\_dir>/packs/financial\_payments\_vn.n.n.n/sepa/examples/swiftfin directory.

---

## SWIFTNet FIN format example

Example files that describe the SWIFTNet FIN format are located in the <install\_dir>/packs/financial\_payments\_vn.n.n.n/sepa/examples/swiftfin directory.

See the [SWIFTNet FIN / SEPA example](#) documentation for more information.

---

## SEPA component examples

Examples are included in the pack installation to assist you in using the SEPA component.

- [Validation using the EPC schemas example](#)

The Validation using the EPC schemas example is used to replace the existing validation framework maps that use the ISO20022 published XML schemas with maps that use the abbreviated Technical Validation Subset (TVS) published by the EPC.

- [Example for France](#)

The CFONB organization defines additional formats used in payments processing for the Customer-Bank processes.

- [\*\*Example for Germany\*\*](#)  
The DTAUS data format was defined within the overall online banking standard that was announced in 1995 by the German banks represented by their associations participating in the Central Banking Committee (ZKA - Zentraler Kreditausschuss).
- [\*\*Example for the UK\*\*](#)  
The Example for the UK demonstrates how to use the BACS / SEPA converter maps.
- [\*\*Example for Italy\*\*](#)  
The domestic format example for Italy includes maps, type trees, and data used to convert SEPA data to SIA Rete Interbacarria (RNI) Italian domestic format, and conversion from the domestic formats to SEPA.
- [\*\*PAIN / PACS example\*\*](#)  
This example shows how to use the PACS to PAIN message and the PAIN to PACS message converter maps.
- [\*\*Passthrough Example for z/OS\*\*](#)  
The Passthrough Example for z/OS, validates a message against a schema in the z/OS environment and contains a sample of the JCL required to run it.
- [\*\*Sample JCL for the validation framework maps example\*\*](#)  
This example demonstrates how to deploy and execute the SEPA validation maps on z/OS batch.
- [\*\*SWIFTNet FIN / SEPA example\*\*](#)  
This example demonstrates how to use the SWIFTNet FIN / SEPA example.

## Validation using the EPC schemas example

The Validation using the EPC schemas example is used to replace the existing validation framework maps that use the ISO20022 published XML schemas with maps that use the abbreviated Technical Validation Subset (TVS) published by the EPC.

- [\*\*Download the EPC schemas\*\*](#)  
Before you can use the EPC Validation example, you must download the EPC schemas from SEPA. Instructions for downloading the schemas are available in the IBM Transformation Extender Pack for Financial Payments release notes.
- [\*\*Example maps and schemas\*\*](#)  
The maps included in the epc\_mapsandschemas.mms source file are identically named as the SEPA validation map source, sepa\_validation in the <install\_dir>/packs/financial\_payments\_vn.n.n.n/sepa/examples/epc\_mapsandschemas/mapsandschemas directory.
- [\*\*How to run the example in Design Studio\*\*](#)  
These steps describe how to run the Validation using the EPC schemas example.
- [\*\*How to run the example in Design Server\*\*](#)  
Run instructions for the EPC SEPA Validation example.

## Download the EPC schemas

Before you can use the EPC Validation example, you must download the EPC schemas from SEPA. Instructions for downloading the schemas are available in the IBM Transformation Extender Pack for Financial Payments release notes.

Download instructions for the Direct Debit and Credit Transfer implementations for this example are provided in a readme file that installs with the pack. Go to the following location and follow the instructions contained in the readme file:

```
<install_dir>/packs/financial_payments_vn.n.n.n/sepa/examples/epc_mapsandschemas/readme_epc_mapsandschemas.txt
```

## Example maps and schemas

The maps included in the epc\_mapsandschemas.mms source file are identically named as the SEPA validation map source, sepa\_validation in the <install\_dir>/packs/financial\_payments\_vn.n.n.n/sepa/examples/epc\_mapsandschemas/mapsandschemas directory.

The example maps replace the sepa\_validation maps, and achieve better performance because of the use of the abbreviated EPC schemas.

The <install\_dir>/packs/financial\_payments\_vn.n.n.n/sepa/examples/epc\_mapsandschemas/mapsandschemas directory includes a batch script called build\_epc\_mapsandschemas.bat. This batch script deploys all of the necessary files to the <install\_dir>/packs/financial\_payments\_vn.n.n.n/sepa/examples/epc\_mapsandschemas/mapsandschemas directory and compiles the epc\_sepa\_validation.mms maps. Instructions for running the batch script are provided below.

## How to run the batch script

1. Open a command prompt window and navigate to the <install\_dir>/packs/financial\_payments\_vn.n.n.n/sepa/examples/epc\_mapsandschemas/mapsandschemas directory.
2. Determine the parameters you want to use. The script uses two mandatory parameters:
  - a. The platform for building the maps. The choices available would be the same as the platforms supported by the corresponding TX version installed
  - b. The type of SEPA map to compile:
    - ct
    - dd
    - both
  - c. The TX install directory path
3. At the prompt, type in: build\_epc\_mapsandschemas.bat followed by the platform choice listed above, the SEPA type choice, and the TX install directory path.

Examples:

- To build both the credit transfer and direct debit maps for Windows operating system, enter the following at the command prompt:

build\_epc\_mapsandschemas.bat WINDOWS both and <TX\_install\_dir>

- To build the credit transfer maps for ZOS operating system, enter the following at the command prompt:  
build\_epc\_mapsandschemas.bat ZOS ct and <TX\_install\_dir>
- To build the direct debit maps for ZLINUX operating system:  
build\_epc\_mapsandschemas.bat ZLINUX dd and <TX\_install\_dir>

---

## How to run the example in Design Studio

These steps describe how to run the Validation using the EPC schemas example.

### About this task

If you use the deploy script, the example maps are deployed to  
<install\_dir>/packs/financial\_payments\_vn.n.n.n/sepa/examples/epc\_mapsandschemas/mapsandschemas from the example directory. The deployed maps replace any existing compiles of the sepa\_validation map source. The maps are run using sepalid.mmc the same way as instructed in the Pack for Financial Payments user documentation.

### Procedure

1. Copy the <install\_dir>/packs/financial\_payments\_vn.n.n.n/sepa/type\_trees directory to the <install\_dir>/packs/financial\_payments\_vn.n.n.n/sepa/examples/epc\_mapsandschemas/mapsandschemas directory.
2. Use the Design Studio to open the following map source file:  
<install\_dir>/packs/financial\_payments\_vn.n.n.n/sepa/examples/epc\_mapsandschemas/mapsandschemas/epc\_sepa\_validation.mms
3. Verify the input and output card settings to ensure the correct files are being used.
4. Build the map in the source file, and then run the following transformation map:
  - sepalid
  - [Notes about the maps](#)

---

## Notes about the maps

Note these points about the maps:

- It is assumed that the schemas have been downloaded from the EPC website, unzipped, renamed, and placed in the correct location. See the release notes for instructions on downloading schemas.
- These maps differ from the main validation maps in that they have much more schema validation than the main validation maps due to the EPC schemas having more restrictions in place. As a result, error messages generated from these maps will be different in many cases, but will produce an error message if the main map has produced an error for the same data.
- The EPC schemas only allow fields that are shaded in yellow in the implementation guides. Using other fields will result in input validation errors, so if other fields are needed, then these maps should not be used.

---

## How to run the example in Design Server

Run instructions for the EPC SEPA Validation example.

### Before you begin

Before you can run the SEPA EPC Validation example, you must add the schemas that you download from SEPA with the schemas that are contained in the **sepaEPC.zip** file. Before you import the **sepaEPC.zip** into the Design Server.

### About this task

These steps describe how to run the EPC SEPA Validation example.

### Procedure

1. Download the schemas from SEPA.
2. Unzip the contents of the **sepaEPC.zip**.
3. Add the schemas that you downloaded from SEPA to  
filedata/schemas/ct  
filedata/schemas/dd  
schemadata/schemas/ct

- schemadata/schemas/dd
4. Compress the following folders into a compressed file called: sepaEPC\_upd.zip.
    - filedata
    - files
    - maps
    - schemadata
    - schemas
  5. Import into the Design Server as described in Importing projects.
  6. Create a package for the project and build all the maps.
  7. Open the sepaEPC\_upd.zip project.
  8. Run the following map:
    - sepalid
  9. View the output. Go to the map Diagram-Result, and hover the cursor over Target, then select View Data.
  10. To download the output file, see [How to download output files](#).
- **[Building multiple maps using Deploy](#)**  
To deploy large numbers of maps (generally, when a project contains more than 10 maps), use the instructions provided here.
  - **[How to download output files](#)**  
Download the output files.
  - **[Notes about the maps](#)**  
Additional points to note about the maps.

---

## Building multiple maps using Deploy

To deploy large numbers of maps (generally, when a project contains more than 10 maps), use the instructions provided here.

### About this task

---

Follow these steps to deploy large numbers of maps:

### Procedure

---

1. From the Design Server UI, click on the Deploy drop down, and click Packages.
2. Define a new Package, then click Add icon.
3. Define a Name field on the Edit Package dialog.
4. From the Project drop down, select the project.
5. Check top level box on Maps and Files drop down, this select all maps and files in the project
6. Save the Package.
7. Return to Design Server UI, main page, click Deploy drop down, scroll down and select Deploy from the list on the left.
8. Select the desired server (see the Server definition on Deployment > Deployment overview) and select the package.
9. Click Deploy.
10. Return to the Design Server UI, main page, click Design, select project (one just deployed).
11. Open the map.

---

## How to download output files

Download the output files.

### About this task

---

To download the output file, go to **Files** in the Design Server UI and download the desired file using these steps:

### Procedure

---

1. Click the output Card Settings you want to download.
2. From the Command text box, copy the name of the file that you want to download. Make certain that you copy only the file name.  
For example, if the Command text box displays:

data/ach\_contested\_dishonored\_return.out

copy only:

ach\_contested\_dishonored\_return.out

3. From Files, click the New icon to open the New File dialog box.
4. Paste the file name in the Name field.
5. Click the Folder field drop down and select data.

6. Click Ok.
  7. Build and re-run the map.
  8. Go to Files and select Download to view the data.
- 

## Notes about the maps

Additional points to note about the maps.

- It is assumed that the schemas have been downloaded from the EPC website, unzipped, renamed and placed in the correct location.
  - These maps differ from the main validation maps in that they have much more schema validation than the main validation maps due to the EPC schemas having more restrictions in place. As a result, error messages generated from these maps will be different in many cases, but will produce an error message if the main map has produced an error for the same data.
  - The EPC schemas only allow fields that are shaded in yellow in the implementation guides. Using other fields will result in input validation errors, so if other fields are needed, then these maps should not be used.
- 

## Example for France

The CFONB organization defines additional formats used in payments processing for the Customer-Bank processes.

This format, used in the examples for France, is described in *Remises informatisees d'ordres de paiement International, au format 320 caracteres*. This document is available through a catalogue on the web site: [www.revue-banque.fr/editionCatalogue.do?shortcut=edition\\_catalogue](http://www.revue-banque.fr/editionCatalogue.do?shortcut=edition_catalogue). This format will be replaced by an XML based format on which the French banking community works currently, and by the SEPA PAIN (payments initiation) format in the case of SEPA payment operations.

It should be noted that the CFONB format is close to the MINOS format, but each bank has the freedom to adapt the format to its specific needs.

- [Directory structure](#)
  - [Example for France components](#)  
Example maps, type trees and example files, install with the pack and are required to run the Example for France.
  - [Example files](#)  
These example files are included with the Example for France.
  - [How to run the example in Design Studio](#)  
Steps to run the Example for France.
  - [How to run the example in Design Server](#)  
Run instructions for the MINOS/CFONB SEPA example (example for France).
- 

## Directory structure

The files needed to run the Example for France are located in subdirectories under:

```
<install_dir>/packs/financial_payments_vn.n.n/n/
```

This table shows the installed location of all of the files required for the example:

Directory:	Description
./sepa/examples/france/data	Sample input test data for MINOS and CFONB conversions.
./sepa/examples/france/maps	Example converter maps for France.
./sepa/examples/france/type_trees	MINOS and CFONB type trees
./sepa/mapsandschemas	Contains all UNIFI schemas used by the maps.
./sepa/type_trees	Contains all UNIFI type trees used by the maps.

## Example for France components

Example maps, type trees and example files, install with the pack and are required to run the Example for France.

The type trees, maps, and example files for the Example for France install in these directories (respectively):

```
<install_dir>/packs/financial_payments_vn.n.n/n/sepa/examples/france/type_trees
```

```
<install_dir>/packs/financial_payments_vn.n.n/n/sepa/examples/france/maps
```

```
<install_dir>/packs/financial_payments_vn.n.n/n/sepa/examples/france/data
```

- [Type trees](#)  
These type trees install with the pack and are required to run the Example for France:
- [Maps](#)  
These maps install with the pack and are required to run the Example for France:

## Type trees

These type trees install with the pack and are required to run the Example for France:

- minos.mtt - Contains a definition for selected MINOS formats as defined in Version 5 of the MINOS Standard and validated against real life sample data.
- CFONB320.mtt - Contains a definition for the CFONB Paiement International (PI) and Virement National France (VN) as defined in Version 3 dated January 2005 and validated against real life sample data.

## Maps

These maps install with the pack and are required to run the Example for France:

- sepa\_fr.mms - Contains these executable maps:
  - sepfr3ct\_pacs\_008\_minos - Converts from \$pacs.008 UNIFI to MINOS Remise Aller Virement (Credit Transfer). The records are all given a default code operation 120.
  - sepfr5dd\_pacs\_003\_minos - Converts from \$pacs.003 UNIFI to MINOS Remise Aller Prelevement (Direct Debit). The records are all given a default code operation of 180 (Prelevement a 4 jours).
  - sepfr1ct\_pain\_001\_cfobn - Converts from pain.001 UNIFI to CFONB Virement National France (Credit Transfer)format. The CFONB records are all given a default code operation of VF.
- fr\_sepa.mms - Contains these executable maps:
  - frsep3ct\_minos\_pacs\_008 - Converts from MINOS Remise Aller Virement (Credit Transfer) to \$pacs.008 UNIFI message.
  - frsep5dd\_minos\_pacs\_003 - Converts from MINOS Remise Aller Prelevement (Direct Debit) to \$pacs.003 UNIFI message.
  - frsep1ct\_cfobn\_pain\_001 - Converts from CFONB Virement National France (Credit Transfer) format to \$pain.001 UNIFI message.

## Example files

These example files are included with the Example for France.

- sepa\_minos\_config.xml - This configuration file is to be used as input card #1 to the following executable maps:

- sepfr3ct\_pacs\_008\_minos.mmc
- sepfr5dd\_pacs\_003\_minos.mmc
- frsep3ct\_minos\_pacs\_008.mmc
- frsep5dd\_minos\_pacs\_003.mmc

This configuration file is used to populate sample valid values such as IBAN and BIC numbers used by SEPA. It can be modified as needed to add more specific values.

- sepa\_cfobn\_config.xml - This configuration file is to be used as input card #1 to the following executable maps:

- frsep1ct\_cfobn\_pain\_001.mmc
- sepfr1ct\_pain\_001\_cfobn.mmc

- bic.xml - File that contains BIC numbers used in the configuration and sample input files.

- FIToFICstmrcdtTrf\_pacs008.xml - Sample data for input card #2 of sepfr3ct\_pacs\_008\_minos.mmc

- FIToFICstmrdrcdtDbt\_pacs003.xml - Sample data for input card #2 of sepfr5dd\_pacs\_003\_minos

- CstmrcdtTrfInitn\_pain001.xml - Sample data for input card #2 of sepfr1ct\_pain\_001\_cfobn.mmc

- fr\_pain001\_cfobn.in - Sample data for input card #2 of frsep1ct\_cfobn\_pain\_001

- fr\_pacs008\_minos.in - Sample data for input card #2 of frsep3ct\_minos\_pacs\_008

- fr\_pacs003\_minos.in - Sample data for input card #2 of frsep5dd\_minos\_pacs\_003

## How to run the example in Design Studio

Steps to run the Example for France.

### About this task

Run the example map as follows:

### Procedure

1. Use the Design Studio to open the following map source files:
  - fr\_sepa.mms
  - sepa\_fr.mms
2. Verify the input and output card settings to ensure the correct files are being used.
3. Build the maps in the source files, and then run the following transformation maps:
  - frsep1ct\_cfobn\_pain\_001
  - frsep3ct\_minos\_pacs\_008
  - frsep5dd\_minos\_pacs\_003
  - sepfr1ct\_pain\_001\_cfobn
  - sepfr3ct\_pacs\_008\_minos
  - sepfr5dd\_pacs\_003\_minos

- **[Customization notes](#)**

Consider these customization points about the maps as you use the Example for France.

## Customization notes

Consider these customization points about the maps as you use the Example for France.

- It is assumed all currencies are EUR.
- In order to determine a BIC from the MINOS format, a translation table lookup is required. This translation table is not supplied with the pack, and must be sourced separately. A default BIC value is used in the configuration file and must be reviewed.
- In order to convert from MINOS Account details to IBAN a translation table lookup is required. This translation table is not supplied with the Pack, and must be sourced separately. Default IBAN values are used in the configuration file and must be reviewed.

## How to run the example in Design Server

Run instructions for the MINOS/CFONB SEPA example (example for France).

### About this task

These steps describe how to run the MINOS/CFONB SEPA example.

### Procedure

1. Import the sepaFR.zip file into the Design Server.
2. Create a package for the project and build all the maps.
3. Open the sepaFR project in the Design Server.
4. Run the following maps:
  - frsep1ct\_cfonb\_pain\_001
  - frsep3ct\_minos\_pacs\_008
  - frsep5dd\_minos\_pacs\_003
  - sepfr1ct\_pain\_001\_cfonb
  - sepfr3ct\_pacs\_008\_minos
  - sepfr5dd\_pacs\_003\_minos
5. View the output. Go to the map Diagram-Result, and hover the cursor over Target, then select View Data.
6. To download the output file, see [How to download output files](#).
  - [How to download output files](#)  
Download the output files.
  - **[Customization notes](#)**  
Consider these customization points about the maps as you use the Example for France.

## How to download output files

Download the output files.

### About this task

To download the output file, go to **Files** in the Design Server UI and download the desired file using these steps:

### Procedure

1. Click on the output Card Settings you want to download.
2. From the Command text box, copy the name of the file that you want to download. Make certain that you copy only the file name.  
For example, if the Command text box displays:

```
data/ach_contested_dishonored_return.out
```

copy only:

```
ach_contested_dishonored_return.out
```
3. From Files, click the New icon to open the New File dialog box.
4. Paste the file name in the Name field.
5. Click the Folder field drop down and select data.
6. Click Ok.
7. Build and run the map.
8. Go to Files and select Download to view the data.

## Customization notes

Consider these customization points about the maps as you use the Example for France.

- It is assumed all currencies are EUR.
- In order to determine a BIC from the MINOS format, a translation table lookup is required. This translation table is not supplied with the pack, and must be sourced separately. A default BIC value is used in the configuration file and must be reviewed.
- In order to convert from MINOS Account details to IBAN a translation table lookup is required. This translation table is not supplied with the Pack, and must be sourced separately. Default IBAN values are used in the configuration file and must be reviewed.

## Example for Germany

The DTAUS data format was defined within the overall online banking standard that was announced in 1995 by the German banks represented by their associations participating in the Central Banking Committee (ZKA - Zentraler Kreditausschuss).

The full standard is the "Homebanking Computer Interface (HBCI)". This format is used for both interbank and customer-bank payments processing in Germany.

In the DTAUS format, a physical file contains:

- a single file header also called Record A
- one or more payment exchange messages called Record C
- a single file trailer called Record E

There is a single format used for both direct debit or credit transfer, identified by Record A (File Header) Field 3 (Identifier). However, a logical file must contain only credits or only direct debits (no mixed payment records). Record A contains the sender and receiver of the file, and is a fixed length of 128 bytes. Each Record C contains details of the orders to be executed (credits or debits) and contains a constant and a variable length section. Record E is used for performing checks.

- [Directory structure](#)
- [Example for Germany components](#)
  - Example maps, type trees and example files, install with the pack and are required to run the Example for Germany.
- [Maps](#)
  - These maps install with the pack and are required to run the Example for Germany:
- [Example files](#)
  - These example files are included with the Example for Germany.
- [How to run the example in Design Studio](#)
  - These steps describe how to run the Example for Germany.
- [How to run the example in Design Server](#)
  - Run instructions for the DATUS SEPA example (example for Germany).

## Directory structure

The files needed to run the Example for Germany are located in sub-directories under:

<install\_dir>/packs/financial\_payments\_vn.n.n.n/

This table shows the installed location of all of the files required for the example:

Directory:	Description
..../sepa/examples/germany/data	sample input test data for DTAUS conversions.
..../sepa/examples/germany/maps	Example converter maps for Germany.
..../sepa/examples/germany/type_trees	Type trees for Germany.
..../sepa/mapsandschemas	Contains all UNIFI schemas used by the maps.
..../sepa/type_trees	Contains all UNIFI type trees used by the maps.

## Example for Germany components

Example maps, type trees and example files, install with the pack and are required to run the Example for Germany.

The type trees, maps, and example files for the Example for Germany install in these directories (respectively):

<install\_dir>/packs/financial\_payments\_vn.n.n.n/sepa/examples/germany/type\_trees

<install\_dir>/packs/financial\_payments\_vn.n.n.n/sepa/examples/germany/maps

<install\_dir>/packs/financial\_payments\_vn.n.n.n/sepa/examples/germany/data

- [Type trees](#)
  - These type trees install with the pack and are required to run the Example for Germany:

## Type trees

These type trees install with the pack and are required to run the Example for Germany:

This subdirectory contains the following .mtt files:

- dtaus1.2.mtt
- DTAUS1.3.mtt
- sepa\_config.mtt
- bic.mtt
- SUM.mtt

The trees dtaus1.2 and dtaus1.3 contain definition for all DTAUS domestic record formats.

## Maps

These maps install with the pack and are required to run the Example for Germany:

- de\_sepa.mms - Contains one executable map:
  - desep1ct\_dtaus\_pain001 - This is a Credit Transfer Map that converts a DTAUS domestic record format to a pain.001 Customer Credit Transfer Initiation message.
- sepa\_de.mms - Contains these executable maps:
  - sepde1ct\_pain001\_dtaus - This map converts a Credit Transfer pain.001 message to the DTAUS domestic record format. The pain.001 message is validated using the pain.001.001.09 schema
  - sepde4dd\_pacs003\_dtaus - This map converts a Direct Debit pacs.003 message to DTAUS record format. The pacs.003 message is validated using the pacs.003.001.08 schema
  - sepde2dd\_pain008\_dtaus - This map converts a direct debit pain.008 message to DTAUS record format. The pain.008 message is validated using the pain.008.001.08 schema.

## Example files

These example files are included with the Example for Germany.

- dtseconf.xml - Input card to map desep1ct\_dtaus\_pain001
- CTDTAUS.txt - Input card to map desep1ct\_dtaus\_pain001
- CstmrCdtTrfInitn\_pain001.xml - Input card to map sepde1ct\_pain001\_dtaus.
- BIC.txt - Input card to map sepde1ct\_pain001\_dtaus and sepde2dd\_pain008\_dtaus.
- SEPACConfig.xml - Input card to map sepde1ct\_pain001\_dtaus and sepde2dd\_pain008\_dtaus.
- FIToFICstmrDrctDbt\_pacs003.xml - Input card to map sepde4dd\_pacs003\_dtaus.
- CstmrDrctDbtInitn\_pain008.xml - Input card to map sepde2dd\_pain008\_dtaus.

## How to run the example in Design Studio

These steps describe how to run the Example for Germany.

### About this task

- The maps included in the de\_sepa.mms, are used to convert from domestic DTAUS format to SEPA format.
- The maps included in the sepa\_de.mms, are used to convert from SEPA to domestic DTAUS format.
- It is assumed that all currencies are EUR.
- In order to convert from Domestic Sort Code to BIC a translation table lookup is required. This translation table is not supplied with the Pack for Financial Payments and must be sourced separately. A default BIC value is used in the maps which must be reviewed.

### Procedure

1. Use the Design Studio to open the following map source files:
  - de\_sepa.mms
  - sepa\_de.mms
2. Verify the input and output card settings to ensure the correct files are being used.
3. Build the maps in the source files, and then run the following transformation maps:
  - desep1ct\_dtaus\_pain001
  - sepde1ct\_pain001\_dtaus
  - sepde2dd\_pain008\_dtaus
  - sepde4dd\_pacs003\_dtaus

Note: Map desep1ct\_dtaus\_pain001 reads input file CTDTAUS.txt in the data file, or it can use the one created by map sepde1ct\_pain001\_dtaus and creates a pain.001 file. This resulting file needs to be a valid SEPA file. Ensure the SEPA validation process includes the BIC values used in the resulting pain.001 file. This map uses a config file to populate sample valid values such as IBAN numbers and SEPA. This config file can be modified as needed to add more user specific values.

## How to run the example in Design Server

Run instructions for the DATUS SEPA example (example for Germany).

### About this task

These steps describe how to run the DATUS SEPA example.

### Procedure

1. Import the sepaDE.zip file into the Design Server.
2. Create a package for the project and build all the maps.
3. Open the sepaDE project in the Design Server.
4. Run the following maps:
  - desep1ct\_dtaus\_pain001
  - sepde1ct\_pain001\_dtaus
  - sepde2dd\_pain008\_dtaus
  - sepde4dd\_pacs003\_dtaus
5. View the output. Go to the map Diagram-Result, and hover the cursor over Target, then select View Data.
6. To download the output file, see [How to download output files](#).
  - [How to download output files](#)Download the output files.

## How to download output files

Download the output files.

### About this task

To download the output file, go to **Files** in the Design Server UI and download the desired file using these steps:

### Procedure

1. Click on the output Card Settings you want to download.
2. From the Command text box, copy the name of the file that you want to download. Make certain that you copy only the file name.  
For example, if the Command text box displays:  
  
data/ach\_contested\_dishonored\_return.out  
  
copy only:  
  
ach\_contested\_dishonored\_return.out
3. From Files, click the New icon to open the New File dialog box.
4. Paste the file name in the Name field.
5. Click the Folder field drop down and select data.
6. Click Ok.
7. Build and re-run the map.
8. Go to Files and select Download to view the data.

## Example for the UK

The Example for the UK demonstrates how to use the BACS / SEPA converter maps.

Note: This format is available for purchase from APACS at [www.apacs.org.uk](http://www.apacs.org.uk), and references Standard 18 - 1 October 2002.pdf BACS Interchange Standards (v18/9 dated 1 October 2002)

The BACS standard contains two formats for data records: BACS input and BACS output. Both of these formats have the same basic field structure of 100 bytes, but the BACS output format is extended by additional fields. The BACS standard also defines Volume and File and User Header records which vary in size. The BACS input and output formats are described in as follows:

- **BACS input** - This format is used by Banks, and their customers, to send payment data to BACS by electronic transfer, or other means. After initial validation, the data is forwarded to the relevant bank(s) using BACS output format. The BACS input format can be either 100 or 106 bytes. The additional 6 bytes are used to specify individual processing dates within BACS. For full details on this process, refer to the BACS User Manual.
- **BACS output** - The BACS output is always 120 bytes. The additional 20 bytes contain fields added after validation by BACS: Error Code, BACS User Number and BACS Reference (unique reference for each payment; used by BACS for query purposes).

The SEPA type tree and converter maps are based upon the output format. If conversion to and from the input format is required, it can be easily achieved since the base 100-byte format is identical, although the extra data provided by the BACS output format is useful in providing additional unique identifiers for each transaction.

There are several types of data records in BACS format. These data records are identified by Transaction Code (TX Code) and vary depending upon whether the sender is a Bank or a Bank's Customer. The SEPA type tree contains a single file description containing all possible record types.

- [\*\*Directory structure\*\*](#)  
The following directory structure is used for the BACS / SEPA example
  - [\*\*Data directory\*\*](#)  
The BACS / SEPA example data directory contains the example files described here.
  - [\*\*Maps\*\*](#)  
The maps directory contains two source maps.
  - [\*\*Type trees\*\*](#)  
The BACS / SEPA example contains one type tree, in the type\_trees directory.
  - [\*\*How to run the example in Design Studio\*\*](#)  
These procedures describe how to run the BACS /SEPA example.
  - [\*\*How to run the example in Design Server\*\*](#)  
Run instructions for the BACS / SEPA example (example for U.K.).
- 

## Directory structure

The following directory structure is used for the BACS / SEPA example

- <install\_dir>/packs/financial\_payments\_vn.n.n.n/sepa/examples/uk/data
  - Sample input test data for BACS conversions.
- <install\_dir>/packs/financial\_payments\_vn.n.n.n/sepa/examples/uk/maps
  - Example UK converter maps.
- <install\_dir>/packs/financial\_payments\_vn.n.n.n/sepa/examples/uk/type\_trees
  - BACS type tree
- <install\_dir>/packs/financial\_payments\_vn.n.n.n/sepa/mapsandschemas
  - Contains all UNIFI schemas used by the maps.
- <install\_dir>/packs/financial\_payments\_vn.n.n.n/sepa/type\_trees
  - Contains all UNIFI type trees used by the maps.

## Data directory

The BACS / SEPA example data directory contains the example files described here.

The files listed here are installed in the following directory location:

<install\_dir>/packs/financial\_payments\_vn.n.n.n/sepa/examples/uk/data:

- bacs\_ct\_test.dat - Sample test for the uksepa1ct\_bacs\_pain\_001 and uksepa2ct\_bacs\_pacs\_008 maps. The file contains both credit transfer and direct debit data, but only the credit transfer records are processed (identified by transaction type in pos'n 17-18 of "Z4", "Z5" or "99").
- bacs\_dd\_test.dat - Sample test for the uksepa1dd\_bacs\_pain\_008 and uksepa2dd\_bacs\_pacs\_003 maps. The file contains both credit transfer and direct debit data, but only the direct debit records are processed (identified by transaction type in pos'n 17-18 of "01", "04" or "17").
- FIToFICstmrDrctDbt\_pacs003.xml- Sample test data for the sepauk2dd\_pacs\_003\_bacs map.
- FIToFICstmrCdtTrf\_pacs008.xml - Sample test data for the sepauk2ct\_pacs\_008\_bacs map.
- CstmrCdtTrfInitn\_pain001.xml - Sample test data for the sepauk1ct\_pain\_001\_bacs map.
- CstmrDrctDbtInitn\_pain008.xml - Sample test data for the sepauk1dd\_pain\_008\_bacs map.

## Maps

The maps directory contains two source maps.

- [\*\*sepa\\_uk.mms\*\*](#)  
The sepa\_uk.mms map source file contains the executable maps:
- [\*\*uk\\_sepa.mms\*\*](#)  
The uk\_sepa.mms map source file contains the executable maps:

## sepa\_uk.mms

The sepa\_uk.mms map source file contains the executable maps:

- sepauk1ct\_pain\_001\_bacs - Converts from pain.001 UNIFI to BACS credit transfer. The BACS records are all given a default transaction type of "99" (Bank Giro Credit).
- sepauk1dd\_pain\_008\_bacs - Converts from pain.008 UNIFI to BACS direct debit. The BACS records are all given a default transaction type of "17" (Direct Debit).

- `sepauk2ct_pacs_008_bacs` - Converts from pacs.008 UNIFI to BACS credit transfer. The BACS records are all given a default transaction type of "99" (Bank Giro Credit).
- `sepauk2dd_pacs_003_bacs` - Converts from pacs.003 UNIFI to BACS direct debit. The BACS records are all given a default transaction type of "17" (Direct Debit).

Note: For all of the maps, the UTL1 trailer is updated with the credit or debit totals.

---

## uk\_sepa.mms

The uk\_sepa.mms map source file contains the executable maps:

- `uksepa1ct_bacs_pain_001` - Converts from BACS Credit Transfer to pain.001 UNIFI. The UNIFI message is created as a MIXD Grouping, meaning that all unique BACS Originator IDs are mapped to the top-level PmtInf structure, and related Destination IDs are mapped to the nested CdtTrfTxInf structure.
- `uksepa1dd_bacs_pain_008` - Converts from BACS Direct Debit to pain.008 UNIFI. The UNIFI message is created as a "MIXD" Grouping, meaning that all unique BACS Originator IDs are mapped to the top-level PmtInf structure, and related Destination IDs are mapped to the nested DrctDbtTxInf structure.
- `uksepa2ct_bacs_pacs_008` - Converts from BACS Credit Transfer to pacs.008 UNIFI.
- `uksepa2dd_bacs_pacs_003` - Converts from BACS Direct Debit to pacs.003 UNIFI.

---

## Type trees

The BACS / SEPA example contains one type tree, in the type\_trees directory.

The bacs.mtt is located in:

`<install_dir>/packs/financial_payments_vn.n.n/n/sepa/examples/uk/type_trees`

Contains a definition for all BACS payment type, as defined in APACS standard 18, and validated against real-life sample data.

Note: The BACS Output record definition is used.

---

## How to run the example in Design Studio

These procedures describe how to run the BACS /SEPA example.

---

### About this task

Follow these instructions to run the example:

---

### Procedure

1. Use the Design Studio to open the following map source files:
  - `uk_sepa.mms`
  - `sepa_uk.mms`
2. Verify the input and output card settings to ensure the correct files are being used.
3. Build the maps in the source files, and then run the following transformation maps:
  - `uksepa1ct_bacs_pain_001`
  - `uksepa1dd_bacs_pain_008`
  - `uksepa2ct_bacs_pacs_008`
  - `uksepa2dd_bacs_pacs_003`
  - `sepauk1ct_pain_001_bacs`
  - `sepauk1dd_pain_008_bacs`
  - `sepauk2ct_pacs_008_bacs`
  - `sepauk2dd_pacs_003_bacs`

---

## How to run the example in Design Server

Run instructions for the BACS / SEPA example (example for U.K.).

---

### About this task

These steps describe how to run the BACS / SEPA example.

---

### Procedure

1. Import the sepaUK.zip file into the Design Server.
2. Create a package for the project and build all the maps.
3. Open the sepaUK project in the Design Server.
4. Run the following maps:
  - `uksepa1ct_bacs_pain_001`

- uksepa1dd\_bacs\_pain\_008
  - uksepa2ct\_bacs\_pacs\_008
  - uksepa2dd\_bacs\_pacs\_003
  - sepauk1ct\_pain\_001\_bacs
  - sepauk1dd\_pain\_008\_bacs
  - sepauk2ct\_pacs\_008\_bacs
  - sepauk2dd\_pacs\_003\_bacs
5. View the output. Go to the map Diagram-Result, and hover the cursor over Target, then select View Data.
6. To download the output file, see [How to download output files](#).
- [How to download output files](#)  
Download the output files.
  - [Customize the maps](#)  
Points that should be noted when customizing the BACS / SEPA maps.

## How to download output files

Download the output files.

### About this task

To download the output file, go to **Files** in the Design Server UI and download the desired file using these steps:

### Procedure

1. Click the output Card Settings you want to download.
2. From the Command text box, copy the name of the file that you want to download. Make certain that you copy only the file name.  
For example, if the Command text box displays:  
  
data/ach\_contested\_dishonored\_return.out  
  
copy only:  
  
ach\_contested\_dishonored\_return.out
3. From Files, click the New icon to open the New File dialog box.
4. Paste the file name in the Name field.
5. Click the Folder field drop down and select data.
6. Click Ok.
7. Build and re-run the map.
8. Go to Files and select Download to view the data.

## Customize the maps

Points that should be noted when customizing the BACS / SEPA maps.

- It is assumed all currencies are EUR.
- In order to convert from Domestic Sort Code to BIC a translation table lookup is required. This translation table is not supplied with the pack, and must be sourced separately. A default BIC value is used in the maps which must be reviewed.
- In order to convert from Domestic Account Code to IBAN the relevant BIC4 is required. This is not available, so all IBANs are created with the following format:  
"GB" + "00" = "XXXX" = Domestic Sort Code + Domestic Account No (where "00" is a dummy checkdigit, and "XXXX" is a placeholder for BIC4).
- It is not always obvious how to determine the field equivalences between BACS and UNIFI. The mapping rules for the following data must be reviewed and/or sourced locally at the time of implementation:
  - BACS Account Type
  - BACS Transaction Type
  - UNIFI TxId
  - UNIFI EndToEndId
  - UNIFI InstrId
  - UNIFI MsgId
  - UNIFI CrDtTm (creation date)
  - UNIFI DbtrAgt (debtor's agent)
  - UNIFI CdtrAgt (creditor's agent)UNIFI CdtrAgt (creditor's agent)
  - UNIFI Grpg (grouping)
  - UNIFI InitgPty (initiating party)
  - UNIFI ReqdExctnDt (execution date)
  - UNIFI MndtId (direct debit mandate id)
  - UNIFI DtOfSgntr (date mandate signed)

## Example for Italy

The domestic format example for Italy includes maps, type trees, and data used to convert SEPA data to SIA Rete Interbacaria (RNI) Italian domestic format, and conversion from the domestic formats to SEPA.

RNI Is a proprietary message format from SIA Interbank Company for Automation (Società Interbancaria per l'Automazione). These messages are used to carry out banking operations such as the following:

- Exchange accounting information
- Determining the balance of each participant
- Procedures for truncated cheques
- Electronic collection orders (RIBAs)
- Direct debits
- Retail credit transfers

These procedures are managed by service providers, or *centri applicativi*, which are bank-owned software companies that carry out a number of activities on the banks' behalf in connection with the exchange of accounting information on interbank payments.

- **[Example for Italy components](#)**

Example maps, type trees and example files, install with the pack and are required to run the Example for Italy.

- **[How to run the example in Design Studio](#)**

This Example for Italy demonstrates how to map a RNI message into a corresponding SEPA message.

- **[How to run the example in Design Server](#)**

Run instructions for the RNI SEPA example (example for Italy).

---

## Example for Italy components

Example maps, type trees and example files, install with the pack and are required to run the Example for Italy.

The type trees, maps, and example files for the Example for Italy install in these directories (respectively):

```
<install_dir>/packs/financial_payments_vn.n.n.n/sepa/examples/italy/type_trees
<install_dir>/packs/financial_payments_vn.n.n.n/sepa/examples/italy/maps
<install_dir>/packs/financial_payments_vn.n.n.n/sepa/examples/italy/data
```

- **[Type trees](#)**

These type trees install with the pack and are required to run the Example for Italy:

- **[Maps](#)**

These maps install with the pack and are required to run the Example for Italy.

- **[Example files](#)**

These example files are included with the Example for Italy.

---

## Type trees

These type trees install with the pack and are required to run the Example for Italy:

This subdirectory contains the following .mtt files:

- bonfici\_ordinari\_mssw.mtt - Type tree that represents the credit transfers RNI messages that make use of message switching transfer protocols.
- bonfici\_ordinari\_ft.mtt - Type tree that represents the credit transfers RNI messages that make use of file transfer protocols.
- incassi\_commerciali\_mssw.mtt - Type tree that represents the direct debit RNI messages that make use of message switching transfer protocols.
- incassi\_commerciali\_ft.mtt - type tree that represents the direct debit RNI messages that make use of file transfer protocols.

Note: The type trees are based on SIA-RI-BON 001 document Rel. 5.6 and SIA-RI-ICI 001 document Rel 5.7.

---

## Maps

These maps install with the pack and are required to run the Example for Italy.

- [it\\_sepa.mms](#) - Map source used to convert an RNI message to SEPA core message format.
- [sepa\\_it.mms](#) - Map source used to convert a SEPA core message format to an RNI message.

---

## Example files

These example files are included with the Example for Italy.

The example files are located in the ..//sepa/examples/italy/data directory, and include input files for both [it\\_sepa.mms](#) and [sepa\\_it.mms](#).

- **[Input files for it\\_sepa.mms](#)**

The following are the input files for [it\\_sepa.mms](#).

- **[Input files for sepa\\_it.mms](#)**

The following are the input files for [sepa\\_it.mms](#).

- [Other configuration files](#)

These configuration files store fields that are required in the RNI message format but are not available in the SEPA core message format.

## Input files for it\_sepa.mms

The following are the input files for it\_sepa.mms.

- m550(03)\_in.rete - An Interbank Credit Transfer instruction message.
- m401(riba).rete - An Interbank Direct Debit instruction message.
- m550(04)\_in.rete - A returned Interbank Credit Transfer instruction message.
- m405\_riba(03)\_in.rete - A returned Interbank Direct Debit instruction message.
- m550(05)\_in.rete - A rejected Interbank Credit Transfer instruction message.
- m405\_riba(01)\_in.rete - A rejected Interbank Direct Debit instruction message.
- sepa\_ct\_config.xml - A configuration file used to store fields that are required in the SEPA credit transfer messages but is not available in the RNI message format.
- sepa\_dd\_config.xml - A configuration file used to store fields that are required in the SEPA direct debit messages but is not available in the RNI message format.

## Input files for sepa\_it.mms

The following are the input files for sepa\_it.mms.

- FIToFICstmrCdtTrf\_pacs008.xml - A SEPA Financial Institution to Financial Institution Customer Credit Transfer request core message.
- FIToFICstmrDrctDbt\_pacs003.xml - A SEPA Financial Institution to Financial Institution Customer Direct Debit request core message.
- PmtRtr\_pacs004ct.xml - A SEPA Payment Return core message on a Credit Transfer request.
- PmtRtr\_pacs004dd.xml - A SEPA Payment Return core message on a Direct Debit request.
- FIToFIPmtStsRpt\_pacs002ct.xml - A SEPA Payment Status Report core message on a Credit Transfer request.
- FIToFIPmtStsRpt\_pacs002dd.xml - A SEPA Payment Status Report core message on a Direct Debit request.

## Other configuration files

These configuration files store fields that are required in the RNI message format but are not available in the SEPA core message format.

- rete\_550\_config.xml
- rete\_401\_riba\_config.xml
- rete\_405\_riba\_config.xml
- bic.xml - Contains test BIC(s) utilized in the example.

## How to run the example in Design Studio

This Example for Italy demonstrates how to map a RNI message into a corresponding SEPA message.

### About this task

The example maps allow the use of a configuration file that represents field values specific to the target format that is not readily available from the source format.

### Procedure

1. Use the Design Studio to open the following map source file:
  - it\_sepa.mms
2. Build the maps in the source file, and then run the following transformation maps:
  - itsep02c\_rete\_pacs\_008 - Converts a RNI 550 message into a SEPA pacs.008
  - itsep04d\_rete\_pacs\_003 - Converts a RNI 401 message into a SEPA pacs.003
  - itsep06c\_rete\_pacs\_004 - Converts a returned RNI 550 message into SEPA pacs.004
  - itsep08d\_rete\_pacs\_004 - Converts a returned RNI 401 message (equivalent to a RNI 405) into SEPA pacs.004
  - itsep10c\_rete\_pacs\_002 - Converts a rejected RNI 550 message into SEPA pacs.002
  - itsep12d\_rete\_pacs\_002 - Converts a rejected RNI 401 message (equivalent to a RNI 405) into SEPA pacs.002
3. The 2nd input card makes use of either sepa\_ct\_config.xml or a sepa\_dd\_config.xml. It contains data which are required in the target SEPA format but is not available in the source RNI format.  
For example:

```
constant name : settlement_method with value "CLRG".
```

You must make a business decision as to how to determine what type of settlement will be applied per RNI message that is being converted to SEPA.

During implementation, this file must be replaced with the actual source (that is,, database repository) or it must be based upon your particular situation.

4. Use the Design Studio to open the following map source file:
  - sepa\_it.mms
5. Build the maps in the source files, and then run the following transformation maps:
  - sept02c\_pacs\_008\_rete - Converts a SEPA pacs.008 to a RNI 550 message

- sepit04d\_pacs\_003\_rete - Converts a SEPA pacs.003 to a RNI 401(riba) message
  - sepit06c\_pacs\_004\_rete - Converts a SEPA pacs.004 to a "returned" RNI 550 message
  - sepit08d\_pacs\_004\_rete - Converts a SEPA pacs.004 to a "returned" RNI 401(riba) equivalent to RNI 405 message.
  - sepit10c\_pacs\_002\_rete - Converts a SEPA pacs.002 to a "rejected" RNI 550 message.
  - sepit12d\_pacs\_002\_rete - Converts a SEPA pacs.002 to a "rejected" RNI 401(riba) equivalent to RNI 405 message.
6. The 2nd input card makes use of either rete\_550\_config.xml, rete\_401\_riba\_config.xml or rete\_405\_riba\_config.xml It contains data which is required in the RNI format but is not available in the SEPA format.

For example:

constant name : mittente with value "12345".

You must determine what is the "mittente" (sender) five digit numeric id to be applied per SEPA transaction that is being converted to a RNI message.

During actual implementation, this file must be replaced with actual source (database repository) or must be based upon your unique situation.

7. Prior to executing the sepa validation map against the SEPA messages from this example, perform the following:

- Backup the bic.xml file that is located in:

<install\_dir>/packs/financial\_payments\_vn.n.n.n/sepa/mapsandschemas

- Copy the bic.xml file from:

<install\_dir>/packs/financial\_payments\_vn.n.n.n/sepa/examples/italy/data

to

<install\_dir>/packs/financial\_payments\_vn.n.n.n/sepa/mapsandschemas

## How to run the example in Design Server

Run instructions for the RNI SEPA example (example for Italy).

### About this task

These steps describe how to run the RNI SEPA example.

### Procedure

1. Import the sepaIT.zip file into the Design Server.
2. Create a package for the project and build all the maps.
3. Open the sepaIT project in the Design Server.
4. Run the following maps:
  - itsep02c\_rete\_pacs\_008
  - itsep04d\_rete\_pacs\_003
  - itsep06c\_rete\_pacs\_004
  - itsep08d\_rete\_pacs\_004
  - itsep10c\_rete\_pacs\_002
  - itsep12d\_rete\_pacs\_002
  - sepit02c\_pacs\_008\_rete
  - sepit04d\_pacs\_003\_rete
  - sepit06c\_pacs\_004\_rete
  - sepit08d\_pacs\_004\_rete
  - sepit10c\_pacs\_002\_rete
  - sepit12d\_pacs\_002\_rete
5. View the output. Go to the map Diagram-Result, and hover the cursor over Target, then select View Data.
6. To download the output file, see [How to download output files](#).

- [Building multiple maps using Deploy](#)

To deploy large numbers of maps (generally, when a project contains more than 10 maps), use the instructions provided here.

- [How to download output files](#)

Download the output files.

## Building multiple maps using Deploy

To deploy large numbers of maps (generally, when a project contains more than 10 maps), use the instructions provided here.

### About this task

Follow these steps to deploy large numbers of maps:

### Procedure

1. From the Design Server UI, click on the Deploy drop down, and click Packages.
2. Define a new Package, then click Add icon.
3. Define a Name field on the Edit Package dialog.
4. From the Project drop down, select the imported project.

5. Check top level box on Maps and Files drop down, this select all maps and files in the project
  6. Save the Package.
  7. Return to Design Server UI, main page, click Deploy drop down, scroll down and select Deploy from the list on the left.
  8. Select the desired server (see the Server definition on Deployment > Deployment overview) and select the package.
  9. Click Deploy.
  10. Return to the Design Server UI, main page, click Design, select imported project (one just deployed).
  11. Open the map.
- 

## How to download output files

Download the output files.

### About this task

To download the output file, go to **Files** in the Design Server UI and download the desired file using these steps:

### Procedure

1. Click on the output Card Settings you want to download.
  2. From the Command text box, copy the name of the file that you want to download. Make certain that you copy only the file name.  
For example, if the Command text box displays:
- ```
data/ach_contested_dishonored_return.out
```
- copy only:
- ```
ach_contested_dishonored_return.out
```
3. From Files, click the New icon to open the New File dialog box.
  4. Paste the file name in the Name field.
  5. Click the Folder field drop down and select data.
  6. Click Ok.
  7. Build and re-run the map.
  8. Go to Files and select Download to view the data.
- 

## PAIN / PACS example

This example shows how to use the PACS to PAIN message and the PAIN to PACS message converter maps.

- [Conversions demonstrated by the example](#)
  - [Conversions demonstrated by the example](#)  
The PAIN / PACS example shows how to convert these messages:
    - [How to run the example in Design Studio](#)
    - [How to run the example in Design Server](#)
- Run instructions for the PAIN / PACS example.
- [Customize the maps](#)  
Note these points about the PAIN /PACS example maps:

## Conversions demonstrated by the example

This example show how to convert:

- pain.001.001.09 CustomerCreditTransferInitiation to a PACS.008.001.08 FIToFICustomerCreditTransfer (CORE SEPA message type)
- pain.008.001.08 CustomerDirectDebitInitiation to a PACS.003.001.08 FIToFICustomerDirectDebit (CORE SEPA message type)
- pacs.003.001.08 FIToFICustomerDirectDebit (CORE SEPA message type) to a pain.002.001.10 PaymentStatusReport
- PACS.008.001.08 FIToFICustomerCreditTransfer (CORE SEPA message type) to a pain.002.001.10 PaymentStatusReport

The conversions are based on:

- UNIFI (ISO 20022) Message Definition Reports for Payments Standards - Initiation and Payments Standards - Clearing and Settlement
- European Payments Council RuleBook and Implementation Guidelines for Credit Transfer and Direct Debit Core

The following directory structure is used:

```
<install_dir>/packs/financial_payments_vn.n.n.n/sepa/examples/pain_pacs/data - sample input test data for conversions
<install_dir>/packs/financial_payments_vn.n.n.n/sepa/examples/pain_pacs/maps - example converter maps
<install_dir>/packs/financial_payments_vn.n.n.n/sepa/mapsandschemas - contains all UNIFI schemas used by the maps
<install_dir>/packs/financial_payments_vn.n.n.n/sepa/type_trees - contains all UNIFI type trees used by the maps
where <install_dir> is the installation location of the Pack for Financial Payments.
```

- [Example maps and example files](#)

The maps and example files used in the PAIN/PACS example are contained in the maps and data sub-directories, respectively.

---

## Example maps and example files

The maps and example files used in the PAIN/PACS example are contained in the maps and data sub-directories, respectively.

### Maps

The following PAIN/PACS maps are contained in the `<install_dir>/packs/financial_payments_vn.n.n/n/sepa/examples/pain_pacs/maps` directory:

- `pacs_pain.mms` - contains two executable maps :
  - `pacs_003_pain_002` - Converts from `pacs.003.001.08 FIToFICustomerDirectDebit` to `pain.002.001.10 PaymentStatusReport`.
  - `pacs_008_pain_002` - Converts from `pacs.008.001.08 FIToFICustomerCreditTransfer` to `pain.002.001.10 PaymentStatusReport`.
- `pain_pacs.mms` - contains two executable maps :
  - `pain_001_pacs_008` Converts from `pain.001.001.09 CustomerCreditTransferInitiation` to a `pacs.008.001.08 FIToFICustomerCreditTransfer`.
  - `pain_008_pacs_003` Converts from `pain.008.001.08 CustomerDirectDebitInitiation` to a `pacs.003.001.08 FIToFICustomerDirectDebit`

### Example files

The following PAIN/PACS example files are contained in the `<install_dir>/packs/financial_payments_vn.n.n/n/sepa/examples/pain_pacs/data` directory :

- `FIToFICstmrDrctDbt_pacs003.xml` : Sample test for the `pacs_003_pain_002` map.
- `FIToFICstmrCdtTrf_pacs008.xml` : Sample test for the `pacs_008_pain_002` map.
- `CstmrCdtTrfInitn_pain001.xml` : Sample test for the `pain_001_pacs_008` map.
- `CstmrDrctDbtInitn_pain008.xml` : Sample test for the `pain_008_pacs_003` map.

---

## Conversions demonstrated by the example

The PAIN / PACS example shows how to convert these messages:

- `pain.001.001.09 CustomerCreditTransferInitiation` to a `pacs.008.001.08 FIToFICustomerCreditTransfer` (CORE SEPA message type)
- `pain.008.001.08 CustomerDirectDebitInitiation` to a `pacs.003.001.08 FIToFICustomerDirectDebit` (CORE SEPA message type)
- `pacs.003.001.08 FIToFICustomerDirectDebit` (CORE SEPA message type) to a `pain.002.001.10 PaymentStatusReport`
- `pacs.008.001.08 FIToFICustomerCreditTransfer` (CORE SEPA message type) to a `pain.002.001.10 PaymentStatusReport`

The conversions are based on:

- UNIFI (ISO 20022) Message Definition Reports for Payments Standards - Initiation and Payments Standards - Clearing and Settlement
- European Payments Council RuleBook and Implementation Guidelines for Credit Transfer v4.0 and Direct Debit Core v4.0

---

## How to run the example in Design Studio

### About this task

Follow these steps to run the PAIN/PACS example:

### Procedure

1. Use the Design Studio to open the following map source files:
  - `pain_pacs.mms`
  - `pacs_pain.mms`
2. Verify the input and output card settings to ensure that the correct files are being used.
3. Build the maps in the source files, and then run the following transformation maps:
  - `pacs_008_pain_002`
  - `pacs_003_pain_002`
  - `pain_001_pacs_008`
  - `pain_008_pacs_003`

---

## How to run the example in Design Server

Run instructions for the PAIN / PACS example.

### About this task

These steps describe how to run the PAIN/PACS example.

## Procedure

---

1. Import the sepaPainPacs.zip file into the Design Server.
2. Create a package for the project and build all the maps.
3. Open the sepaPainPacs project in the Design Server.
4. Run the following maps:
  - pacs\_008\_pain\_002
  - pacs\_003\_pain\_002
  - pain\_001\_pacs\_008
  - pain\_008\_pacs\_003
5. View the output. Go to the map Diagram-Result, and hover the cursor over Target, then select View Data.
6. To download the output file, see [How to download output files](#).
  - [How to download output files](#)  
Download the output files.

---

## How to download output files

Download the output files.

### About this task

---

To download the output file, go to **Files** in the Design Server UI and download the desired file using these steps:

## Procedure

---

1. Click on the output Card Settings you want to download.
2. From the Command text box, copy the name of the file that you want to download. Make certain that you copy only the file name.  
For example, if the Command text box displays:  
  
data/ach\_contested\_dishonored\_return.out  
  
copy only:  
  
ach\_contested\_dishonored\_return.out
3. From Files, click the New icon to open the New File dialog box.
4. Paste the file name in the Name field.
5. Click the Folder field drop down and select data.
6. Click Ok.
7. Build and re-run the map.
8. Go to Files and select Download to view the data.

---

## Customize the maps

Note these points about the PAIN /PACS example maps:

- It is assumed all currencies are EUR.
- Any rules with the comment // NEEDS CODING will have to be modified. The fields that need more coding deal with specific account lookup information, uniqueness over time rules, and settlement dates that cannot be determined from the original data.

---

## Passthrough Example for z/OS

The Passthrough Example for z/OS, validates a message against a schema in the z/OS environment and contains a sample of the JCL required to run it.

- [Using the example files](#)

---

## Using the example files

- This map is a sample validation against a schema, in order to validate a SEPA message to be structurally correct. It is not to be confused with the full validation framework that the SEPA component offers.
- The purpose of the example map sepa\_passthrough.mms is to do a schema validation for a pacs.008 message on z/OS. The map will take a pacs.008 data file message and validate it against the pacs.008.001.08 schema. The schema used in this example is pacs.008.001.08.xsd which resides in the following installation path:

<install\_dir>/packs/financial\_payments\_vn.n.n.n/sepa/mapsandschemas

Where <install\_dir> is the installation location of the Pack for Financial Payments.

- For validation against a schema on z/OS, the type tree that represents the schema needs to be modified in the Location for the Doc Object, so that it points to the DDNAME where the schema will be referenced. The format for this modification is //DD:PC008XSD. The type tree, pacs.008.001.08\_mvs.mtt, contains that change.
  - The map needs to be built for the z/OS platform, generating an .mvs file, which then needs to be ported in Binary mode to the z/OS platform. The schema, pacs.008.001.08.xsd, and data file, FIToFICstmrCdtTrf\_pacs008.xml also need to be ported in Binary mode to z/OS, following the convention of a unique eight character name. The map and the schema must be in the same location in the z/OS environment.
- The JCL provided in the maps subdirectory, which can also be ported to z/OS in TEXT (ASCII) mode, shows an example for the name of the data sets and file names used.
- The execution of this map produces no output, as its objective is mainly for validation purposes. In the event the validation fails, modify the JCL by adding a trace parameter, such as: PARM='MAPDDS /P256:12 -IF1 INPAC008 -TI'

## Sample JCL for the validation framework maps example

This example demonstrates how to deploy and execute the SEPA validation maps on z/OS batch.

- [Example files](#)
  - [How to run the example](#)
- Use these files to run the Sample JCL for the Validation Framework Maps.

## Example files

These files contained in the <install\_dir>/packs/financial\_payments\_vn.n.n.n/sepa/examples/sample\_jcl directory are used to run the example:

- sepavalidation.jcl - One approach to run the validation framework for SEPA on z/OS.
- put\_mvs\_sample.scp - Example FTP script to transfer compiled maps, inputs, and schemas for the validation framework.
- params.parm - Example parameters file used from the sepavalidation.jcl. The file includes run time parameters for the map execution.

## How to run the example

Use these files to run the Sample JCL for the Validation Framework Maps.

## About this task

The purpose of this example is to show one approach to set up and execute the validation maps provided in the Pack for Financial Payments SEPA component. The included JCL contains comments on the various steps performed. It is assumed that the base product for z/OS batch is already installed and functional on your z/OS machine.

Because of limitations in porting files to z/OS through the Integration Flow Designer, this example includes an FTP script file that can be used to transfer all of the files needed to run the validation framework maps included in the pack.

Perform the following steps to deploy and execute the SEPA validation maps on z/OS batch:

## Procedure

- Create a copy of the <install\_dir>/packs/financial\_payments\_vn.n.n.n/sepa directory to hold the z/OS modified files.
- Open the following type trees in the new type\_trees directory and update the schema location to point to the DDNAME for the schemas as they will be defined on your z/OS system:
  - bic.mtt
  - curcode.mtt
  - camt.027.001.07.mtt
  - camt.029.001.09.mtt
  - camt.087.001.06.mtt
  - camt.056.001.08.mtt
  - pacs.002.001.10.mtt
  - pacs.003.001.08.mtt
  - pacs.004.001.09.mtt
  - pacs.007.001.09.mtt
  - pacs.008.001.08.mtt
  - pacs.028.001.03.mtt
  - pain.001.001.09.mtt
  - pain.002.001.10.mtt
  - pain.007.001.09.mtt
  - pain.008.001.08.mtt
  - pain.009.001.06.mtt
  - pain.010.001.06.mtt
  - pain.011.001.06.mtt
  - pain.012.001.06.mtt
  - sepa\_config.mtt
  - tracelog.mtt
  - xliff.mtt

3. Build all of the above maps for the z/OS platform.
  4. If multiple users will be executing the jobs, or if a non-owner of the data sets will be executing the jobs, edit the file `sepamsgs.xml` to use the ddname of this file.  
As an example: `<!DOCTYPE xliff SYSTEM "xliff.dtd">`  
changes to: `<!DOCTYPE xliff SYSTEM "//DD:XLIFF">`  
where XLIFF is replaced with the dataset name being used to represent the `sepamsgs.xml` file.
  5. Using the `put_mvs_sample.scp` file, make the appropriate modifications to refer to the specific hostname you are transferring the files to, then execute this script.
  6. Make any modifications to the JCL for anything specific to your execution environment. Make any modifications to the map run parameters you may want to change in the `params.parm` file.
  7. Execute the map with the JCL.
- 

## SWIFTNet FIN / SEPA example

This example demonstrates how to use the SWIFTNet FIN / SEPA example.

- [Example files](#)
  - [How to run the example in Design Studio](#)  
These examples demonstrate how to map a SWIFTNet FIN message to a SEPA message, as well as how to map a SEPA message to a SWIFTNet FIN message.
  - [How to run the example in Design Server](#)  
Run instructions for the SWIFTNet FIN / SEPA example.
- 

## Example files

The SWIFT FIN Example files install in the following directories:

```
<install_dir>/packs/financial_payments_vn.n.n.n/sepa/examples/swiftfin/data
<install_dir>/packs/financial_payments_vn.n.n.n/sepa/examples/swiftfin/maps
<install_dir>/packs/financial_payments_vn.n.n.n/sepa/examples/swiftfin/type_trees
```

where `install_dir` represents the installed location of the Pack for Financial Payments.

- [Data directory](#)  
This directory contains sample data for the FIN to SEPA ISO 20022 and SEPA ISO 20022 to FIN conversion examples.
  - [Maps](#)  
These maps are contained in the maps subdirectory.
  - [Type trees](#)  
The SWIFTNet FIN example contains two type trees located in the `type_trees` directory:
- 

## Data directory

This directory contains sample data for the FIN to SEPA ISO 20022 and SEPA ISO 20022 to FIN conversion examples.

The message types are described below.

- FIN to SEPA ISO 20022 (used in `fn_sepamms`)
  - MT101 - Request for Transfer
  - MT102 - Multiple Customer Credit Transfer
  - MT102 - Reject of Multiple Customer Credit Transfer
  - MT102 - Return of Multiple Customer Credit Transfer
  - MT102 - Plus STP variant of MT102
  - MT103 - Single Customer Credit Transfer
  - MT103 - Reject of Single Customer Credit Transfer
  - MT103 - Return of Single Customer Credit Transfer
  - MT103 - Plus STP variant of MT103

The conversion examples assume that MT 102 and MT 103 Reject and Return messages are defined as per the SWIFT User Handbook, Usage Guidelines, see the Payments Reject/Return Guidelines chapter. The conversion examples expect the first three mandatory lines of Tag 72 to be defined as per these guidelines, for example:

`:72:/REJT/2!n or :72:/RETN/2!n (where 2!n = tag of field in error)`

`:72:/2!c2!n/[29x] (error code plus optional text description)`

`:72:/MREF/16x (field 20 of original message)`

- SEPA ISO 20022 to FIN (used in `sepa_fn.mms`)
  - pacs.002 - PaymentStatusReport
  - pacs.004 - PaymentReturn
  - pacs.008 - FItoFICustomerCreditTransfer
  - pain.001 - CustomerCreditTransferInitiation
- Other files
  - `sepa_config.xml` - Contains configuration parameters for the examples.
  - `bic.xml` - Contains test BIC(s) used in the example.

- swift\_config.xml - Contains configuration parameters for the examples.

## Maps

These maps are contained in the maps subdirectory.

- fn\_sepamms - This map source contains these SWIFT FIN converter examples:

Map name	Input file
mt101_pain_001	mt101_seqa.inp and mt101_seqb.inp
mt102_pacs_002	mt102_rjct.inp
mt102_pacs_004	mt102_retn.inp
mt102_pacs_008	mt102.inp
mt102plus_pacs_008	mt102plus.inp
mt103_pacs_002	mt103_reject.inp
mt103_pacs_004	mt103_retrn.inp
mt103_pacs_008	mt103.inp
mt103plus_pacs_008	mt103plus.inp

There are two input files for mt101\_pain\_001:

- mt101\_seqa.inp - Caters for the situation where the Ordering Customer (50a) is present only in Sequence A, and produces a pain\_001 with a Grouping of MIXD.
- mt101\_seqb.inp - has 50a present in every Sequence B, and produces a pain\_001 with a Grouping of SNGL.  
Refer to the UNIFI documentation for pain.001 for more information about Grouping.
- [\*\*sepa\\_fn.mms\*\*](#)  
The sepa\_fn.mms map source is used to convert SEPA messages to a SWIFT FIN format

## sepa\_fn.mms

The sepa\_fn.mms map source is used to convert SEPA messages to a SWIFT FIN format

The SEPA messages are detailed in this table:

Map name	Input file
pacs_002_mt102	pacs_002_mt102.xml
pacs_002_mt103	pacs_002_mt103.xml
pacs_004_mt102	pacs_004_mt102.xml
pacs_004_mt103	pacs_004_mt103.xml
pacs_008_mt102	pacs_008_mt102.xml
pacs_008_mt102plus	pacs_008_mt102plus.xml
pacs_008_mt103	pacs_008_mt103.xml
pacs_008_mt103plus	pacs_008_mt103plus.xml
pain_001_mt101	pain_001_mt101.xml

## Type trees

The SWIFTNet FIN example contains two type trees located in the type\_trees directory:

- swift\_iso7775.mtt - This type tree represents the SWIFT FIN Payments messages (MT1nn). The type tree supports SWIFT 2007 FIN Standards.
- swiftroute.mtt - This is a utility type tree used to pass parameters to and from the RUN maps

## How to run the example in Design Studio

These examples demonstrate how to map a SWIFTNet FIN message to a SEPA message, as well as how to map a SEPA message to a SWIFTNet FIN message.

### About this task

The example maps allow the use of a configuration file that represents field values specific to the target format that is not readily available from the source format.

Follow these steps to run the example:

### Procedure

1. Use the Design Studio to open the following map source files:
  - fn\_sepamms
  - sepa\_fn.mms

2. Build the maps in the source files, and then run the following transformation maps.

- mt101\_pain\_001
- mt102\_pacs\_002
- mt102\_pacs\_004
- mt102\_pacs\_008
- mt102plus\_pacs\_008
- mt103\_pacs\_002
- mt103\_pacs\_004
- mt103\_pacs\_008
- mt103plus\_pacs\_008
- pacs\_002\_mt102
- pacs\_002\_mt103
- pacs\_004\_mt102
- pacs\_004\_mt103
- pacs\_008\_mt102
- pacs\_008\_mt102plus
- pacs\_008\_mt103
- pacs\_008\_mt103plus
- pain\_001\_mt101

3. Review the contents of sepa\_config.xml. This file contains data which are required in the SEPA format but is not available in the SWIFT format.

For example:

constant name : inout\_indicator with value **O**.

A business decision must be made as to how to determine what type of direction will be applied per SEPA transaction that is being converted to a SWIFT message.

During actual implementation, this file must be replaced with an actual source (that is, the database repository), or it must be based upon your unique situation.

A value of **O** populates an outbound specific application header block and an **I** will populate all inbound specific application header block.

- [Conversion notes](#)

---

## Conversion notes

The conversions also take into account the EPC Implementation Guidelines for SEPA. In many cases, the quality of the input data determines whether a successful conversion is possible.

In particular, the following FIN to SEPA and SEPA to FIN points should be noted:

- FIN to SEPA - MT data considerations
  - All accounts should be specified as an IBAN.
  - Use BIC/BEI when possible, as conversion of other forms (for example, name and address) is less successful.
  - When converting Name Address to ISO 20022 a country is usually mandatory, which is not available in FIN. In these cases a default of US has been provided.
  - When mapping charges to ISO 20022 a mandatory Charges Party is required, which is not available in FIN. In these cases a default BIC of "XXXXXXX" has been provided.
- SEPA to FIN - ISO 20022 data considerations  
Parties in ISO 20022 have diverse formats, many of which have no equivalent in FIN, and so conversion is not always successful. In order to maximize the chances of success ensure parties are defined as:

BIC/BEI

or

Postal Address (with at least 1 AdrLine)

---

## How to run the example in Design Server

Run instructions for the SWIFTNet FIN / SEPA example.

---

### About this task

These steps describe how to run the EPC SEPA Validation example.

---

### Procedure

1. Import the sepaSwiftFin.zip file into the Design Server UI. See the release notes for import instructions.

2. Create a package for the project and build all the maps.

3. Open the sepaSwiftFin project in the Design Server UI.

4. Run the following maps:

- mt101\_pain\_001
- mt102\_pacs\_002
- mt102\_pacs\_004
- mt102\_pacs\_008
- mt102plus\_pacs\_008
- mt103\_pacs\_002

- mt103\_pacs\_004
- mt103\_pacs\_008
- mt103plus\_pacs\_008
- pacs\_002\_mt102
- pacs\_002\_mt103
- pacs\_004\_mt102
- pacs\_004\_mt103
- pacs\_008\_mt102
- pacs\_008\_mt102plus
- pacs\_008\_mt103
- pacs\_008\_mt103plus
- pain\_001\_mt101

5. View the output. Go to the map Diagram-Result, and hover the cursor over Target, then select View Data.

6. To download the output file, see [How to download output files](#).

- [Building multiple maps using Deploy](#)

To deploy large numbers of maps (generally, when a project contains more than 10 maps), use the instructions provided here.

- [How to download output files](#)

Download the output files.

## Building multiple maps using Deploy

To deploy large numbers of maps (generally, when a project contains more than 10 maps), use the instructions provided here.

### About this task

Follow these steps to deploy large numbers of maps:

### Procedure

1. From the Design Server UI, click on the Deploy drop down, and click Packages.
2. Define a new Package, then click Add icon.
3. Define a Name field on the Edit Package dialog.
4. From the Project drop down, select the imported project.
5. Check top level box on Maps and Files drop down, this select all maps and files in the project
6. Save the Package.
7. Return to Design Server UI, main page, click Deploy drop down, scroll down and select Deploy from the list on the left.
8. Select the desired server (see the Server definition on Deployment > Deployment overview) and select the package.
9. Click Deploy.
10. Return to the Design Server UI, main page, click Design, select imported project (one just deployed).
11. Open the map.

## How to download output files

Download the output files.

### About this task

To download the output file, go to **Files** in the Design Server UI and download the desired file using these steps:

### Procedure

1. Click on the output Card Settings you want to download.
2. From the Command text box, copy the name of the file that you want to download. Make certain that you copy only the file name.  
For example, if the Command text box displays:  
  
data/ach\_contested\_dishonored\_return.out  
  
copy only:  
  
ach\_contested\_dishonored\_return.out
3. From Files, click the New icon to open the New File dialog box.
4. Paste the file name in the Name field.
5. Click the Folder field drop down and select data.
6. Click Ok.
7. Build and re-run the map.
8. Go to Files and select Download to view the data.

## Overview of the SWIFT component

The SWIFT component supports the SWIFT MT and SWIFT MX message standards. This allows for easy integration of SWIFT messages into your existing systems.

The SWIFT components can be deployed in many different ways, including through the Command Server and Launcher editions. They may also be used with IBM Transformation Extender Integration Server to utilize IBM Integration Bus, IBM Sterling B2B Integrator, or IBM Business Process Manager Advanced.

SWIFT components can also be embedded in your application through use of the APIs provided by the Transformation Extender for Application Programming edition.

Major benefits of the SWIFT component include:

- Compliance with both MT and MX SWIFT standards
  - Regular updates at the pack level that are aligned with scheduled standards changes
  - Type trees for easy integration of the SWIFT standards with your applications
  - Support for MT and MX translation
  - Comprehensive validation and error reporting
- [How to set up the Transformation Extender project](#)  
The Pack for Financial Payments directory structure must be configured within an IBM Transformation Extender project.
- [SWIFT common components](#)  
The Pack for Financial Payments includes components that are used in MT and MX transformations and validations.
- [SWIFT MT components](#)  
These major components support the SWIFT MT standard:
- [SWIFT MX components](#)  
The following are major components of the SWIFT MX standard.
- [Folder structure](#)  
The SWIFT component folders in the Transformation Extender Pack for Financial Payments are located under  
`<install_dir>/packs/financial_payments_vn.n.n.n/swift/`.

---

## How to set up the Transformation Extender project

The Pack for Financial Payments directory structure must be configured within an IBM Transformation Extender project.

### About this task

---

These steps describe how to define the pack directory structure within an Transformation Extender project.

### Procedure

---

1. Open Design Studio.
2. Select **File > New > Extender Project**.
3. Decide on a unique Project name, for example `financial_payments_vn.n.n.n/swift`, where `n.n.n.n` is the version of the Pack for Financial Payments.
4. Clear the Use default location and click on Browse.
5. Locate and choose the default installation directory for the packs. For example, on Windows: `C:/<install_dir>/packs/financial_payments_vn.n.n.n/swift` where `<install_dir>` indicates the installation location of the Pack for Financial Payments, and `n.n.n.n` indicates the version of the Pack for Financial Payments.
6. Click Ok.
7. Click Finish.

---

## SWIFT common components

The Pack for Financial Payments includes components that are used in MT and MX transformations and validations.

These common components consist of sample data, maps, type trees, and schemas. These components are located in the common directory at  
`<install_dir>/packs/financial_payments_vn.n.n.n/swift/common`.

Throughout this documentation, `<install_dir>` is the installation location of the Pack for Financial Payments, and `n.n.n.n` is the current version of the Pack for Financial Payments.

For more information, see [Common SWIFT components](#).

---

## SWIFT MT components

These major components support the SWIFT MT standard:

- SWIFT MT type tree.
- Java™ Validation Component (JVC) to ensure strict compliance with the SWIFT MT standard.
- Logical Message Format (LMF) type tree and maps.
- Examples

The SWIFT MT components are installed in the following location:

`<install_dir>/packs/financial_payments_vn.n.n.n/swift/mt`

Throughout this documentation, `<install_dir>` is the installation location of the Pack for Financial Payments, and `n.n.n.n` is the current version of the Pack for Financial Payments.

For further information, see [SWIFT MT Components](#).

## SWIFT MX components

The following are major components of the SWIFT MX standard.

- A full set of SWIFT MX schemas and type tree.
- A framework for validating SWIFT MX messages.
- Examples that include SWIFT MT and MX translation maps

The SWIFT MX components are installed in the following location: <install\_dir>/packs/financial\_payments\_vn.n.n.n/swift/mx

In the above location, <install\_dir> is the installation location of the Pack for Financial Payments, and n.n.n.n is the current version of the Pack for Financial Payments.

## Folder structure

The SWIFT component folders in the Transformation Extender Pack for Financial Payments are located under <install\_dir>/packs/financial\_payments\_vn.n.n.n/swift/.

Folder	Description
common/data	Sample BIC and currency code and country code lookup files, and a sample jar file to upload schemas to IBM Sterling B2B Integrator.
common/maps	Maps to create lookup files from the SWIFT bank and currency files.
common/schemas	Schemas used to describe the BIC and currency lookup files.
common/type_trees	Type trees for the BIC and currency lookup files
mt/examples	Examples for statement splitting, batch validation and JVC validation using the Launcher. Also includes IBM Sterling B2B Integrator Generic Deenvelope and Generic Envelope examples.
mt/jvc/maps	Map to invoke the JVC. Placeholder for the JVC configuration file.
mt/jvc/metadata	Files used by the JVC for syntax validation.
mt/lmf/from_lmf	Maps for converting LMF format to MT format.
mt/lmf/mapping_tables	Documentation for the mappings from SWIFT to LMF in spreadsheet format.
mt/lmf/to_lmf	Maps for converting MT into LMF.
mt/type_trees	Core SWIFT type trees, JVC and LMF type trees, and general purpose MT type trees.
mx/data	Sample data.
mx/examples	<ul style="list-style-type: none"><li>• Bundling and unbundling MX with the Business Application Header.</li><li>• MT and MX Translation maps.</li><li>• IBM Sterling B2B Integrator Generic Deenvelope service and Generic Envelope service maps.</li></ul>
mx/maps	MX validation framework maps.
mx/msd	Deployment script for the MT and MX translation maps and MX framework validation.
mx/schemas	Latest versions of the SWIFT MX schemas as of the time of this pack release.
mx/type_trees	Full set of SWIFT MX type trees, and type trees used by the MT and MX translation maps.

In addition, two jar files for the JVC are placed in the IBM Transformation Extender installation directory:

- jvcwrap.jar - This file is the version independent wrapper class for simple invocation of the JVC.
- jvalccyy.jar - This file is the core JVC jar file (where ccyy indicates the release version of the pack, for example 432).
- [Additional Configuration for MT JVC validation](#)

## Additional Configuration for MT JVC validation

The swiftMTjvcConfig.tar.gz file is contained within UIProjectImports directory.

The UIProjectImports directory is located in the following location:

<install\_dir>/packs/financial\_payments\_vn.n.n.n/UIProjectImports

Where, <install\_dir> is the installation location of the Pack for Financial Payments, and n.n.n.n is the current version of the Pack for Financial Payments.

Extract the following from swiftMTjvcConfig.tar.gz file.

- jvcwrap.jar
- jvalccyy.jar

Copy jars to <brand\_install\_dir>/extjar

## Common SWIFT components

There are artifacts in the SWIFT component of the Pack for Financial Payments that are used in both the MT and MX processing maps. These artifacts are included in the common directory, which is located at the same level as the mt and mx directories.

The common directory contains the following components:

- Common data components
- Common map components
- Common schema components
- Common type tree components

These components are described in detail in this documentation.

- **[Common data components](#)**

Common data components consist of the bic.xml file; currencycodedecimals.xml file; and the swiftnnnnxsd4si.jar file.

- **[Common map components](#)**

The bictoxml and currencytoxml executable maps are under common/maps/bankfiletoxml.

- **[Common schema components](#)**

The bic.xsd and the ccy.xsd schemas are common schema components.

- **[Common type tree components](#)**

Common type tree components consist of: bankfiletoxml.mtt, bic.mtt, and ccy.mtt

---

## Common data components

Common data components consist of the bic.xml file; currencycodedecimals.xml file; and the swiftnnnnxsd4si.jar file.

The common data components are described here:

- data/bic.xml - This file is used for validating BIC values in the transaction data. The bic.xml file provided by the pack is a sample file that provides a starting point for using BIC validation.
- data/currencycodedecimals.xml - This file is used for validating currency code and country code values in the transaction data.
- data/swiftnnnnxsd4si.jar - This file is used for uploading the XML schemas from the Pack to the IBM Sterling B2B Integrator repository. See [Deploying the XML schemas to IBM Sterling B2B Integrator \(Deprecated\)](#).

---

## Common map components

The bictoxml and currencytoxml executable maps are under common/maps/bankfiletoxml.

- bictoxml map - This map creates the bic.xml file based on the FI file in the BIC directory. This file is included in the BIC directory and can be downloaded from SWIFT.

Input Card 1: BIC

Output Card 1: BICxml - The output generates name-value pairs of the FI file.

An example of the output is shown here:

```
<BIC
id="_home_markdown_jenkins_workspace_Transform_in_SSVD8_11.0.1_com.ibm.help.spe_fsp_pk.doc_Swift_901_concepts_c_pack_swif
t_common_map_components_TESTCC37XXX" subtype="BEID">Test Bank</BIC>
```

- BIC id - Represents the actual 8-character, or 11-character formatted BIC.

- subtype - Represents the precise type of financial institution, for example, a bank, or a broker.

- element value - This is the description of the institution.

- currencytoxml map - This map creates the currencycodedecimals.xml file, based on the CU file in the BIC directory. This file is included in the BIC directory that can be downloaded from SWIFT.

Input Card 1: Currency

Output Card 1: Currencyxml. The output generates name-value pairs of the CU file.

An example of the output is shown here:

```
<currencycode
id="_home_markdown_jenkins_workspace_Transform_in_SSVD8_11.0.1_com.ibm.help.spe_fsp_pk.doc_Swift_901_concepts_c_pack_swif
t_common_map_components_EUR" countrycode="DE">2</currencycode>
```

- currencycode id - Represents the currency corresponding to the countrycode.

- country code - Two character country representation.

- element value - Number of fractional decimal places allowed for the currency and country code combination

---

## Common schema components

The bic.xsd and the ccy.xsd schemas are common schema components.

The common schema components are described here:

- schemas/bic.xsd - This schema is used by the bic.mtt type tree, and provides the formatting rules for the bic.xml data file.

- schemas/ccy.xsd - This schema is used by the ccy.mtt type tree, and provides the formatting rules for the currencycodedecimals.xml data file. There are two groups in this schema. The default group uses both the currency and country codes, and there is an additional definition that uses the country code and currency codes independently.

---

## Common type tree components

Common type tree components consist of: bankfiletoxml.mtt, bic.mtt, and ccy.mtt

Common type tree components consist of the following:

- type\_trees/bankfiletoxml.mtt - This type tree is used with the bankfiletoxml to convert SWIFT provided BIC provider list data to the bic.xml file used by many of the maps for BIC validation.
- type\_trees/bic.mtt - This type tree is used with the bic.xml data file and contains the structures defined in the bic.xsd.
- type\_trees/ccy.mtt - This type tree is used with the currencycodedecimals.xml data file and contains the structures defined in the ccy.xsd.

---

## SWIFT MT Components

For a description of the supported SWIFT standards, see the release notes.

- **[SWIFT MT core type trees](#)**

The following type trees are provided to describe the SWIFT MT standard:

- **[Other SWIFT MT trees](#)**

Other trees that are supplied with the MT components are described in the table provided here.

- **[Java Validation Component](#)**

The Java Validation Component (JVC) provides a framework for validating SWIFT messages.

- **[Logical Message Format](#)**

The Logical Message Format (LMF) provides an independent representation of SWIFT messages, which are based on business elements and classes.

- **[Universal Confirmation Components](#)**

The Universal Confirmation (UC) validation component provides a framework to generate and validate the Universal Confirmation messages.

---

## SWIFT MT core type trees

The following type trees are provided to describe the SWIFT MT standard:

- swift\_iso7775\_ccyy
- swift\_iso15022\_ccyy

The primary purpose of the type trees listed above is to provide a full syntactic definition of the SWIFT MT user-to-user messages, thereby allowing the maps to be created to use these type trees for easy integration and exchange of SWIFT MT messages with other applications.

The SWIFT type trees contain all of the user-to-user messages (Categories 1 through 9), plus the Service and System Message definitions (Category 0).

The SWIFT MT type trees are installed in the following location:

<install\_dir>/packs/financial\_payments\_vn.n.n.n/swift/mt/type\_trees

Throughout this documentation, <install\_dir> indicates the installation location of the Pack for Financial Payments, and n.n.n.n represents the current version of the Pack for Financial Payments .

- **[Syntax validation in SWIFT MT type trees](#)**
- **[Supported message types in SWIFT MT type trees](#)**

---

## Syntax validation in SWIFT MT type trees

The syntactic definition covers the following elements:

- Blocks 1, 2, 3, and 5 - envelope validation:
  - Check the block structure.
  - Ensure that only permitted fields are present, and in the correct order.
- Block 4 - message body validation:
  - Make certain that only permitted sequences and fields are present.
  - Check the sequence structure and the order of sequences.
  - Check the order of fields within the message or within the sequence.
  - Check for the allowed format options of a field (for example, 98A and 98C).
  - Check the structure of sub-fields with a field.
  - Check for valid qualifiers of a field.
  - Check the cardinality of qualifiers, fields, and sequences (that is, optional, mandatory, repeating).
- Data size
  - For each field, or subfield, check the minimum and maximum size restrictions.
- Data type
  - For each field, or subfield, check data type (date, integer, decimal, text). For text fields, character set validation is not performed.
- Code words

- For each field, or subfield, check for a valid code word. Code words are only validated when an error code is quoted in the SWIFT User Handbook.

## Supported message types in SWIFT MT type trees

There are two type trees that contain the standards information for the MT message types, and the type tree name contains the year of support for the version of the MT messages:

- [swift\\_iso7775\\_ccyy](#)
- [swift\\_iso15022\\_ccyy](#)

See the tables provided here for a list of the MT messages supported in each of the type trees.

Note: The SWIFT type trees do not recognize the message separators used with any of the various SWIFTNet Interface products (for example, the S-Block used by SWIFT Alliance Access). See the SWIFT MT Other type trees documentation for details on parsing messages containing message separators.

- [\*\*swift\\_iso7775\\_ccyy\*\*](#)

These SWIFT MT messages are supported in the swift\_iso7775\_ccyy type tree.

- [\*\*swift\\_iso15022\\_ccyy\*\*](#)

These SWIFT MT message types are supported in the swift\_iso15022\_ccyy type tree.

## swift\_iso7775\_ccyy

These SWIFT MT messages are supported in the swift\_iso7775\_ccyy type tree.

<b>Category 0: Service and System Messages</b>
<b>Service Messages</b>
#15(x), #21, #22
<b>System Messages</b>
MT008, MT009, MT010, MT011, MT012, MT015, MT019, MT020, MT021, MT022, MT023, MT024, MT025, MT028, MT029, MT030, MT031, MT032, MT035, MT036, MT037, MT041, MT042, MT043, MT044, MT045, MT046, MT047, MT048, MT049, MT051, MT052, MT055, MT056, MT057, MT061, MT062, MT063, MT064, MT065, MT066, MT067, MT068, MT069, MT070, MT071, MT072, MT073, MT074, MT077, MT081, MT082, MT083, MT090, MT092, MT094, MT096, MT097
See the FIN System Messages manual in the SWIFT User Handbook for more details on Service and System messages.
<b>Category 1: Customer Payments and Cheques</b>
MT101, MT102, MT103, MT104, MT105, MT107, MT110, MT111, MT112
Note:
<ul style="list-style-type: none"> <li>MT102 Core, MT103 Core - Tag 119 in Block 3 User Header not = "STP"</li> <li>MT102 Plus, MT103 Plus - Tag 119 in Block 3 User Header = "STP"</li> </ul>
<b>Category 2: Financial Institution Transfers</b>
MT200, MT201, MT202, MT202_COV, MT203, MT204, MT205, MT205_COV, MT210, MT293, MT295, MT296
Note: MT202_COV and MT205_COV - Tag 119 in Block 3 User Header = "COV"
Note: MT295 - use MTn95 - see Common Group Messages
Note: MT296 - use MTn96 - see Common Group Messages
<b>Category 3: Treasury Markets (FX, Money Markets and Derivatives)</b>
MT300, MT304, MT305, MT306, MT320, MT330, MT340, MT341, MT350, MT360, MT361, MT362, MT364, MT365
<b>Category 4: Collections and Cash Letters</b>
MT400, MT410, MT412, MT416, MT420, MT422, MT430, MT450, MT455, MT456, MT499
<b>Category 5: Securities Markets</b>
MT516, MT526, MT53x, MT559, MT581
<b>Category 6: Treasury Markets (Commodities, Syndications &amp; Reference Data)</b>
MT600, MT601, MT604, MT605, MT606, MT607, MT608, MT620
<b>Category 7: Documentary Credits and Guarantees</b>
MT700, MT701, MT705, MT707, MT708, MT710, MT711, MT720, MT721, MT730, MT732, MT734, MT740, MT742, MT744, MT747, MT749, MT750, MT752, MT754, MT756, MT759, MT760, MT761, MT765, MT767, MT768, MT769, MT775, MT785, MT786, MT787, MT798, MT799
<b>Category 8: Travellers Cheques</b>
MT801, MT802
<b>Category 9: Cash Management &amp; Customer Status</b>
MT900, MT910, MT920, MT935, MT940, MT941, MT942, MT950, MT970, MT971, MT972, MT973, MT985, MT986
<b>Category n: Common Group Messages</b>
MTn90 - Used for MT190, 290, 390, 490, 590, 690, 790, 890, 990
MTn91 - Used for MT 191, 291, 391, 491, 591, 691, 791, 891, 991
MTn92 - Used for MT 192, 292, 392, 492, 592, 692, 792, 892, 992
MTn95 - Used for MT 195, 295, 395, 495, 595, 695, 795, 895, 995
MTn96 - Used for MT 196, 296, 396, 496, 596, 696, 796, 896, 996
MTn98 - Used for MT 198, 298, 398, 498, 598, 698, 798, 898, 998
MTn99 - Used for MT 199, 299, 399, 599, 699, 899, 999

## swift\_iso15022\_ccyy

These SWIFT MT message types are supported in the swift\_iso15022\_ccyy type tree.

Category 3: Treasury Markets (FX, Money Markets and Derivatives)
MT307, MT321, MT370, MT380, MT381
Category 5: Securities Markets
MT500, MT501, MT502, MT503, MT504, MT505, MT506, MT507, MT508, MT509, MT510, MT513, MT514, MT515, MT517, MT518, MT519, MT524, MT527, MT530, MT535, MT536, MT537, MT538, MT540, MT541, MT542, MT543, MT544, MT545, MT546, MT547, MT548, MT558, MT564, MT565, MT566, MT567, MT568, MT569, MT575, MT576, MT578, MT584, MT586
Category 6: Treasury Markets (Commodities, Syndication, and Reference Data)
MT670, MT671

## Other SWIFT MT trees

Other trees that are supplied with the MT components are described in the table provided here.

genericfin	General-purpose trees for processing batches of SWIFT messages.
ir_fin_15022_ccyy	Used by the LMF maps for creating ISO 15022 messages.
lmf_ccyy	Main LMF trees.
swiftroute	General-purpose trees for general processing.
validator	Used by the JVC maps for starting the JVC.

The other SWIFT trees provide support for various activities within the pack. These trees can be used as templates and examples if it is required.

Note: These trees are tested for the various activities for which they are used within the pack, but they are not tested for all possible uses. Test trees thoroughly if you use them for your own purposes. Testing is necessary to ensure that the type tree meets your requirements.

The purpose of the type tree is described as follows:

- genericfin - This type tree processes batches of SWIFT MT Messages. Each message is separated by various delimiters typically found in various SWIFTNet interface products.  
The batch of messages is defined within the Generic\ Batch group, which contains a repeating Message element. Message is partitioned into four types: AckNack, System Message, FINPay (MT121) and UserMessage (for MT Categories 1 - 9). Each message can be delimited by various delimiters: S-Block, SOH, space and \$.  
The type tree cannot process the detailed content of SWIFT MT messages, as Block 4 is defined as a 'Blob'. It instead provides a high-level overview of a batch of SWIFT MT messages.  
Note: A 'blob' is a type tree item that is typically used in representing any text strings that are interpreted as characters with any syntax, and no length restrictions.
- swiftroute - The type tree has various purposes. It is associated with control-logic and map-routing activities and for use as a 'scratchpad' to store and format text strings.
- lmf\_ccyy - This type tree is the main LMF type tree. See the Logical Message Format documentation for details on the LMF type tree. All elements in the type tree are text defined fields.
- ir\_fin\_15022\_ccyy- The type tree contains a generic representation of the messages from the iso\_15022 type tree for those MT messages that have LMF maps. The messages and the elements they contain are represented as generic business elements (for example, Party and Narrative) in order to simplify the output mapping of these message types.  
The type tree does not contain any syntax parsing capability. Do not use the type tree to parse SWIFT messages. This type tree is designed for use as output. All elements in the type tree are text defined fields.
- validator - The type tree is a simple type tree. It is used by the validate messages map to start the JVC. See the JVC documentation for details about validate\_messages.

## Java Validation Component

The Java Validation Component (JVC) provides a framework for validating SWIFT messages.

The JVC can be used to check messages before they are transmitted to the SWIFT network to ensure compliance. It provides a configurable validation framework, with error reporting functionality, that allows the source of the error to be quickly located.

- [Overview of the JVC](#)  
The Java Validation Component (JVC) validates the following items in accord with the latest SWIFT standards.
- [How to configure the JVC](#)  
The JVC is controlled by the jvalccyy.prop file.
- [JVC installation](#)  
When JVC is installed on Windows, the jvalccyy.jar and jvcwrap.jar files are placed in the ITX installation directory.

## Overview of the JVC

The Java Validation Component (JVC) validates the following items in accord with the latest SWIFT standards.

- Message syntax - presence and order of fields and sequences.
- Field syntax - size and data type.
- Field and sequence cardinality - optional, mandatory, repeating.
- Codewords
- Network validated rules at both message and field level
- Message envelope syntax - SWIFT header and trailer.
- Market practice rules.

The JVC validates all user-to-user messages, that is, all of Category 1 (MT1nn), through Category 9 (MT9nn). Category 0 (MT0nn) Service and System Messages are not validated by the JVC.

## How to configure the JVC

The JVC is controlled by the jvalccyy.prop file.

In addition, certain properties can be overridden during map execution by the supplied map validate\_messages .

- [\*\*jvalccyy.prop\*\*](#)  
The jvalccyy.prop file contains a number of configurable parameters for controlling the behavior of the JVC.
- [\*\*How to specify a jvalccyy.prop file location\*\*](#)  
The location for the jvalccyy.prop file can be specified in a number of ways. The following options are available, and are shown in the sequence which the JVC will use to locate jvalccyy.prop.
- [\*\*message\\_definitions.directory\*\*](#)  
The message\_definitions.directory identifies the location of the metadata files used by the JVC for syntactic validation, and for enabling and disabling message-specific field level and Network Rule validation.
- [\*\*validate\\_messages\*\*](#)  
The validate messages map can be used to start the JVC. It can be found in  
`<install_dir>/packs/financial_payments_vn.n.n.n/swift/mt/jvc/maps/validate_messages`

## jvalccyy.prop

The jvalccyy.prop file contains a number of configurable parameters for controlling the behavior of the JVC.

These parameters tell the JVC where to find other files that it uses, as well as specifying certain validation options.

When running a map that calls the JVC, the jvalccyy.prop file can be located in a directory of your choosing. Refer to the section *How to specify jvalccyy.prop location* for more details. If the jvalccyy.prop file is not found on start up, the JVC terminates with an error message.

Note that the recommended best practice is to use a single jvalccyy.prop file for each version of the pack. This is due to several factors, including performance considerations and the fact that jvalccyy.prop contains locations of various other components, such as BIC files, currency files, and XML metadata files. If there is a requirement to use different maps with different validation settings, then the JVC Validator override flags can be used to achieve this. Refer to the validate\_messages section for details.

The parameters set by the jvalccyy.prop file include swift.pack.version=srgccy

This is the current version of the Pack for Financial Payments and is not to be changed. It is used to validate that the correct version of jvalccyy.jar is used.

If this value does not match the version number contained in the jvalccyy.jar, the JVC terminates with an error message.

message.definition.directory

This parameter identifies the location of the metadata files used by the JVC for syntactic validation, and for enabling and disabling message-specific field level and Network Rule validation.

If the metadata files are not found in the specified location, the JVC will terminate with an error message.

rules.networkvalidation.enabled = true|false

- true - The default value, and the recommended setting. It performs all syntax and semantic validation, including validation defined by the “special functions” in the SWIFT manuals (for example “Party Field Option J”, “Payments Reject/Return” and so forth).
- false - Performs only basic syntax validation.

bic.file.enabled=true|false

- true - Indicates that the BIC/BEI validation is required.
- false - Default value. Indicates that BIC/BEI validation is not required

Note: If bic.file.enabled=false, then the following Network Rules are also disabled:

- C05
- E57
- T27

bic.definition.file

This parameter identifies the location of the bic.xml lookup file used by the JVC for BIC/BEI validation. .

If bic.file.enabled=true and the bic.xml file is not found, the JVC terminates with an error message.

If bic.file.enabled=false, the JVC does not attempt to open the bic.xml file

`countrycurrencycodes.definition.file`

This parameter identifies the location of the `currencycodedecimals.xml` lookup file used by the JVC for Currency Code and Country Code validation.

`ReportingMode=verbose|terse`

- `verbose` - This is the default value. Validation results and error messages are returned in a multi-line, XML-like syntax.
- `terse` - Validation results and error messages are usually returned as a single line of plain text. Some messages may contain multiple lines, if they include data from a multi-line field that was in error.

`validateCopyofFields=true|false`

- `true` - Causes validation to be performed on the `CopyOfFields` elements in message types MTn92, MTn95, and MTn96.
- `false` - This is the default value. Causes `CopyOfFields` validation not to be performed.

`FileActMode.enabled=true|false`

Controls the validation for Tag 51A in MTs 101, 102, 103, 103\_REMIT, 104, 107, and 416. No other MTs, Tags, or validation rules are affected.

- `true` - Enable FileAct mode, and allow Tag 51A to be present.
- `false` - Disable FileAct mode, and return error code D63 if Tag 51A is present.

Note: If `FileActMode.enabled` is not present in `jvalccyy.prop`, or is present but contains any other value, validation checking defaults to `FileActMode.enabled=false`.  
`MessageLength.enabled=true|false`

Controls the validation for message length for all MT messages.

- `true` – Enable message length validation and return error code M50 if the message exceeds the permitted length.
- `false` – Disable message length checking.

Note: If `MessageLength.enabled` is not present in `jvalccyy.prop`, or is present but contains any other value, validation checking will default to `MessageLength.enabled=true`.

`checkE76.CCY=ccyymmdd`

where CCY is a former European currency used in E76 error validation, and `ccyymmdd` is the value date after which this currency must not be used. These parameters should not be changed in normal operation.

`rules.marketpractice.enabled=true|false`

- `true` - Market Practice validation is performed based upon the SWIFT document V1.1 published August 26, 2002.
- `false` - This is the default value. Market Practice validation is not performed.

SWIFT has discontinued this document, therefore, this feature is no longer maintained.

`rules.marketpractice.baseclass`

Indicates the base class name used for Market Practice validation. The default value shown here should not be changed:

`com.ibm.websphere.dtx.ip.swift.srgccyy.validation`

`rules.marketpractice.MTnnn`

Indicates the rules that are to be validated for a particular message. The rule name (for example S101) is appended to the `BaseClass` to obtain the class to use. For example:

`com.ibm.websphere.dtx.ip.swift.srgccyy.validation.S101`

These features are used for enabling and configuring basic Java logging capabilities:

- `log.enable=true|false` - Default is false.
- `log.filename=` Default value for `swiftlog.txt`
- `log.append=true|flase` - Default is false.
- `log.filesize=1000000` - Maximum log file size in bytes before file is recycled.
- `log.level=ALL` - Allowable values are OFF, ALL (default), SEVERE, WARNING, INFO, FINE

Editing will be required to change `PACK_INSTALL_DIR` with actual location where the pack zip distro was extracted.

## How to specify a `jvalccyy.prop` file location

The location for the `jvalccyy.prop` file can be specified in a number of ways. The following options are available, and are shown in the sequence which the JVC will use to locate `jvalccyy.prop`.

Note: To set up the `jvalccyy.prop` file for the ITX Design Server, see the instructions for the `swiftMTjvcConfig.tar.gz` file that are contained in the release notes.

### 1. Use a JVM system property

Specify the location by using a JVM system property called "`DTX_SWIFT_CONFIG_DIR`" that contains the fully qualified path of the directory containing the `jvalccyy.prop` file.

JVM system properties are set by using the `-D` option. This can be specified in a number of ways:

- Update the `config.yaml` file with this command:

```
[JVM Options]optionN=<install_dir>/packs/financial_payments_vn.n.n.n/swift/mt/jvc/maps
```

where N is the next available option number.

Example:

```
[JVM Options]option1=<install_dir>/packs/financial_payments_v9.0.1.0/swift/mt/jvc/maps
```

- When using IBM Integration Bus, you can use the following command to add the JVM System Property DTX\_SWIFT\_CONFIG\_DIR to the IBM Integration Bus Execution Group:

```
mqsiclchangeproperties < BrokerName > -e < ExecutionGroupName > -o ComIbmJVMManager -n jvmSystemProperty -v"-DDTX_SWIFT_CONFIG_DIR=<absolute_directory>"
```

To check the jvmSystemProperty value of the JVM Option that is defined for the IBM Integration Bus Execution Group, use the following IBM Integration Bus command:

```
mqsiereportproperties < BrokerName > -e < ExecutionGroupName > -o ComIbmJVMManager -n jvmSystemProperty
```

## 2. Use the Global System environment variable

---

To use this method, specify the location using a global environment variable called DTX\_SWIFT\_CONFIG\_DIR that contains the fully-qualified path of the directory containing jvalccyy.prop. For example, when using the Command Server, set DTX\_SWIFT\_CONFIG\_DIR by using the Windows **SET** command or the UNIX **EXPORT** command.

## 3. Use the validate\_messages map

---

To use this method, set the rule for element JvalPropPath in Output Card 1 to the directory containing jvalccyy.prop. For example, ="c:/jvc/jvalccyy.prop"

## 4. Use the default location

---

To specify use of the default location, set the map rule for item "JvalPropPath" in "validate\_messages" to =NONE.

jvalccyy.prop is installed in <install\_dir>/packs/financial\_payments\_vn.n.n.n/swift/mt/jvc/maps for all versions of ITX.

For Transformation Extender V8.n, the jvalccyy.prop file can be located in any directory, but the map calling the JVC must be located in the same directory as jvalccyy.prop.

For ITX V9.5 jvalccyy.prop must be moved to the following location:

- Windows - <install\_dir>
- Linux® - <install\_dir>/bin

## How to choose the best method

---

The choice of method to use to define the location of the jvalccyy.prop is entirely dependent on individual circumstances, and no one method is better than another. Choose the method that is best suited to your environment.

As a guide, here is a brief summary of the advantages and disadvantages of the different methods:

- JVM System Property - The advantages and disadvantages are similar to those of using a Global Environment Variable, but apply only when ITX has to initiate a Java™ Virtual Machine (JVM) to run the JVC. For example, this option will not work with ITX Design Server.
- Global System Environment Variable - An advantage with this method is flexibility. The location can be changed without any need to recompile ITX maps or redeploy ITX components. You would, however, need to restart any running process, such as Launcher, after any changes are made to the variable. A disadvantage is that this is not an ITX component but a system-wide variable, and internal security policies may prevent access to these settings.
- "validate\_messages" map - The main advantage with this method is that all configuration is contained within ITX components, so it might be simpler to manage. The main disadvantage is that the map must be recompiled and redeployed, but this may not be an issue if the location rarely changes.
- Use the default location - This is the simplest method, and might be the preferred option for backwards compatibility with existing configurations. The main disadvantage is that you need to be aware of the default location, which varies between ITX versions.

## messagedefinitions.directory

---

The messagedefinitions.directory identifies the location of the metadata files used by the JVC for syntactic validation, and for enabling and disabling message-specific field level and Network Rule validation.

If the metadata files are not found in the specified location, the JVC will terminate with an error message.

rules.networkvalidation.enabled

Whether all syntax and semantic validation, including the validation defined by the "special functions" in the SWIFT manuals (for example "Party Field Option J", "Payments Reject/Return," and so forth), are performed. Valid values: true, false

true

Default value and recommended setting. All validation is performed.

false

Only basic syntax validation is performed.

bic.file.enabled

Whether BIC/BEI validation is required. Valid values: true, false

true

BIC/BEI validation is required.

false

Default value. BIC/BEI validation is not required.

If bic.file.enabled=true and the bic.xml file is not found, the JVC terminates with an error message.

#### bic.definition.file

Location of the bic.xml lookup file used by the JVC for BIC/BEI validation. The default location on ITX is

<install\_dir>/packs/financial\_payments\_vn.n.n.n/swift/common/data/bic.xml

If bic.file.enabled=true and the bic.xml file is not found, the JVC terminates with an error message.

#### countrycurrencycodes.definition.file

Location of the currencycodedecimals.xml lookup file used by the JVC for Currency Code and Country Code validation. The default location on ITX is

<install\_dir>/packs/financial\_payments\_vn.n.n.n/swift/common/data/currencycodedecimals.xml

The currencycodedecimals.xml file that is included with the product is an example only. You are responsible for providing your own currencycodedecimals.xml file, which is generated from data that you download from SWIFT or ISO. See the [Common map components](#) topic for details.

#### ReportingMode

Valid values: verbose, terse

##### verbose

Default. Validation results and error messages are returned in a multi-line, XML-like syntax.

##### terse

Validation results and error messages are usually returned as a single line of plain text. Some messages may contain multiple lines, if they include data from a multi-line field that was in error.

#### validateCopyofFields

Whether validation is performed on the CopyOfFields elements in message types MTn92, MTn95, and MTn96. Valid values: true, false

##### true

CopyOfFields validation is performed.

##### false

Default. CopyOfFields validation is not performed.

#### checkE76.CCY=ccyymmdd

where CCY is a former European currency used in E76 error validation. Valid value: ccyymmdd

##### ccyymmdd

The value date after which this currency must not be used. These parameters should not be changed in normal operation.

#### rules.marketpractice.enabled

This feature is no longer maintained. Indicates whether Market Practice validation is performed based upon the SWIFT document V1.1 published August 26, 2002.

Valid values: true, false

##### true

Market Practice validation is performed.

##### false

Default. Market Practice validation is not performed.

#### rules.marketpractice.baseclass

Indicates the Base Class name used for Market Practice validation. Do not change the default value shown.

##### com.ibm.websphere.dtx.ip.swift.srgccyy

Default value, where ccyy indicates the SRG version supported by the Pack for Financial Payments.

#### rules.marketpractice.MTnnn

The rules that are to be validated for a particular message. The rule name (for example, S101) is appended to the BaseClass to obtain the class to use.

##### com.ibm.websphere.dtx.ip.swift.srgccyy.validation.S101

Example value.

These features are used for enabling and configuring basic Java logging capabilities:

- log.enable=true|false - Default is false.
- log.filename= Default value for ITX: <install\_dir>/packs/financial\_payments\_vn.n.n.n/swift/swiftlog.txt
- log.append=true|false - Default is false.
- log.filesize=1000000 - Maximum log file size in bytes before file is recycled.
- log.level=ALL - Allowable values are OFF, ALL (default), SEVERE, WARNING, INFO, FINE

The following features in jvalccyy.prop are not currently in use:

- codelist.grammar.file - Not currently in use.
- 110n.locale=es - Not currently in use and commented out.
- 110n.country=ES - Not currently in use and commented out.

---

## validate\_messages

The validate messages map can be used to start the JVC. It can be found in <install\_dir>/packs/financial\_payments\_vn.n.n.n/swift/mt/jvc/maps/validate\_messages

This map can be incorporated directly into your workflow, or adapted to meet your needs. It can also be called from another map by using the RUN command. The map has one input and two output cards:

- Input Card 1 - SWIFT message, which is defined as a blob item type tree.
- Output Card 1 - Validator settings for defining pack version number (do not change in normal operation), validator flags, and jvalccyy.prop location.

- Output Card 2: ValidatorResults. This card is used to call the JVC by use of the JVCWrapper validate method. It uses parameters from Output Card 1 to collect the validation results.

Validator flags can be used to modify the behavior of the JVC. They are specified as a single eight-character string, and the position within the string determines which flag is being set. The meaning and usage of each flag is shown in the table.

Position	Name	Allowable Values	Description
1	Preload Definitions	0	Load message definitions as needed (recommended).
		1	Preload all message definitions
2	Map Server Mode	0	Enforce normal CRLF checking (recommended)
		1	Use <NL> as a line delimiter instead of <CR><LF>
3	Network Rules	0	Do not enforce network rules
		1	Enforce network rules
		2	Check jvalccyy.prop to determine if network rules enforced
4	Check Methods	0	Do not enforce field-level rules
		1	Enforce field-level rules
		2	Check metadata file to determine if field-level rules enforced
5	BIC Validation	0	Do not enforce BIC validation
		1	Enforce BIC validation
		2	Check jvalccyy.prop to determine if BIC validation enforced
6	Market Practice	0	Enforce market practice rules
		1	Do not enforce market practice rules
		2	Check jvalccyy.prop to determine if market practice rules enforced
7	FileActMode	0	FileActMode is disabled
		1	FileActMode is enabled
		2	Check jvalccyy.prop to determine FileActMode
8	MessageLength	0	MessageLength check is disabled
		1	MessageLength check is enabled
		2	Check jvalccyy.prop to determine MessageLength

Use a value of 0 or 1, for example, for the Network Rules, BIC Validation, or Market Practice flags in order to override the settings that are specified in the jvalccyy.prop file. This lets different maps, or map instances, that are in the same directory to use different validation criteria. For example, one map or map instance can enforce BIC validation, while another has BIC validation disabled.

Note that the flags are positional, and that the character position determines which parameter is being acted on. For example, to override the value for Flag position 8 (Message Length), all 7 previous Flags must also be supplied.

For backward compatibility with earlier pack versions, only the first six Flags are mandatory for Flag overrides. If only six flags are supplied, the remaining two flags default to the behavior that existed before the two parameter overrides were introduced, as follows:

- Flag 7 FileActMode=false
- Flag 8 MessageLength=true

## JVC installation

When JVC is installed on Windows, the jvalccyy.jar and jvcwrap.jar files are placed in the ITX installation directory.

For the TX version prior to 10.2.1, when JVC is installed on Windows, the jvalccyy.jar and jvcwrap.jar files are placed in the ITX installation directory.

For TX version 10.2.1 and up, the jvalccyy.jar and jvcwrap.jar will be included in swiftMTjvcConfig.tar.gz under the UIProjectImports folder. You must modify the CLASSPATH environment variable to include jvalccyy.jar and jvcwrap.jar.

On the Design Server, extract jvcwrap.jar and jvalccyy.jar files from swiftJVConfigIBM.tar.gz.

Copy jars to the location set on the Design Server config.yaml, under property server.persistence.libs (previously TX\_ADDITIONAL\_LIBS), for example: /opt/tlxlibs.

Restart the Design Server: ./ITX stop, and then ./ITX start

- [Installing JVC on non-Windows environments](#)

## Installing JVC on non-Windows environments

Copy all of the .jar files from your <itx\_runtime\_install\_dir>/extjar directory to the <itx\_runtime\_install\_dir>/libs directory on your non-Windows system.

## Logical Message Format

The Logical Message Format (LMF) provides an independent representation of SWIFT messages, which are based on business elements and classes.

To see the detailed mapping tables, go to <install\_dir>/packs/financial\_payments\_vn.n.n.n/swift/mt/lmf/mapping\_tables.

To see the detailed mapping tables see the swiftLMFmappingDoc.zip file.

- [LMF overview](#)  
The LMF provides a set of business messages that are independent of network message format or structure.
  - [LMF structure](#)  
The entities within the LMF, as defined in lmf\_ccyy, consist of Messages, Sequences, and Classes.
  - [LMF supported message types](#)  
The LMF components consists of two sets of type tree (lmf\_ccyy and ir\_fin\_15022\_ccyy), and a set of to\_lmf and from\_lmf maps for translating MT messages to LMF format, and LMF format to MT respectively.
  - [LMF message mapping](#)
- 

## LMF overview

The LMF provides a set of business messages that are independent of network message format or structure.

This allows different message formats to be mapped to each other through the LMF. When changes in the elements in those messages occur (for example, at SWIFT standards releases), changing the LMF reflects those changes and minimizes the impact on the integration maps with your host application.

The LMF is implemented as ITX type tree. The LMF is based on the ISO 15022 format, and was originally conceived as a way of assisting migration from SWIFT ISO7775 Payments, Statements and Securities messages to their ISO 15022 equivalents. For this reason, the internal structure of the LMF is similar to ISO 15022 business elements, and is described in this documentation.

Note: The LMF uses an XML-like syntax. There is no schema for the LMF, therefore, LMF instance data cannot be used in situations where the presence of a schema is required for correct operation.

---

## LMF structure

The entities within the LMF, as defined in lmf\_ccyy, consist of Messages, Sequences, and Classes.

- **Messages** - Business messages are the highest level concept in the LMF. These messages might correspond directly to an equivalent network message, or more often cover a number of different network messages. For instance, the LMF "Transfers" instruction message covers SWIFT MT 202 and MT 210.
  - **Sequences** - Each LMF message is constructed from a series of sequences, like the ones that are used in SWIFT messages. Each sequence might contain a number of other entities, either classes, or other sequences called subsequences.
  - **Classes** - Within each sequence, the individual business elements are called classes. Classes represent each actual business data element, such as a reference. Whereas network message formats contain fields, these are not present in the LMF. Instead, the classes represent types of data (for example, flags, indicators, and dates), and the specific data elements are instances of those classes.
- Each class consists of a set of attributes and subfields. The attributes describe how the class is being used. The subfields contain the actual data that is in use (for example, the date value, or the reference value). The attributes consist of:
- s - Not used.
  - f - Format value. The format value varies from class to class. It is enforced by a restriction list in the type tree. It describes the format of the data that is held in the class, for example DateOnly to represent the SWIFT 98A date format
  - q - Qualifier value. Varies from class to class. Enforced by a restriction list. It contains the qualifier that describes the business meaning of the class.
  - dss - Contains the data source scheme, if used. The data source scheme is a field that is used in SWIFT ISO 15022 messages.

## LMF supported message types

The LMF components consists of two sets of type tree (lmf\_ccyy and ir\_fin\_15022\_ccyy), and a set of to\_lmf and from\_lmf maps for translating MT messages to LMF format, and LMF format to MT respectively.

Each **to\_lmf** map uses one input card, and one output card:

Input Card 1: MT message in the current swift\_iso7775\_ccyy or swift\_iso15022\_ccyy type tree.

Output Card 1: LMF message corresponding to the MT message in the lmf type tree

Each **from\_lmf** map uses one input card, and one output card:

Input Card 1: LMF message corresponding to the MT message in the lmf type tree.

Output Card 1: MT message in either iso7775\_ccyy or ir\_fin\_15022\_ccyy type tree. For further information on this type tree see [Other SWIFT MT trees](#).

Supported MT Message(s)	To_lmf map name	From_lmf map name
MT101	swift_mt101_tolmf	swift_mt101_fromlmlf
MT102	swift_mt102_tolmf	swift_mt102_fromlmlf
MT103	swift_mt103_tolmf	swift_mt103_fromlmlf
MT103+	swift_mt103_tolmf	swift_mt103_fromlmlf
MT190	swift_mtn90_tolmf	swift_mtn90_fromlmlf
MT191	swift_mtn91_tolmf	swift_mtn91_fromlmlf
MT192	swift_mtn92_tolmf	swift_mtn92_fromlmlf
MT195	swift_mtn95_tolmf	swift_mtn95_fromlmlf
MT196	swift_mtn96_tolmf	swift_mtn96_fromlmlf
MT202	swift_mt202_tolmf	swift_mt202_fromlmlf
MT202 COV	swift_mt202_COV_tolmf	swift_mt202_COV_fromlmlf

Supported MT Message(s)	To_Lmf map name	From_Lmf map name
MT205	swift_mt205_tolmf	swift_mt205_fromlmf
MT205 COV	swift_mt205_COV_tolmf	swift_mt205_COV_fromlmf
MT210	swift_mt210_tolmf	swift_mt210_fromlmf
MT290	swift_mtn90_tolmf	swift_mtn90_fromlmf
MT291	swift_mtn91_tolmf	swift_mtn91_fromlmf
MT292	swift_mtn92_tolmf	swift_mtn92_fromlmf
MT295	swift_mtn95_tolmf	swift_mtn95_fromlmf
MT296	swift_mtn96_tolmf	swift_mtn96_fromlmf
MT300	swift_mt300_tolmf	swift_mt300_fromlmf
MT304	swift_mt304_tolmf	swift_mt304_fromlmf
MT305	swift_mt305_tolmf	swift_mt305_fromlmf
MT320	swift_mt320_tolmf	swift_mt320_fromlmf
MT390	swift_mtn90_tolmf	swift_mtn90_fromlmf
MT391	swift_mtn91_tolmf	swift_mtn91_fromlmf
MT392	swift_mtn92_tolmf	swift_mtn92_fromlmf
MT395	swift_mtn95_tolmf	swift_mtn95_fromlmf
MT396	swift_mtn96_tolmf	swift_mtn96_fromlmf
MT490	swift_mtn90_tolmf	swift_mtn90_fromlmf
MT491	swift_mtn91_tolmf	swift_mtn91_fromlmf
MT492	swift_mtn92_tolmf	swift_mtn92_fromlmf
MT495	swift_mtn95_tolmf	swift_mtn95_fromlmf
MT496	swift_mtn96_tolmf	swift_mtn96_fromlmf
MT502	swift_mt502_tolmf	swift_mt502_fromlmf
MT509	swift_mt509_tolmf	swift_mt509_fromlmf
MT515	swift_mt515_tolmf	swift_mt515_fromlmf
MT535	swift_mt535_tolmf	swift_mt535_fromlmf
MT536	swift_mt536_tolmf	swift_mt536_fromlmf
MT537	swift_mt537_tolmf	swift_mt537_fromlmf
MT540	swift_settinst_mt54x_tolmf	swift_settinst_mt54x_fromlmf
MT541	swift_settinst_mt54x_tolmf	swift_settinst_mt54x_fromlmf
MT542	swift_settinst_mt54x_tolmf	swift_settinst_mt54x_fromlmf
MT543	swift_settinst_mt54x_tolmf	swift_settinst_mt54x_fromlmf
MT544	swift_settconf_mt54x_tolmf	swift_settconf_mt54x_fromlmf
MT545	swift_settconf_mt54x_tolmf	swift_settconf_mt54x_fromlmf
MT546	swift_settconf_mt54x_tolmf	swift_settconf_mt54x_fromlmf
MT547	swift_settconf_mt54x_tolmf	swift_settconf_mt54x_fromlmf
MT548	swift_mt548_tolmf	swift_mt548_fromlmf
MT549	swift_mt549_tolmf	swift_mt549_fromlmf
MT564	swift_mt564_tolmf	swift_ca_event_fromlmf
MT565	swift_mt565_tolmf	swift_ca_event_fromlmf
MT566	swift_mt566_tolmf	swift_ca_event_fromlmf
MT567	swift_mt567_tolmf	swift_ca_event_fromlmf
MT568	swift_mt568_tolmf	swift_ca_event_fromlmf
MT578	swift_mt578_tolmf	swift_mt578_fromlmf
MT590	swift_mtn90_tolmf	swift_mtn90_fromlmf
MT591	swift_mtn91_tolmf	swift_mtn91_fromlmf
MT592	swift_mtn92_tolmf	swift_mtn92_fromlmf
MT595	swift_mtn95_tolmf	swift_mtn95_fromlmf
MT596	swift_mtn96_tolmf	swift_mtn96_fromlmf
MT690	swift_mtn90_tolmf	swift_mtn90_fromlmf
MT691	swift_mtn91_tolmf	swift_mtn91_fromlmf
MT692	swift_mtn92_tolmf	swift_mtn92_fromlmf
MT695	swift_mtn95_tolmf	swift_mtn95_fromlmf
MT696	swift_mtn96_tolmf	swift_mtn96_fromlmf
MT790	swift_mtn90_tolmf	swift_mtn90_fromlmf
MT791	swift_mtn91_tolmf	swift_mtn91_fromlmf
MT792	swift_mtn92_tolmf	swift_mtn92_fromlmf
MT795	swift_mtn95_tolmf	swift_mtn95_fromlmf
MT796	swift_mtn96_tolmf	swift_mtn96_fromlmf
MT890	swift_mtn90_tolmf	swift_mtn90_fromlmf
MT891	swift_mtn91_tolmf	swift_mtn91_fromlmf
MT892	swift_mtn92_tolmf	swift_mtn92_fromlmf
MT895	swift_mtn95_tolmf	swift_mtn95_fromlmf
MT896	swift_mtn96_tolmf	swift_mtn96_fromlmf
MT900	swift_mt900_tolmf	swift_mt900_fromlmf
MT910	swift_mt910_tolmf	swift_mt910_fromlmf
MT940	swift_mt940_tolmf	swift_mt940_fromlmf

Supported MT Message(s)	To_Lmf map name	From_Lmf map name
MT942	swift_mt942_tolmf	swift_mt942_fromlmf
MT950	swift_mt950_tolmf	swift_mt950_fromlmf
MT990	swift_mtn90_tolmf	swift_mtn90_fromlmf
MT991	swift_mtn91_tolmf	swift_mtn91_fromlmf
MT992	swift_mtn92_tolmf	swift_mtn92_fromlmf
MT995	swift_mtn95_tolmf	swift_mtn95_fromlmf
MT996	swift_mtn96_tolmf	swift_mtn96_fromlmf

## LMF message mapping

Logical Message Format (LMF) transformation maps are available for the following message types: Payments, Transfers, Treasury Markets, Trading, Statements, Settlement, Corporate Actions, Cash Management, and Common Group Messages.

- [LMF Header](#)  
Specific information and usage requirements (for example, mandatory, optional, not used, etc.) specific to mapping SWIFTNet messages into the LMF are given for each header attribute.
- [Payments](#)  
The following tables provide LMF Transformation Maps for payments messages.
- [Alternative LMF mapping for Tag 35B](#)  
The documentation provided here presents an alternative LMF mapping of Tag 35B, using the format given by the Usage Rules.

## LMF Header

Specific information and usage requirements (for example, mandatory, optional, not used, etc.) specific to mapping SWIFTNet messages into the LMF are given for each header attribute.

The attributes are shown in the order in which they appear in the LMF header.

Attribute	Description
LMFType	This specifies the LMF message type for the SWIFTNet message being mapped in. A restriction list of valid values is provided in the LMF; the valid values are listed in the LMFType attributes table. See <a href="#">LMFType attributes</a>
version	Set to =NONE. A valid version must be specified for this attribute to indicate the LMF version of the message. However, for ease of maintaining backward compatibility of LMF versions, a type tree syntax object is used to generate/validate this attribute. By default, the value for the current LMF version will be generated when mapping to LMF.
direction	Mandatory for SWIFT. Specifies the processing direction for the logical message. Therefore, the value must be O to signify an outbound message (from internal applications to be sent outside the organization). This will actually create a SWIFT inbound message. For messages received from the SWIFT network, this value will be I.
sndaddr	Mandatory for SWIFT. Specifies the LT address for the sender of the message. For messages received from SWIFT this will be populated by the Sender's LT.
rcvaddr	Mandatory for SWIFT. Specifies the address/identifier (typically a BIC) for the receiver of the message. This may be an 8, 11 or 12 character BIC. For messages received from SWIFTNet this will contain the LT of the receiver.
function	Optional for SWIFT. Specifies the function/reason for the logical message. A restriction list with valid values is provided in the LMF type tree. The value must be equal to the datafield function in the Msgfunct class if it is mapped to the LMF (for example, 22A in MT 300). If a Msgfunct class is not mapped to the LMF (e.g. MT 103), the value is normally set to NEWM, but you are free to choose any other value so long as it is in the restriction list.
Subfunction	Optional for SWIFT. Specifies the subfunction for the logical message. A restriction list with valid values is provided in the LMF type tree. The value must be equal to the datafield subfunction in the Msgfunct class if it is mapped to the LMF (e.g. 94A in MT300).
msgtype	Mandatory for SWIFT. The value varies according to the SWIFT message type. See <a href="#">LMFType attributes</a>
userref	Optional for SWIFT. Specifies any user reference material in the SWIFT User Header Block 3. Contains tags 103, 113, 108, 119, 115, 423, 106, 424, 111, 165 and 433, in that order, delimited by ^. See the SWIFT User Handbook for full definitions of these tags.
headertext	Optional for SWIFT. Specifies other header information. For messages sent to SWIFT (Input Messages), the value is set to: [[Priority Block]^*[Delivery Monitoring Code]^*[Obsolescence Period Code]^^*[Session Number]^*[Sequence Number]]  For messages received from SWIFT (Output Messages), the value is set to: [Input Time]^*[MIR]^*[Output Date]^*[Output Time]^*[Priority Block]^*[Session Number]^*[Sequence Number]
priority	Mandatory for SWIFT. Specifies the priority of the SWIFT message. Valid values are "N" (Normal) or "U" (Urgent).
Deliverymonitoring	Mandatory for SWIFT. Specifies the priority of the SWIFT message. Valid values are "N" (Normal) or "U" (Urgent).

- [LMFType attributes](#)  
The LMFType attribute specifies the LMF message type for the SWIFT message being mapped.
- [msgtype attributes](#)  
The LMF msgtype value varies according to SWIFT message type.

## LMFType attributes

The LMFType attribute specifies the LMF message type for the SWIFT message being mapped.

A restriction list of valid values is provided in the LMF. Valid values are:

<b>SWIFT message</b>	<b>LMF message</b>
MT101, MT102, MT103, MT103+	PAYMENT
MT202, MT210	TRANSFER
MT300, MT304, MT305,	FX
MT320	LOANDEPOSIT
MT502, MT509	TRADEORDER
MT515	TRADECONF
MT564, MT565, MT566, MT567, MT568	CORPEVENT
MT540, MT541, MT542, MT543, MT578	SETTINST
MT564, MT565, MT566, MT567, MT568	SETTCOMP
MT900, MT910	CONFIRMATION
MT940, MT942, MT950	STATEMENT
MTn90, MTn91, MTn92, MTn95, MTn96	COMMON

## msgtype attributes

The LMF msgtype value varies according to SWIFT message type.

These are the LMF msgtype values for the following SWIFT message types.

<b>SWIFT message</b>	<b>LMF msgtype</b>
MT101	101
MT102	102
MT103	103
MT103+	103+
MT202	202
MT210	210
MT300	300
MT304	304
MT305	305
MT320	320
MT502	502
MT509	537
MT515	515
MT535	535
MT536	536
MT537	537
MT540	540
MT541	541
MT542	542
MT543	543
MT544	544
MT545	545
MT546	546
MT547	547
MT548	548
MT564	564
MT565	565
MT566	566
MT567	567
MT568	568
MT578	578
MT900	900
MT910	910
MT940	940
MT950	950
MTn90	one of: 190, 290, 390, 490, 590, 690, 790, 890 or 990
MTn91	one of: 191, 291, 391, 491, 591, 691, 791, 891 or 991
MTn92	one of: 192, 292, 393, 492, 592, 692, 792, 892 or 992
MTn95	one of: 195, 295, 395, 495, 595, 695, 795, 895 or 995
MTn96	one of: 196, 296, 396, 496, 596, 696, 796, 896 or 996

## Payments

The following tables provide LMF Transformation Maps for payments messages.

- MT 101 request for transfer**

This table shows the mappings that are needed to map an MT 101 into the LMF.

## MT 101 request for transfer

This table shows the mappings that are needed to map an MT 101 into the LMF.

Note: In the table below, when a Party field, consisting of only an Account, or NCS code, is mapped to the LMF in one of the "C" tag fields (for example 57C), an additional Party structure is mapped to the LMF. This additional Party structure consists of only the Qualifier attribute, and no other fields are mapped. This additional "Empty" Party structure is there to signify that the MT field does not contain any Party data, but only Account data.

SWIFT: MT 101					LMF: payment_M			
Field	M/O	Seq	Tag	Rpt	Class	Qualifier	Format	Data Fields
<b>LMF Sequence: PAYMGENL</b>								
Sender's Reference	M	A	20		Reference	PAYM_BTCH	ReferenceVar	ReferenceVal
Customer Specified Reference	O	A	21R		Reference	PAYM_CSBR	ReferenceVar	ReferenceVal
Message Index/Total	M	A	28D		PageIndicator	PAGE	PageIndicatorVar	PageNumber ContinuationIndicator
Requested Execution Date	M	A	30		DateTime	PAYM_RQEX	DateOnly	DateVal
Authorization	O	A	25		Account	PAYM_AUTH	AccountNumber	AccountNumberVal
<b>End of LMF sequence: PAYMGENL</b>								
<b>LMF sequence: PAYMPRTY (separate sequence for each party; the sequence mod value is set to PAYM_INST)</b>								
Instructing Party	O	A	50C		Party	PAYM_INST	PartyBIC	BIC
	O	A	50L		Party	PAYM_INST	PartyAlternateId	AlternateId
<b>End of sequence: PAYMPRTY</b>								
<b>LMF sequence: PAYMPRTY (separate sequence for each party; the sequence mod value is set to PAYM_ORCU)</b>								
Ordering Customer	O	A	50G		Account	PAYM_CASH	AccountNumber	AccountNumberVal
					Party	PAYM_ORCU	PartyBIC	BIC
	O	A	50H		Account	PAYM_CASH	AccountNumber	AccountNumberVal
					Party	PAYM_ORCU	PartyNameAddress	NameAddress
	O	A	50F		Account (Account format)	PAYM_CASH	PartyAccountNumber	AccountNumberVal
					Account (FATFID format)	PAYM_CASH	PartyAccountNumber	AccountCodeVal AccountNumberVal AccountTypeCode
					Party	PAYM_ORCU	PartyNameAddress	NameAddress
<b>End of LMF sequence: PAYMPRTY</b>								
<b>LMF sequence: PAYMPRTY (separate sequence for each party; the sequence mod value is set to PAYM_ORDI)</b>								
Account Servicing Institution	O	A	52A		Account	PAYM_CASH	AccountTypeNumberOnly	AccountTypeCode AccountNumberVal
					Party	PAYM_ORDI	PartyBIC	BIC
					*Party	PAYM_ORDI	PartyAlternateId	TypeOfId AlternateId
	O	A	52C		Account	PAYM_CASH	AccountNumber	AccountNumberVal
					*Party	PAYM_ORDI	PartyAlternateId	TypeOfId AlternateId
<b>End of LMF sequence: PAYMPRTY</b>								
<b>LMF sequence: PAYMPRTY (separate sequence for each party; the sequence mod value is set to PAYM_SNDI)</b>								
Sending Institution	O	C	51A		Account	PAYM_CASH	AccountTypeNumberOnly	AccountTypeCode AccountNumberVal
					Party	PAYM_SNDI	PartyBIC	BIC
<b>End of LMF sequence: PAYMPRTY</b>								
<b>LMF sequence: PAYDET)</b>								
Transaction Reference	M	B	21		Reference	PAYM_TRRF	ReferenceVar	ReferenceVal
F/X Deal Reference	O	B	21F		Reference	PAYM_FXRF	ReferenceVar	ReferenceVal

SWIFT: MT 101					LMF: payment_M			
Field	M/O	Seq	Tag	Rpt	Class	Qualifier	Format	Data Fields
Instruction Code	O	B	23E	R	Indicator	PAYM_INSC	IndicatorDesc	IndicatorVal DescVal
Transaction Amount	M	B	32B		Amount	PAYM_TRAN	AmountCurrOnl y	CurrencyCode AmountVal
Remittance Information	O	B	70		Narrative	PAYM_RMIF	Narrative4Lines	NarrativeVal
Regulatory Reporting	O	B	77B		Narrative	PAYM_RGRP	NarrativeData	NarrativeVal
Original Ordered Amount	O	B	33B		Amount	PAYM_ORDR	AmountCurrOnl y	CurrencyCode AmountVal
Details of Charges	M	B	71A		Indicator	PAYM_CHGD	IndicatorOnly	IndicatorVal
Charges Account	O	B	25A		Account	PAYM_CHAR	AccountNumber	AccountNumberV al
Exchange Rate	O	B	36		Rate	PAYM_EXCH	RateOnly	RateVal
<b>End of LMF sequence: PAYDET</b>								
<b>LMF sequence: PAYMDET\PAYMPRTY (separate sequence for each party; the sequence mod value is set to PAYM_INST)</b>								
Instructing Party	O	A	50C		Party	PAYM_INST	PartyBIC	BIC
	O	A	50L		Party	PAYM_INST	PartyAlternateId	AlternateId
<b>End of sequence: PAYMPRTY</b>								
<b>LMF sequence: PAYMDET\PAYMPRTY (separate sequence for each party; the sequence mod value is set to PAYM_ORCU)</b>								
Ordering Customer	O	B	50G		Account	PAYM_CASH	AccountNumber	AccountNumberV al
					Party	PAYM_ORCU	PartyBIC	BIC
	O	B	50H		Account	PAYM_CASH	AccountNumber	AccountNumberV al
					Party	PAYM_ORCU	PartyNameAddr ess	NameAddress
	O	B	50F		Account (Account format)	PAYM_CASH	PartyAccountNu mber	AccountNumberV al
					Account (FATFID format)	PAYM_CASH	PartyAccountNu mber	AccountCodeVal AccountNumberV al AccountTypeCod e
					Party	PAYM_ORCU	PartyNameAddr ess	NameAddress
<b>End of LMF sequence: PAYMDET\PAYMPRTY</b>								
<b>LMF sequence: PAYMDET\PAYMPRTY (separate sequence for each party; the sequence mod value is set to PAYM_ORDI)</b>								
Account Servicing Institution	O	A	52A		Account	PAYM_CASH	AccountTypeNu mberOnly	AccountTypeOnly AccountNumberV al
					Party	PAYM_ORDI	PartyBIC	BIC
					*Party	PAYM_ORDI	PartyAlternateId	TypeOfId AlternateId
	O	A	52C		Account	PAYM_CASH	AccountNumber	AccountNumberV al
					*Party	PAYM_ORDI	PartyAlternateId	TypeOfId AlternateId
<b>End of LMF sequence: PAYMDET\PAYMPRTY</b>								
<b>LMF sequence: PAYMDET\PAYMPRTY (separate sequence for each party; the sequence mod value is set to PAYM_INTM)</b>								
Intermediary	O	B	56A		Account	PAYM_CASH	AccountTypeNu mberOnly	AccountTypeCod e AccountNumberV al
					Party	PAYM_INTM	PartyBIC	BIC
					*Party	PAYM_INTM	PartyAlternateId	TypeOfId AlternateId
	O	B	56C		Account	PAYM_CASH	AccountNumber	AccountNumberV al
					*Party	PAYM_INTM	PartyAlternateId	TypeOfId AlternateId
	O	B	56D		Account	PAYM_CASH	AccountTypeNu mbeOnly	AccountTypeCod e AccountNumberV al
					Party	PAYM_INTM	PartyNameAddr ess	NameAddress
					*Party	PAYM_INTM	PartyAlternateId	TypeOfId AlternateId
<b>End of LMF sequence: PAYMDET\PAYMPRTY</b>								
<b>LMF sequence: PAYMDET\PAYMPRTY (separate sequence for each party; the sequence mod value is set to PAYM_ACCW)</b>								

SWIFT: MT 101					LMF: payment_M			
Field	M/O	Seq	Tag	Rpt	Class	Qualifier	Format	Data Fields
Account with Institution	O	B	57A		Account	PAYM_CASH	AccountTypeNumberOnly	AccountTypeCode AccountNumberValue
					Party	PAYM_ACCW	PartyBIC	BIC
					*Party	PAYM_ACCW	PartyAlternateId	TypeOfId AlternateId
	O	B	57C		Account	PAYM_CASH	AccountNumber	AccountNumberValue TypeOfId AlternateId
					*Party	PAYM_ACCW	PartyAlternateId	TypeOfId AlternateId
	O	B	57D		Account	PAYM_CASH	AccountTypeNumberOnly	AccountTypeCode AccountNumberValue
					Party	PAYM_ACCW	PartyNameAddress	NameAddress
					*Party	PAYM_ACCW	PartyAlternateId	TypeOfId AlternateId
<b>End of LMF sequence: PAYMDET\PAYMPRTY</b>								
<b>LMF sequence: PAYMDET\PAYMPRTY (separate sequence for each party; the sequence mod value is set to PAYM_BENE)</b>								
Beneficiary Customer	M	B	59A		Account	PAYM_CASH	AccountNumber	AccountNumberValue
					Party	PAYM_BENE	PartyBIC	BIC
		B	59		Account	PAYM_CASH	AccountNumber	AccountNumberValue
					Party	PAYM_BENE	PartyNameAddress	NameAddress
		B	59F		Account	PAYM_CASH	AccountNumber	AccountNumberValue
					Party	PAYM_BENE	PartyNameStr	NameAddress
<b>End of LMF sequence: PAYMDET\PAYMPRTY</b>								

## Alternative LMF mapping for Tag 35B

The documentation provided here presents an alternative LMF mapping of Tag 35B, using the format given by the Usage Rules.

In the SWIFT User Handbook, the Usage Rules for 35B state the following:

When an ISIN identifier is not used, one of the following codes must be used as the first characters of the Description of Security (Subfield 2):

- [/2!a/] The ISO two-digit country code, followed by the national scheme number.
- [/TS/] Followed by the ticker symbol.
- [/XX/] Bilaterally agreed, or proprietary scheme which can be further identified by a code, or short description identifying the scheme used.
- [/4!c/] Code identifying the type of security identifier used. This code must be one published by ISO 20022 (ExternalFinancialInstrumentIdentificationTypeCode).

Since Usage Rules are not enforced on the SWIFT Network, the SWIFT component does not enforce these rules either. Although SWIFT has suggested an alternative format for 35B, it is not validated. It is therefore up to you, as a user of this format, to ensure that it conforms to the format expected by the recipient of the message.

In the LMF maps, Tag 35B is treated as a simple text field, either with, or without an ISIN.

It is possible, however, to amend the LMF maps to separate these fields and map them individually to and from the LMF, if that is a requirement of your system.

Note: This alternative LMF mapping has not been incorporated into the LMF maps in the pack, and so any implementation of these rules must be reapplied to subsequent releases of the pack.

See the example for details.

- [ToLMF map](#)  
This example is based on MT540, but it is valid for any occurrence of 35B.
- [FromLMF map](#)  
A similar method can be applied to the FromLMF map to extract data from these sub-fields and map to the output MT message.

## ToLMF map

This example is based on MT540, but it is valid for any occurrence of 35B.

There will be a rule such as the following:

```
ToLMF_FinInstID (EITHER (ISIN# General Line SubField:ISIN<>IDOffI #35B NonISO15022 General Field:54x_BSequence , "N/A") ,
 EITHER (Narrative_35x General Line SubField[1] IN IDOffI #35B NonISO15022 General
Field:54x_BSequence , "N/A") ,
 EITHER (Narrative_35x General Line SubField[2] IN IDOffI #35B NonISO15022 General
```

```

Field:54x_BSequence , "N/A") ,
 EITHER (Narrative_35x General Line SubField[3] IN IDOffI #35B NonISO15022 General
Field:54x_BSequence , "N/A") ,
 EITHER (Narrative_35x General Line SubField[4] IN IDOffI #35B NonISO15022 General
Field:54x_BSequence , "N/A"))

```

This rule can be amended to look for the alternative format given in the Usage Rules, and call another functional map:

```

= IF (LEFT (Narrative_35x General Line SubField[1]:Narrative<>IDOffI #35B NonISO15022 General Field:54x_BSequence, 1) =
"/" &
(MID (Narrative_35x General Line SubField[1]:Narrative<>IDOffI #35B NonISO15022 General Field:54x_BSequence, 4, 1) ="/" |
MID (Narrative_35x General Line SubField[1]:Narrative<>IDOffI #35B NonISO15022 General Field:54x_BSequence, 6, 1) ="/")
&
SIZE (WORD (Narrative_35x General Line SubField[1]:Narrative<>IDOffI #35B NonISO15022 General Field:54x_BSequence, "/", 3
)) > 0,
ToLMF_FinInstID_AltrntId (Narrative<>IDOffI #35B NonISO15022 General Field:54x_BSequence),
ToLMF_FinInstID (EITHER (ISIN# General Line SubField:ISIN<>IDOffI #35B NonISO15022 General Field:54x_BSequence , "N/A") ,
EITHER (Narrative_35x General Line SubField[1] IN IDOffI #35B NonISO15022 General
Field:54x_BSequence , "N/A") ,
EITHER (Narrative_35x General Line SubField[2] IN IDOffI #35B NonISO15022 General
Field:54x_BSequence , "N/A") ,
EITHER (Narrative_35x General Line SubField[3] IN IDOffI #35B NonISO15022 General
Field:54x_BSequence , "N/A") ,
EITHER (Narrative_35x General Line SubField[4] IN IDOffI #35B NonISO15022 General
Field:54x_BSequence , "N/A"))

```

The rules for the ToLMF\_FinInstID\_AltrntId map are as follows, assuming an input card name of 35B\_Desc):

LMF Sub-field	Mapping rule
f	= "FinInstIDVar"
TickerSymbol_S	= WORD ( Narrative_35x General Line SubField[1]:35B_Desc, "/", 2 )
LocalID_S	= WORD ( Narrative_35x General Line SubField[1]:35B_Desc, "/", 3 )
NarrativeVal_S[1]	= Narrative_35x General Line SubField[2]:35B_Desc
NarrativeVal_S[2]	= Narrative_35x General Line SubField[3]:35B_Desc
NarrativeVal[3]	= Narrative_35x General Line SubField[4]:35B_Desc

All other fields can be set to NONE.

Using these rules, an input of the following:

```
:35B:/TS/IBM Ord
SCTY DESC EXTRA 1
SCTY DESC EXTRA 2
```

Gives the following LMF element:

```
<FinInstID s="" f="FinInstIDVar" q="">
 <TickerSymbol>TS</TickerSymbol>
 <LocalId>IBM Ord</LocalId>
 <NarrativeVal>SCTY DESC EXTRA 1</NarrativeVal>
</FinInstID>
```

## FromLMF map

A similar method can be applied to the FromLMF map to extract data from these sub-fields and map to the output MT message.

The following simple rule can be changed to look for the presence of LocalID and map accordingly:

Rule:

```
FromLMF_FinInstID_35B (FinInstID_C Class:TRADDET_LMF_In)
```

Changed rule:

```
= IF (PRESENT (LocalId_S SubField:FinInstID_C Class:TRADDET_LMF_In) ,
FromLMF_FinInstID_AltrntId (FinInstID_C Class:TRADDET_LMF_In) ,
FromLMF_FinInstID_35B (FinInstID_C Class:TRADDET_LMF_In))
```

The mapping rules for FromLMF\_FinInstID\_Altrntid are as follows, assuming an input card name of FinInstid\_IR:

FinInstID_C Field	Mapping rule
FieldTag	="35"
GeneralFormatOption	="B"
NarrativeVal_S[1]	="/" + TickerSymbol_S SubField:FinInstId_IR + "/" + LocalId_S SubField:FinInstId_IR
NarrativeVal_S[s]	=NarrativeVal_S SubField:FinInstId_IR

## Universal Confirmation Components

The Universal Confirmation (UC) validation component provides a framework to generate and validate the Universal Confirmation messages.

It is used to generate confirmations based from different channels:

- FIN MT103
- CSV file structure
- SWIFT gpi Tracker PaymentStatusRequest API

You can perform the checks on messages before transmission to ensure compliance. It provides a configurable transformation and validation framework with the error reporting functionality that helps to locate the source of the error.

- [Overview of the Universal Confirmation Components](#)  
The Universal Confirmation translation component supports the following:
- [Configuration Files](#)  
The Universal Confirmation validation and translation controlled by configuration files.

---

## Overview of the Universal Confirmation Components

The Universal Confirmation translation component supports the following:

- MT103 to MT199.
- MT103(s) to CSV file structure for payment confirmation.
- CSV file structure for payment confirmation of MT103 to MT199.
- MT103 to PaymentStatusRequest gpi API Tracker version 3.0.
- PaymentStatusRequest gpi API to MT199.

The Universal Confirmation validation component validates api, csv and fin messages.

MT103, MT199 messages in accordance with the latest SWIFT standard.

- Message syntax: Presence and order of fields and sequences.
- Field syntax: Size and data type.
- Field and sequence cardinality: Optional, mandatory, repeating.
- Codewords and constraints.
- Network validated rules at both message and field level.
- Market practice rules.

The CSV file structure for payment confirmation of MT103, FIN Universal Confirmations CSV file structure for payment confirmation of MT 103. standard.

The API messages are based from SWIFT gpi API Tracker version 3.0.

---

## Configuration Files

The Universal Confirmation validation and translation controlled by configuration files.

There are two types of configuration files:

- Validation Configuration Files.
- Translation Configuration Files.
- [Validation Configuration Files](#)  
The validation configuration files contain several configurable parameters to control the behavior of the Universal Confirmation validation example.
- [Translation Configuration File](#)
- [Configuration file location](#)  
The configuration files are required for the validation and translation maps to run successfully. If these files are absent, the map terminates.

---

## Validation Configuration Files

The validation configuration files contain several configurable parameters to control the behavior of the Universal Confirmation validation example.

These files are shipped with api and csv validation examples. These files are in the config folder and identified by the trailing codeword `_cfg` in the filename.

The configuration file contains the configurable parameters for each field to control the behavior of the Universal Confirmation validation. These parameters specify certain validation options.

When you run the api or csv validation maps, the configuration files must be present. See "Configuration file location" topic for more details.

If the configuration and translation files are not found, the map execution terminates.

---

### debug=true|false

Controls the content of validation results.

- true: This is a default value and the recommended setting. Validation results contain all field name, value, offset, data type, and error description (if there are any validation failures).
- false: When you set the value to false, it shows only the field that has an error with the field name, value, offset, data type, and error description.

---

### validate=true|false

This flag appears for each field or group and controls the validation of the respective field or group.

- true: This is a default value. It requires the field or group validation.

- false: When you set the value to false, it does not require the field or group validation.

## min=0|1

---

Identifies the field or group being optional or mandatory.

- 0: Indicates that the field or group is optional.
- 1: Indicates that the field or group is mandatory.

## Max=n

---

Identifies the number of occurrences of respected the field or group.

- n: Indicates the maximum number of times a field or group can be repeated.
- \*: Indicates that the field or group can be repeated infinite time.

## type

---

Identifies the type of field or group.

## constraints

---

Identifies the validation to be applied on the respected field or group.

## codelist

---

Contains the list of valid codewords for the respected field or group.

- **Validation Configuration File scenarios**

For example, camt.a01.001.04\_cfg.json file in universal\_confirmation/api/data/config has below configuration for field update\_payment\_scenario:

## Validation Configuration File scenarios

For example, camt.a01.001.04\_cfg.json file in universal\_confirmation/api/data/config has below configuration for field update\_payment\_scenario:

```
"update_payment_scenario": {
 "validate": true,
 "min": 1,
 "max": 1,
 "type": "PaymentScenario1Code",
 "codelist": [
 "COVE",
 "CCTR"
]
}
```

### "validate":

- If the Validate property is set to true, the field will be validated for configurations provided.
- If the Validate property is set to false, the field will not be validated for configurations provided.

"max": 1: Indicates only 1 maximum allowed occurrence for the field.

"codelist": Indicates the codewords "COVE" and "CCTR" being valid codewords for the field.

Constraints: Indicates the validation rules to be enforced.

## Translation Configuration File

---

The translation file uc\_translation\_cfg.json contains a few system generated parameters for creation of UC Message MT199, which are not present in input message api, csv or fin.

- **Translation Configuration File scenarios**

The translation file serves a purpose of holding system generated fields. Fields from the translation file are used to generate the output only, if they are not available in the input file.

## Translation Configuration File scenarios

The translation file serves a purpose of holding system generated fields. Fields from the translation file are used to generate the output only, if they are not available in the input file.

## General Information

---

The following fields can be set in the translation configuration file.

## MT header fields

---

session\_number: Mapped to field Session Number of block 1 of MT199.  
sequence\_number: Mapped to field Sequence Number of block 1 of MT199.  
logical\_terminal: Mapped to field DestinationAddress of block 2 of MT199.  
branch\_code: Mapped to field DestinationAddress of block 2 of MT199.  
priority\_block: Mapped to field PriorityBlock of block 2 of MT199.  
delivery\_monitoring\_code: Mapped to field DeliveryMonitoringCd of block 2 of MT199.  
obsolescence\_period\_code: Mapped to field ObsolescencePeriodCd of block 2 of MT199.  
business\_service: Mapped to field ServiceTypeID Tag111 of block 3 of MT199.  
update\_payment\_scenario: Mapped to field update\_payment\_scenario of api.

## PaymentStatusRequest

---

status: Mapped to field StatusCode of Tag79 Line2 of block 4 of MT199.  
reason: Mapped to field ReasonCode of Tag79 Line2 of block 4 of MT199.  
return: Mapped to field return of api.

## Additional information

---

forwarded\_to\_agent: Mapped to field ForwardedTo BIC of Tag79 Line3 of block 4 of MT199.  
settlement\_method: Mapped to field Settlmnt\_Method of Tag79 Line3 of block 4 of MT199.  
clearing\_system: Mapped to field Clearing\_System of Tag79 Line3 of block 4 of MT199.

## foreign\_exchange\_details

---

target\_currency: Mapped to field TargetCCY CUR of Tag79 Line5 of block 4 of MT199.  
exchange\_rate: Mapped to field ExchangeRate of Tag79 Line5 of block 4 of MT199.

## MT103 to CSV

---

In addition to above generic details, following functionality is specially for MT103 to csv translation.

In MT103 to csv, multiple occurrences of MT103 can be provided in the input file. A csv record (basic or advance) is generated for each MT103 instance.

The translation file for MT103 to csv can be provided with multiple occurrences.

In case of multiple occurrences of data provided in a translation file, reference of MT103 (Tag20) is matched with field instruction\_reference of translation configuration file in order to identify which occurrence of the translation file should be used for the current instance of MT103 to generate the csv record.

If the reference of MT103 (Tag20) is not found in the translation configuration file, an error is reported in the MT103\_config\_mismatch.out file stating reference to MT103 file.

If a single occurrence of data is provided in the translation file, the same instance is used to generate a csv record.

## CSV to MT199

---

Following functionality is specially for CSV to MT199 translation.

In csv to MT199, multiple occurrences of csv record (basic or advance) can be provided in the input file. MT199 instance are generated for each csv record.

The translation file for csv to MT199 can be provided with multiple occurrences.

In case, multiple occurrences of data is provided in uc\_translation\_cfg.json translation configuration file, Related\_Reference field of each CSV record is matched with field instruction\_reference of translation configuration file in order to identify which occurrence of translation configuration file should be used for current csv record to generate the MT199 instances.

If the field Related\_Reference of csv record is not found in the translation configuration file, an error is reported in the csv\_config\_mismatch.out file stating reference to that csv record.

If a single occurrence of data is provided in the translation file, the same instance is used to generate all MT199 instances.

## Configuration file location

---

The configuration files are required for the validation and translation maps to run successfully. If these files are absent, the map terminates.

The files are in <install\_dir>/packs/financial\_payments\_vn.n.n/n/swift/mt/examples/universal\_confirmation/\*/data/config directory.

Where, \* = api, csv, fin.

The configuration files are identified by the trailing codeword `_cfg` in the filename. All translation files named uc\_translation\_cfg.json are located at the above mentioned directory.

---

## SWIFT MX components

This documentation describes the SWIFT MX components.

For a description of specific versions of the SWIFT standards that are supported, see the release notes. The versions of the ITX base product that are required for this pack are described in the system requirements.

- [SWIFT MX components overview](#)

The schema versions that are included in the release are determined by publication date.

- [SWIFT MX schemas and type tree](#)

The SWIFT component of the Pack for Financial Payments includes all of the published SWIFT schemas, and also includes the type tree representation of these schemas.

- [MX validation](#)

This documentation describes MX validation.

---

## SWIFT MX components overview

The schema versions that are included in the release are determined by publication date.

The following are the main components of the SWIFT component MX:

- MX schemas and type tree
- MX validation framework
- MX schema validation maps
- MX extended validation maps
- Examples

## Terminology

---

The following terms are used throughout this documentation:

- **Extended validation** - In the MX Validation Framework, these rules are applied to the following generic MX fields:
  - XML elements of type BIC, or BICorBEI are checked for valid BIC or BEI.
  - XML elements that contain a currency code are checked for a valid code.
  - XML elements that contain a country code are checked for a valid code.
  - XML elements that contain an Amount and a Currency. These elements are checked for a valid currency and a valid number of decimal places as specified for that currency.
  - XML elements that contain an IBAN Identifier are checked for structure (country code, check digit, and bank account number)

## SWIFT MX schemas and type tree

---

The SWIFT component of the Pack for Financial Payments includes all of the published SWIFT schemas, and also includes the type tree representation of these schemas.

## MX validation

---

This documentation describes MX validation.

- [MX validation overview](#)

MX validation in the Pack for Financial Payments uses the following maps:

- [MX validation framework map](#)

The individual MX validation maps are controlled by this single executable map. The map source file is mx\_validation\_framework. The executable map name is swiftval.

- [MX configuration file](#)

The settings for MX validation are controlled by the mxconfig.xml file in the mx/maps directory.

- [Individual MX validation maps](#)

The MX validation maps validate for schema, BIC, IBAN, currency code, decimal places, and country codes for supported message types.

- [List of supported messages](#)

The supported MX messages are listed in the tables that are provided here.

- [Error reporting in the MX validation framework](#)

If errors are detected in the MX message, an error report is generated.

- [Adding maps to the MX validation framework](#)  
Additional XML messages can be added to the framework to perform XML schema validation and optionally your own extended validation rules.
- [How to use different levels of validation in the MX validation framework](#)  
You can perform different levels of validation for different message types.
- [Debugging the MX validation framework](#)  
The MX framework map is designed to record the steps it performs in parsing the data to determine the correct **RUN** map to use.

## MX validation overview

MX validation in the Pack for Financial Payments uses the following maps:

- Individual MX validation maps for each supported message type.
- A framework map that can be used to execute the maps, or as an example to build your own execution framework.
- [Framework map](#)  
The framework map works by attempting to identify the message type from the input data, by looking for the schema name in the schema name space.
- [MX validation maps](#)  
The individual MX validation maps are organized into map source files according to their business function.
- [MX schema validation maps](#)  
A separate map source file contains maps that only perform schema validation. The map source file is called `mx_schema_validation`, and contains a map for each supported MX message.

## Framework map

The framework map works by attempting to identify the message type from the input data, by looking for the schema name in the schema name space.

For example:

```
<Document xmlns="urn:iso:std:iso:20022:tech:xsd:colr.003.001.02"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
```

This information is used to determine the correct validation map name, and when it is identified, the relevant map is then run. An error report is produced if a validation failure is detected, and is also produced if it is not possible to identify the message type from the input data. See [Error reporting in the MX validation framework](#).

Configuration file, `mxconfig.xml`, is used to control the level of validation that is performed by each of the individual MX validation maps. See [MX configuration file](#).

If the message type cannot be identified from the input data, then, it is possible to provide the message type in the `mxconfig` file so that the framework map can identify the correct map to run. See [MX configuration file](#).

You can customize the framework to add your own validation maps, if required. But the relationship between schema name and map name must be maintained in order for the framework map to identify the correct validation map to run. See [Adding maps to the MX validation framework](#).

You can use different levels of validation for different message types, if required, by using separate configuration files for different message types. See [How to use different levels of validation in the MX validation framework](#).

The framework map can be run in debug mode to help with problem diagnosis. See [Debugging the MX validation framework](#). The framework map is called `swiftval`, and can be found in `validation_framework`. See [Framework map](#).

## MX validation maps

The individual MX validation maps are organized into map source files according to their business function.

For example, the `mx_acmt_validation` map source file contains the validation maps for all of the acmt messages. See [List of supported messages](#).

Each map is named according to a convention based on the MX schema name. For example:

Schema name	Validation map name
<code>pacs.008.001.04</code>	<code>ps008104_pacs_008_001_04</code>

The validation map name is derived from the schema name as follows:

Character 1 of the validation map name	represents character 1 of the schema name
Character 2 of the validation map name	represents character 4 of the schema name
Characters 3 - 5 of the validation map name	represents characters 6 - 8 of the schema name
Character 6 of the validation map name	represents character 12 of the schema name
Characters 7 and 8 of the validation map name	represents characters 14 and 15 of the schema name
Character 9 of the validation map name	represents the underscore character
Characters 10 - 24 of the validation map name	represents the full schema name, but with periods replaced with underscores

Note: There are exceptions to this naming scheme for the Application Header schema (`$ahV10.xsd`) and the E&I schemas that are named `swift.eni$xxxx.nnn.nnn.nn.xsd`. See [List of supported messages](#).

Each validation map performs a range of validation tasks. These following tasks are performed:

- Schema validation to check the MX instance conforms to the schema definition.
- BIC validation to ensure that only valid BICs are used.
- Country validation to ensure that only valid country codes are used.

- Currency validation to ensure that only valid currency codes are used.
- Decimal place validation to ensure the correct number of decimals per currency is used.
- IBAN validation to ensure that the IBAN contains a valid currency code and valid check digits as stated in the guidelines in the Swift MT User Handbook for IBANs (see Message Format Validation Rules, Part II Components, Chapter 4 Special Functions, topic IBAN).

Schema validation is always performed in the validation maps, but each of the other validation tasks can be turned off or on as required by using the supplied configuration file. See [MX configuration file](#).

The validation maps use the bic.xml and currencycodedecimals.xml files to verify BICs, Currency Codes, Country Codes, and Decimal Places information.

The bic.xml file contains a list of BICs created from the SWIFT published BIC address list.

The currencycodedecimals.xml file contains a list of countries, their country codes, and the number of decimal places their currency allows. This data is obtained from the SWIFT published list.

The SWIFT component of the Pack for Financial Payments includes a utility for creating both of these files. See [Common SWIFT components](#).

The individual MX validation maps can be run from the framework map, or they can be called individually as an independent executable map.

If you execute these maps outside of the framework map, then you must build your own error handling and error reporting routines. If so, the framework map can be used as an example in building your own routines See [Individual MX validation maps](#).

## MX schema validation maps

A separate map source file contains maps that only perform schema validation. The map source file is called mx\_schema\_validation , and contains a map for each supported MX message.

Use the MX schema validation maps only if you want to perform schema validation, and you do not want to perform extended validation. Be aware of the following points regarding these maps.

- The map names are the same as for the extended validation maps, as shown in the [List of supported messages](#).
- The maps use the same number of input and output cards as the extended validation maps.
- The maps are not compiled in the deploy script.
- If you want to use the maps instead of the extended validation maps, you must compile and deploy them manually. Compiling the maps overwrites the extended validation executable maps as the maps are named the same.

## MX validation framework map

The individual MX validation maps are controlled by this single executable map. The map source file is mx\_validation\_framework. The executable map name is swiftval.

The map has the following five inputs and three outputs:

Input or Output	Description
Input 1 (mx)	The MX data file. This file can be of any message type that is supported by the pack, or can be of a type that you add.
Input 2 (mxconfig)	The MX map configuration data. See“ <a href="#">MX configuration file</a> .
Input 3 (bicDirectory)	The BIC information file.
Input 4 (ccyCtryCodeList)	The Currency Code, Country Code, and Decimal Places information file.
Input 5 (errorMsgList)	The source file for all of the error messages.
Output 1 (Framework)	This card performs the main processing in the map, and it creates an audit log of the results. It determines the schema name from the input MX instance in input 1 (mx). If this is unsuccessful, it then tries to obtain the schema name from the mxconfig.xml file in input 2 (mxconfig). Having obtained the schema name, it then performs a <b>RUN</b> command to invoke the relevant validation map for this schema. If the map is unable to obtain the schema name from either of these two methods, the map will not perform any validation.
Output 2 (GatherResults)	This card analyzes the results of the <b>RUN</b> command. If the <b>RUN</b> command completed successfully, this card stores the result of the <b>RUN</b> command of the validation map. If the <b>RUN</b> command did not complete successfully, this card performs a <b>LOOKUP</b> on file errcodes.txt in input 5 (errorMsgList), to obtain the relevant error message.
Output 3 (FinalOutput)	This card determines whether Trace is turned on in mxconfig in input 2 (mxconfig). If Trace is turned on, this card returns a concatenation of Output 1 (Framework) plus Output 2 (GatherResults). If Trace is not turned on, it returns Output 2 (GatherResults).

Prior to running the MX validation framework, build the maps for Windows using the build\_deploy\_mx\_validation.bat file. The build\_deploy\_mx\_validation.bat file creates the deployment\_mx\_validation directory under the mx/maps directory.

Prior to running the MX validation frame, build the maps as described in [Building multiple maps using Deploy](#).

To run the MX Validation Framework, execute the swiftval map, overriding the first input with the MX data to validate.

Input Card 2 may be overridden with different mxconfig files in order to use different levels of validation for different message types (See [How to use different levels of validation in the MX validation framework](#)).

For most cases the other inputs should remain static for optimal performance.

## MX configuration file

The settings for MX validation are controlled by the mxconfig.xml file in the mx/maps directory.

The default content of the mxconfig.xml file is shown here:

```
<?xml version="1.0" encoding="UTF-8"?>
<Configuration>
 <XMLSchemaName></XMLSchemaName>
 <MapTrace>F</MapTrace>
 <ExtendedValidation>
 <BICValidation>T</BICValidation>
 <IBANValidation>T</IBANValidation>
 <CountryCodeValidation>T</CountryCodeValidation>
 <CurrencyCodeValidation>T</CurrencyCodeValidation>
 <DecimalPlacesValidation>T</DecimalPlacesValidation>
 </ExtendedValidation>
</Configuration>
```

- **Schema name**

The XMLSchemaName element can be used to provide the namespace information if the message data does not contain the schema namespace information.

- **Map trace**

This setting controls the optional debug log of the control map.

- **Extended validation settings**

Extended validation settings have a Boolean T/F value. The value T represents enabled, and the value F represents disabled.

## Schema name

The XMLSchemaName element can be used to provide the namespace information if the message data does not contain the schema namespace information.

An XMLSchemaName element is shown here:

```
<XMLSchemaName>acmt.008.001.01</XMLSchemaName>
```

This feature is not supported in IBM Sterling B2B Integrator

## Map trace

This setting controls the optional debug log of the control map.

Setting this element to T produces an audit log of all of the steps that the framework map records, and the values that the steps produce. Here is an example of the output from the trace:

```
<MapVersion>SWIFT MX Validation vn.n.n.n release</MapVersion>
<MapRunDate>20120531</MapRunDate>
<MapStartTime>15:05:59296</MapStartTime>
<config_SchemaName>ABSENT</config_SchemaName>
<xmllns_SchemaName>colr.003.001.02</xmllns_SchemaName>
<SchemaName> colr.003.001.02</SchemaName>
<MapName>ai004101_admi_004_001_01
<Command>-A -OBI -IH1 - 4.0.477</Command>
<Run>Error(-1), "XMLParser: Input XML data is invalid."
SAXParseException, Error [line: 10 column: 41] Datatype error:
Type:NumberFormatException, Message:Invalid chars encountered.
</Run>
<MapEndTime>15:05:59421</MapEndTime>
</Framework>Error(-1), "XMLParser: Input XML data is invalid."
SAXParseException, Error [line: 10 column: 41] Datatype
error:NumberFormatException, Message:Invalid Chars encountered
```

## Extended validation settings

Extended validation settings have a Boolean T/F value. The value T represents enabled, and the value F represents disabled.

The extended validation settings are listed here:

- The BICValidation setting validates the BIC against the list of BICs in the bic.xml file.
- The IBANValidation setting validates the IBAN for country code values that are found in the currencycode.xml file, and validates the check digit.
- The CountryCodeValidation setting validates all country codes that are found in the document against the currencycode.xml file.
- The CurrencyCodeValidation setting validates all currency codes that are found in the document against the currencycode.xml file.
- The DecimalPlacesValidation setting validates all of the currency amounts for the number of decimal places that are allowed for the specified currency. The currencycode.xml file is used for the validation of places. This validation is disabled if CountryCodeValidation is set to false.

## Individual MX validation maps

The MX validation maps validate for schema, BIC, IBAN, currency code, decimal places, and country codes for supported message types.

The map source for these maps is organized into map source files named according to their business function. For example, mx\_acmt\_validation is for all of the acmt messages. See [List of supported messages](#) for a full list of supported MX messages.

These maps all have the same five inputs and four outputs:

Input/output	Description
Input 1 (mx)	The MX data file compatible with the map input card definition. This card is overridden when this map is called from the swiftval framework map.
Input 2 (mxconfig)	The MX map configuration data. See <a href="#">MX configuration file</a> .
Input 3 (bicDirectory)	The BIC information file.
Input 4 (ccyCtryCodeList)	The currency code and decimal places information file.
Input 5 (errorMsgList)	The source file for all of the error messages.
Output 1 (SchemaValidation)	This card uses the GETXMLERRORMSG function to retrieve all of the XML schema errors in the XML document. This card uses the Sink adapter.
Output 2 (CurrencyAndAmountsList)	This card collects all of the elements from the data that needs to be validated. This card uses the Sink adapter.
Output 3 (validate)	This card uses the output from the CurrencyAndAmountsList output card to validate the data. It captures the errors through a series of rule checks to validate BIC, IBAN, country code, currency code, and decimal place values.
Output 4 (ErrorCollector)	This card collects all of the error information that is generated by the map. The SchemaValidation and the validate card outputs are captured in this card, and then sent as output from the map through the file adapter by default. When used as a <b>RUN</b> map, this output is passed back to the framework by use of the <b>ECHOIN</b> function.

You can run the individual MX validation maps using the framework map, or you can run the validation map that is in the relevant map source file to override the first input with MX data to validate. The other inputs remain static for optimal performance. The first four inputs for these maps are overridden when using the framework map in calling these extended validation maps.

Note: For the MX Header schema \$ahV10 only schema validation is possible, since it does not contain any elements that require extended validation

## List of supported messages

The supported MX messages are listed in the tables that are provided here.

- [Account Management list of supported messages](#)  
The table shows the supported MX messages for Account Management in the mx\_acmt\_validation executable map.
- [Cash Management list of supported messages](#)  
The table shows the supported MX messages for Cash Management in the mx\_camt\_validation executable map.
- [Collateral Management list of supported messages](#)  
The table shows the supported MX messages for Collateral Management in the mx\_colr\_validation executable map.
- [MX Application Headers list of supported messages](#)  
The table shows the supported MX messages for MX Application Headers in the mx\_head\_validation executable map.
- [Payments, Clearing and Settlement list of supported messages](#)  
The table shows the supported MX messages for Payments, Clearing and Settlement in the mx\_pacs\_validation executable map.
- [Payments Initiation list of supported messages](#)  
The table shows the supported MX messages for Payments Initiation in the mx\_pain\_validation executable map.
- [Reference Data list of supported messages](#)  
The table shows the supported MX messages for Reference Data in the mx\_reda\_validation executable map.
- [Securities Clearing list of supported messages](#)  
The table shows the MX messages for Securities Clearing in the mx\_secl\_validation executable map.
- [Securities Events list of supported messages](#)  
The table shows the supported MX messages for Securities Events in the mx\_seev\_validation executable map.
- [Securities Management list of supported messages](#)  
The table shows the supported MX messages for Securities Management in the mx\_semt\_validation executable map.
- [Securities Settlement list of supported messages](#)  
The table shows the MX messages for Securities Settlement in the mx\_sese\_validation executable map.
- [Securities Trading list of supported messages](#)  
The table shows the supported MX messages for Securities Trading in the mx\_setr\_validation executable map.
- [Trade Services Management list of supported messages](#)  
The table shows the supported MX messages for Trade Services Management in the mx\_tsmt\_validation executable map.

## Account Management list of supported messages

The table shows the supported MX messages for Account Management in the mx\_acmt\_validation executable map.

Type trees	Schema name	Validation map name
acmt_001_001_08	acmt.001.001.08.xsd	at001108_acmt_001_001_08
acmt_002_001_08	acmt.002.001.08.xsd	at002108_acmt_002_001_08
acmt_003_001_08	acmt.003.001.08.xsd	at003108_acmt_003_001_08
acmt_004_001_06	acmt.004.001.06.xsd	at004106_acmt_004_001_06

Type trees	Schema name	Validation map name
acmt_005_001_06	acmt.005.001.06.xsd	at005106_acmt_005_001_06
acmt_006_001_07	acmt.006.001.07.xsd	at006107_acmt_006_001_07
acmt_007_001_05	acmt.007.001.05.xsd	at007105_acmt_007_001_05
acmt_008_001_05	acmt.008.001.05.xsd	at008105_acmt_008_001_05
acmt_009_001_04	acmt.009.001.04.xsd	at009104_acmt_009_001_04
acmt_010_001_04	acmt.010.001.04.xsd	at010104_acmt_010_001_04
acmt_011_001_04	acmt.011.001.04.xsd	at011104_acmt_011_001_04
acmt_012_001_04	acmt.012.001.04.xsd	at012104_acmt_012_001_04
acmt_013_001_04	acmt.013.001.04.xsd	at013104_acmt_013_001_04
acmt_014_001_05	acmt.014.001.05.xsd	at014105_acmt_014_001_05
acmt_015_001_04	acmt.015.001.04.xsd	at015104_acmt_015_001_04
acmt_016_001_04	acmt.016.001.04.xsd	at016104_acmt_016_001_04
acmt_017_001_04	acmt.017.001.04.xsd	at017104_acmt_017_001_04
acmt_018_001_04	acmt.018.001.04.xsd	at018104_acmt_018_001_04
acmt_019_001_04	acmt.019.001.04.xsd	at019104_acmt_019_001_04
acmt_020_001_04	acmt.020.001.04.xsd	at020104_acmt_020_001_04
acmt_021_001_04	acmt.021.001.04.xsd	at021104_acmt_021_001_04

## Cash Management list of supported messages

The table shows the supported MX messages for Cash Management in the mx\_camt\_validation executable map.

Type trees	Schema name	Validation map name
camt_003_001_08	camt.003.001.08.xsd	ct003108_camt_003_001_08
camt_004_001_10	camt.004.001.10.xsd	ct004110_camt_004_001_10
camt_005_001_11	camt.005.001.11.xsd	ct005111_camt_005_001_11
camt_006_001_11	camt.006.001.11.xsd	ct006111_camt_006_001_11
camt_007_001_10	camt.007.001.10.xsd	ct007110_camt_007_001_10
camt_008_001_11	camt.008.001.11.xsd	ct008111_camt_008_001_11
camt_009_001_08	camt.009.001.08.xsd	ct009108_camt_009_001_08
camt_010_001_09	camt.010.001.09.xsd	ct010109_camt_010_001_09
camt_011_001_08	camt.011.001.08.xsd	ct011108_camt_011_001_08
camt_012_001_08	camt.012.001.08.xsd	ct012108_camt_012_001_08
camt_013_001_04	camt.013.001.04.xsd	ct013104_camt_013_001_04
camt_014_001_05	camt.014.001.05.xsd	ct014105_camt_014_001_05
camt_015_001_04	camt.015.001.04.xsd	ct015104_camt_015_001_04
camt_016_001_04	camt.016.001.04.xsd	ct016104_camt_016_001_04
camt_017_001_05	camt.017.001.05.xsd	ct017105_camt_017_001_05
camt_018_001_05	camt.018.001.05.xsd	ct018105_camt_018_001_05
camt_019_001_07	camt.019.001.07.xsd	ct019107_camt_019_001_07
camt_020_001_04	camt.020.001.04.xsd	ct020104_camt_020_001_04
camt_021_001_06	camt.021.001.06.xsd	ct021106_camt_021_001_06
camt_023_001_07	camt.023.001.07.xsd	ct023107_camt_023_001_07
camt_024_001_08	camt.024.001.08.xsd	ct024108_camt_024_001_08
camt_025_001_07	camt.025.001.07.xsd	ct025107_camt_025_001_07
camt_026_001_10	camt.026.001.10.xsd	ct026110_camt_026_001_10
camt_027_001_10	camt.027.001.10.xsd	ct027110_camt_027_001_10
camt_028_001_12	camt.028.001.12.xsd	ct028112_camt_028_001_12
camt_029_001_13	camt.029.001.13.xsd	ct029113_camt_029_001_13
camt_030_001_06	camt.030.001.06.xsd	ct030106_camt_030_001_06
camt_031_001_07	camt.031.001.07.xsd	ct031107_camt_031_001_07
camt_032_001_05	camt.032.001.05.xsd	ct032105_camt_032_001_05
camt_033_001_07	camt.033.001.07.xsd	ct033107_camt_033_001_07
camt_034_001_07	camt.034.001.07.xsd	ct034107_camt_034_001_07
camt_035_001_06	camt.035.001.06.xsd	ct035106_camt_035_001_06
camt_036_001_06	camt.036.001.06.xsd	ct036106_camt_036_001_06
camt_037_001_10	camt.037.001.10.xsd	ct037110_camt_037_001_10
camt_038_001_05	camt.038.001.05.xsd	ct038105_camt_038_001_05
camt_039_001_06	camt.039.001.06.xsd	ct039106_camt_039_001_06
camt_040_001_04	camt.040.001.04.xsd	ct040104_camt_040_001_04
camt_041_001_04	camt.041.001.04.xsd	ct041104_camt_041_001_04
camt_042_001_04	camt.042.001.04.xsd	ct042104_camt_042_001_04
camt_043_001_04	camt.043.001.04.xsd	ct043104_camt_043_001_04
camt_044_001_03	camt.044.001.03.xsd	ct044103_camt_044_001_03
camt_045_001_03	camt.045.001.03.xsd	ct045103_camt_045_001_03
camt_046_001_08	camt.046.001.08.xsd	ct046108_camt_046_001_08

camt_047_001_08	camt.047.001.08.xsd	ct047108_camt_047_001_08
camt_048_001_07	camt.048.001.07.xsd	ct048107_camt_048_001_07
camt_049_001_07	camt.049.001.07.xsd	ct049107_camt_049_001_07
camt_050_001_07	camt.050.001.07.xsd	ct050107_camt_050_001_07
camt_051_001_07	camt.051.001.07.xsd	ct051107_camt_051_001_07
camt_052_001_10	camt.052.001.10.xsd	ct052110_camt_052_001_10
camt_052_001_12	camt.052.001.12.xsd	ct052112_camt_052_001_12
camt_053_001_10	camt.053.001.10.xsd	ct053110_camt_053_001_10
camt_053_001_12	camt.053.001.12.xsd	ct053112_camt_053_001_12
camt_054_001_10	camt.054.001.10.xsd	ct054110_camt_054_001_10
camt_054_001_12	camt.054.001.12.xsd	ct054112_camt_054_001_12
camt_055_001_12	camt.055.001.12.xsd	ct055112_camt_055_001_12
camt_056_001_11	camt.056.001.11.xsd	ct056111_camt_056_001_11
camt_057_001_08	camt.057.001.08.xsd	ct057108_camt_057_001_08
camt_058_001_09	camt.058.001.09.xsd	ct058109_camt_058_001_09
camt_059_001_08	camt.059.001.08.xsd	ct059108_camt_059_001_08
camt_060_001_07	camt.060.001.07.xsd	ct060107_camt_060_001_07
camt_069_001_05	camt.069.001.05.xsd	ct069105_camt_069_001_05
camt_070_001_06	camt.070.001.06.xsd	ct070106_camt_070_001_06
camt_071_001_05	camt.071.001.05.xsd	ct071105_camt_071_001_05
camt_086_001_05	camt.086.001.05.xsd	ct086105_camt_086_001_05
camt_087_001_09	camt.087.001.09.xsd	ct087109_camt_087_001_09
camt_101_001_02	camt.101.001.02.xsd	ct101102_camt_101_001_02
camt_102_001_03	camt.102.001.03.xsd	ct102103_camt_102_001_03
camt_103_001_03	camt.103.001.03.xsd	ct103103_camt_103_001_03
camt_104_001_01	camt.104.001.01.xsd	ct104101_camt_104_001_01

## Collateral Management list of supported messages

The table shows the supported MX messages for Collateral Management in the mx\_colr\_validation executable map.

Type trees	Schema name	Validation map name
colr_003_001_05	colr.003.001.05.xsd	cr003105_colr_003_001_05
colr_004_001_05	colr.004.001.05.xsd	cr004105_colr_004_001_05
colr_005_001_06	colr.005.001.06.xsd	cr005106_colr_005_001_06
colr_006_001_05	colr.006.001.05.xsd	cr006105_colr_006_001_05
colr_007_001_06	colr.007.001.06.xsd	cr007106_colr_007_001_06
colr_008_001_06	colr.008.001.06.xsd	cr008106_colr_008_001_06
colr_009_001_05	colr.009.001.05.xsd	cr009105_colr_009_001_05
colr_010_001_05	colr.010.001.05.xsd	cr010105_colr_010_001_05
colr_011_001_05	colr.011.001.05.xsd	cr011105_colr_011_001_05
colr_012_001_05	colr.012.001.05.xsd	cr012105_colr_012_001_05
colr_013_001_05	colr.013.001.05.xsd	cr013105_colr_013_001_05
colr_014_001_05	colr.014.001.05.xsd	cr014105_colr_014_001_05
colr_015_001_05	colr.015.001.05.xsd	cr015105_colr_015_001_05
colr_016_001_05	colr.016.001.05.xsd	cr016105_colr_016_001_05
colr_019_001_01	colr.019.001.01.xsd	cr019101_colr_019_001_01
colr_020_001_01	colr.020.001.01.xsd	cr020101_colr_020_001_01
colr_021_001_01	colr.021.001.01.xsd	cr021101_colr_021_001_01
colr_022_001_01	colr.022.001.01.xsd	cr022101_colr_022_001_01
colr_023_001_01	colr.023.001.01.xsd	cr023101_colr_023_001_01
colr_024_001_01	colr.024.001.01.xsd	cr024101_colr_024_001_01

## MX Application Headers list of supported messages

The table shows the supported MX messages for MX Application Headers in the mx\_head\_validation executable map.

Type trees	Schema name	Validation map name
\$ahV10	\$ahV10.xsd	\$V0_\$ahV10
head_001_001_02	head_001_001_02.xsd	hd001102_head_001_001_02

## Payments, Clearing and Settlement list of supported messages

The table shows the supported MX messages for Payments, Clearing and Settlement in the mx\_pacs\_validation executable map.

Type trees	Schema name	Validation map name
pacs_002_001_14	pacs.002.001.14.xsd	ps002114_pacs_002_001_14
pacs_003_001_11	pacs.003.001.11.xsd	ps003111_pacs_003_001_11
pacs_004_001_13	pacs.004.001.13.xsd	ps004113_pacs_004_001_13
pacs_007_001_13	pacs.007.001.13.xsd	ps007113_pacs_007_001_13
pacs_008_001_10	pacs.008.001.10.xsd	ps008110_pacs_008_001_10
pacs_008_001_12	pacs.008.001.12.xsd	ps008112_pacs_008_001_12
pacs_009_001_11	pacs.009.001.11.xsd	ps009111_pacs_009_001_11
pacs_010_001_06	pacs.010.001.06.xsd	ps010106_pacs_010_001_06
pacs_028_001_06	pacs.028.001.06.xsd	ps028106_pacs_028_001_06

## Payments Initiation list of supported messages

The table shows the supported MX messages for Payments Initiation in the mx\_pain\_validation executable map.

Type trees	Schema name	Validation map name
pain_001_001_12	pain.001.001.12.xsd	pn001112_pain_001_001_12
pain_002_001_14	pain.002.001.14.xsd	pn002114_pain_002_001_14
pain_007_001_12	pain.007.001.12.xsd	pn007112_pain_007_001_12
pain_008_001_11	pain.008.001.11.xsd	pn008111_pain_008_001_11
pain_009_001_08	pain.009.001.08.xsd	pn009108_pain_009_001_08
pain_010_001_08	pain.010.001.08.xsd	pn010108_pain_010_001_08
pain_011_001_08	pain.011.001.08.xsd	pn011108_pain_011_001_08
pain_012_001_08	pain.012.001.08.xsd	pn012108_pain_012_001_08
pain_017_001_04	pain.017.001.04.xsd	pn017104_pain_017_001_04
pain_018_001_04	pain.018.001.04.xsd	pn018104_pain_018_001_04

## Reference Data list of supported messages

The table shows the supported MX messages for Reference Data the mx\_reda\_validation executable map.

Type trees	Schema name	Validation map name
reda_001_001_04	reda.001.001.04.xsd	ra001104_reda_001_001_04
reda_002_001_04	reda.002.001.04.xsd	ra002104_reda_002_001_04
reda_004_001_07	reda.004.001.07.xsd	ra004107_reda_004_001_07
reda_005_001_03	reda.005.001.03.xsd	ra005103_reda_005_001_03
reda_074_001_01	reda.074.001.01.xsd	ra074101_reda_074_001_01

## Securities Clearing list of supported messages

The table shows the MX messages for Securities Clearing in the mx\_secl\_validation executable map.

Type trees	Schema name	Validation map name
secl_001_001_03	secl.001.001.03.xsd	sl001103_secl_001_001_03
secl_002_001_03	secl.002.001.03.xsd	sl002103_secl_002_001_03
secl_003_001_03	secl.003.001.03.xsd	sl003103_secl_003_001_03
secl_004_001_03	secl.004.001.03.xsd	sl004103_secl_004_001_03
secl_005_001_02	secl.005.001.02.xsd	sl005102_secl_005_001_02
secl_006_001_02	secl.006.001.02.xsd	sl006102_secl_006_001_02
secl_007_001_03	secl.007.001.03.xsd	sl007103_secl_007_001_03
secl_008_001_03	secl.008.001.03.xsd	sl008103_secl_008_001_03
secl_009_001_03	secl.009.001.03.xsd	sl009103_secl_009_001_03
secl_010_001_03	secl.010.001.03.xsd	sl010103_secl_010_001_03

## Securities Events list of supported messages

The table shows the supported MX messages for Securities Events in the mx\_seev\_validation executable map.

Type trees	Schema name	Validation map name
seev_001_001_11	seev.001.001.11.xsd	sv001111_seev_001_001_11
seev_002_001_09	seev.002.001.09.xsd	sv002109_seev_002_001_09
seev_003_001_09	seev.003.001.09.xsd	sv003109_seev_003_001_09

seev_004_001_09	seev.004.001.09.xsd	sv004109_seev_004_001_09
seev_005_001_09	seev.005.001.09.xsd	sv005109_seev_005_001_09
seev_006_001_10	seev.006.001.10.xsd	sv006110_seev_006_001_10
seev_007_001_10	seev.007.001.10.xsd	sv007110_seev_007_001_10
seev_008_001_09	seev.008.001.09.xsd	sv008109_seev_008_001_09
seev_031_001_14	seev.031.001.14.xsd	sv031114_seev_031_001_14
seev_031_002_12	seev.031.002.12.xsd	sv031212_seev_031_002_12
seev_031_002_14	seev.031.002.14.xsd	sv031214_seev_031_002_14
seev_032_001_08	seev.032.001.08.xsd	sv032108_seev_032_001_08
seev_032_002_08	seev.032.002.08.xsd	sv032208_seev_032_002_08
seev_033_001_12	seev.033.001.12.xsd	sv033112_seev_033_001_12
seev_033_002_12	seev.033.002.12.xsd	sv033212_seev_033_002_12
seev_034_001_14	seev.034.001.14.xsd	sv034114_seev_034_001_14
seev_034_002_14	seev.034.002.14.xsd	sv034214_seev_034_002_14
seev_035_001_15	seev.035.001.15.xsd	sv035115_seev_035_001_15
seev_035_002_15	seev.035.002.15.xsd	sv035215_seev_035_002_15
seev_036_001_15	seev.036.001.15.xsd	sv036115_seev_036_001_15
seev_036_002_13	seev.036.002.13.xsd	sv036213_seev_036_002_13
seev_036_002_15	seev.036.002.15.xsd	sv036215_seev_036_002_15
seev_037_001_15	seev.037.001.15.xsd	sv037115_seev_037_001_15
seev_037_002_15	seev.037.002.15.xsd	sv037215_seev_037_002_15
seev_038_001_08	seev.038.001.08.xsd	sv038108_seev_038_001_08
seev_038_002_08	seev.038.002.08.xsd	sv038208_seev_038_002_08
seev_039_001_12	seev.039.001.12.xsd	sv039112_seev_039_001_12
seev_039_002_12	seev.039.002.12.xsd	sv039212_seev_039_002_12
seev_040_001_12	seev.040.001.12.xsd	sv040112_seev_040_001_12
seev_040_002_12	seev.040.002.12.xsd	sv040212_seev_040_002_12
seev_041_001_13	seev.041.001.13.xsd	sv041113_seev_041_001_13
seev_041_002_13	seev.041.002.13.xsd	sv041213_seev_041_002_13
seev_042_001_12	seev.042.001.12.xsd	sv042112_seev_042_001_12
seev_042_002_12	seev.042.002.12.xsd	sv042212_seev_042_002_12
seev_044_001_12	seev.044.001.12.xsd	sv044112_seev_044_001_12
seev_044_002_12	seev.044.002.12.xsd	sv044212_seev_044_002_12
seev_045_001_04	seev.045.001.04.xsd	sv045104_seev_045_001_04
seev_046_001_01	seev.046.001.01.xsd	sv046101_seev_046_001_01
seev_047_001_03	seev.047.001.03.xsd	sv047103_seev_047_001_03
seev_048_001_01	seev.048.001.01.xsd	sv048101_seev_048_001_01
seev_049_001_01	seev.049.001.01.xsd	sv049101_seev_049_001_01
seev_050_001_02	seev.050.001.02.xsd	sv050102_seev_050_001_02
seev_051_001_01	seev.051.001.01.xsd	sv051101_seev_051_001_01
seev_052_001_02	seev.052.001.02.xsd	sv052102_seev_052_001_02
seev_053_001_02	seev.053.001.02.xsd	sv053102_seev_053_001_02

## Securities Management list of supported messages

The table shows the supported MX messages for Securities Management the mx\_semt\_validation executable map.

Type trees	Schema name	Validation map name
semt_001_001_03	semt.001.001.03.xsd	st001103_semt_001_001_03
semt_001_001_04	semt.001.001.04.xsd	st001104_semt_001_001_04
semt_002_001_02	semt.002.001.02.xsd	st002102_semt_002_001_02
semt_002_001_11	semt.002.001.11.xsd	st002111_semt_002_001_11
semt_002_002_11	semt.002.002.11.xsd	st002211_semt_002_002_11
semt_003_001_02	semt.003.001.02.xsd	st003102_semt_003_001_02
semt_003_001_11	semt.003.001.11.xsd	st003111_semt_003_001_11
semt_003_002_11	semt.003.002.11.xsd	st003211_semt_003_002_11
semt_004_001_02	semt.004.001.02.xsd	st004102_semt_004_001_02
semt_005_001_02	semt.005.001.02.xsd	st005102_semt_005_001_02
semt_006_001_03	semt.006.001.03.xsd	st006103_semt_006_001_03
semt_007_001_03	semt.007.001.03.xsd	st007103_semt_007_001_03
semt_012_001_01	semt.012.001.01.xsd	st012101_semt_012_001_01
semt_013_001_06	semt.013.001.06.xsd	st013106_semt_013_001_06
semt_013_002_06	semt.013.002.06.xsd	st013206_semt_013_002_06
semt_014_001_07	semt.014.001.07.xsd	st014107_semt_014_001_07
semt_014_002_07	semt.014.002.07.xsd	st014207_semt_014_002_07
semt_015_001_09	semt.015.001.09.xsd	st015109_semt_015_001_09
semt_015_002_09	semt.015.002.09.xsd	st015209_semt_015_002_09

semt_016_001_09	semt.016.001.09.xsd	st016109_semt_016_001_09
semt_016_002_09	semt.016.002.09.xsd	st016209_semt_016_002_09
semt_017_001_12	semt.017.001.12.xsd	st017112_semt_017_001_12
semt_017_002_12	semt.017.002.12.xsd	st017212_semt_017_002_12
semt_018_001_13	semt.018.001.13.xsd	st018113_semt_018_001_13
semt_018_002_13	semt.018.002.13.xsd	st018213_semt_018_002_13
semt_019_001_10	semt.019.001.10.xsd	st019110_semt_019_001_10
semt_019_002_10	semt.019.002.10.xsd	st019210_semt_019_002_10
semt_020_001_07	semt.020.001.07.xsd	st020107_semt_020_001_07
semt_020_002_07	semt.020.002.07.xsd	st020207_semt_020_002_07
semt_021_001_08	semt.021.001.08.xsd	st021108_semt_021_001_08
semt_021_002_08	semt.021.002.08.xsd	st021208_semt_021_002_08
semt_022_001_05	semt.022.001.05.xsd	st022105_semt_022_001_05
semt_022_002_05	semt.022.002.05.xsd	st022205_semt_022_002_05
semt_023_001_02	semt.023.001.02.xsd	st023102_semt_023_001_02
semt_024_001_01	semt.024.001.01.xsd	st024101_semt_024_001_01
semt_041_001_02	semt.041.001.02.xsd	st041102_semt_041_001_02
semt_042_001_01	semt.042.001.01.xsd	st042101_semt_042_001_01

## Securities Settlement list of supported messages

The table shows the MX messages for Securities Settlement in the mx\_sese\_validation executable map.

Type trees	Schema name	Validation map name
sese_001_001_09	sese.001.001.09.xsd	se001109_sese_001_001_09
sese_002_001_09	sese.002.001.09.xsd	se002109_sese_002_001_09
sese_003_001_09	sese.003.001.09.xsd	se003109_sese_003_001_09
sese_004_001_09	sese.004.001.09.xsd	se004109_sese_004_001_09
sese_005_001_09	sese.005.001.09.xsd	se005109_sese_005_001_09
sese_006_001_09	sese.006.001.09.xsd	se006109_sese_006_001_09
sese_007_001_09	sese.007.001.09.xsd	se007109_sese_007_001_09
sese_008_001_09	sese.008.001.09.xsd	se008109_sese_008_001_09
sese_009_001_08	sese.009.001.08.xsd	se009108_sese_009_001_08
sese_010_001_07	sese.010.001.07.xsd	se010107_sese_010_001_07
sese_011_001_09	sese.011.001.09.xsd	se011109_sese_011_001_09
sese_012_001_11	sese.012.001.11.xsd	se012111_sese_012_001_11
sese_013_001_11	sese.013.001.11.xsd	se013111_sese_013_001_11
sese_014_001_09	sese.014.001.09.xsd	se014109_sese_014_001_09
sese_018_001_09	sese.018.001.09.xsd	se018109_sese_018_001_09
sese_019_001_08	sese.019.001.08.xsd	se019108_sese_019_001_08
sese_020_001_07	sese.020.001.07.xsd	se020107_sese_020_001_07
sese_020_002_07	sese.020.002.07.xsd	se020207_sese_020_002_07
sese_021_001_06	sese.021.001.06.xsd	se021106_sese_021_001_06
sese_021_002_06	sese.021.002.06.xsd	se021206_sese_021_002_06
sese_022_001_06	sese.022.001.06.xsd	se022106_sese_022_001_06
sese_022_002_06	sese.022.002.06.xsd	se022206_sese_022_002_06
sese_023_001_11	sese.023.001.11.xsd	se023111_sese_023_001_11
sese_023_002_11	sese.023.002.11.xsd	se023211_sese_023_002_11
sese_024_001_12	sese.024.001.12.xsd	se024112_sese_024_001_12
sese_024_002_12	sese.024.002.12.xsd	se024212_sese_024_002_12
sese_025_001_11	sese.025.001.11.xsd	se025111_sese_025_001_11
sese_025_002_11	sese.025.002.11.xsd	se025211_sese_025_002_11
sese_026_001_10	sese.026.001.10.xsd	se026110_sese_026_001_10
sese_026_002_10	sese.026.002.10.xsd	se026210_sese_026_002_10
sese_027_001_07	sese.027.001.07.xsd	se027107_sese_027_001_07
sese_027_002_07	sese.027.002.07.xsd	se027207_sese_027_002_07
sese_028_001_10	sese.028.001.10.xsd	se028110_sese_028_001_10
sese_028_002_10	sese.028.002.10.xsd	se028210_sese_028_002_10
sese_029_001_06	sese.029.001.06.xsd	se029106_sese_029_001_06
sese_029_002_06	sese.029.002.06.xsd	se029206_sese_029_002_06
sese_030_001_09	sese.030.001.09.xsd	se030109_sese_030_001_09
sese_030_002_09	sese.030.002.09.xsd	se030209_sese_030_002_09
sese_031_001_09	sese.031.001.09.xsd	se031109_sese_031_001_09
sese_031_002_09	sese.031.002.09.xsd	se031209_sese_031_002_09
sese_032_001_11	sese.032.001.11.xsd	se032111_sese_032_001_11
sese_032_002_11	sese.032.002.11.xsd	se032211_sese_032_002_11
sese_033_001_11	sese.033.001.11.xsd	se033111_sese_033_001_11

sese_033_002_11	sese.033.002.11.xsd	se033211_sese_033_002_11
sese_034_001_09	sese.034.001.09.xsd	se034109_sese_034_001_09
sese_034_002_09	sese.034.002.09.xsd	se034209_sese_034_002_09
sese_035_001_11	sese.035.001.11.xsd	se035111_sese_035_001_11
sese_035_002_11	sese.035.002.11.xsd	se035211_sese_035_002_11
sese_036_001_08	sese.036.001.08.xsd	se036108_sese_036_001_08
sese_036_002_08	sese.036.002.08.xsd	se036208_sese_036_002_08
sese_037_001_07	sese.037.001.07.xsd	se037107_sese_037_001_07
sese_037_002_07	sese.037.002.07.xsd	se037207_sese_037_002_07
sese_038_001_09	sese.038.001.09.xsd	se038109_sese_038_001_09
sese_038_002_09	sese.038.002.09.xsd	se038209_sese_038_002_09
sese_039_001_06	sese.039.001.06.xsd	se039106_sese_039_001_06
sese_039_002_06	sese.039.002.06.xsd	se039206_sese_039_002_06
sese_040_001_04	sese.040.001.04.xsd	se040104_sese_040_001_04
sese_040_002_04	sese.040.002.04.xsd	se040204_sese_040_002_04

## Securities Trading list of supported messages

The table shows the supported MX messages for Securities Trading in the mx\_setr\_validation executable map.

Type trees	Schema name	Validation map name
setr.001.001.04	setr.001.001.04.xsd	sr001104_setr_001_001_04
setr.002.001.04	setr.002.001.04.xsd	sr002104_setr_002_001_04
setr.003.001.04	setr.003.001.04.xsd	sr003104_setr_003_001_04
setr.004.001.04	setr.004.001.04.xsd	sr004104_setr_004_001_04
setr.004.002.01	setr.004.002.01.xsd	sr004201_setr_004_002_01
setr.005.001.04	setr.005.001.04.xsd	sr005104_setr_005_001_04
setr.006.001.05	setr.006.001.05.xsd	sr006105_setr_006_001_05
setr.006.002.01	setr.006.002.01.xsd	sr006201_setr_006_002_01
setr.007.001.04	setr.007.001.04.xsd	sr007104_setr_007_001_04
setr.008.001.04	setr.008.001.04.xsd	sr008104_setr_008_001_04
setr.009.001.04	setr.009.001.04.xsd	sr009104_setr_009_001_04
setr.010.001.04	setr.010.001.04.xsd	sr010104_setr_010_001_04
setr.010.002.01	setr.010.002.01.xsd	sr010201_setr_010_002_01
setr.011.001.04	setr.011.001.04.xsd	sr011104_setr_011_001_04
setr.012.001.05	setr.012.001.05.xsd	sr012105_setr_012_001_05
setr.012.002.01	setr.012.002.01.xsd	sr012201_setr_012_002_01
setr.013.001.04	setr.013.001.04.xsd	sr013104_setr_013_001_04
setr.014.001.04	setr.014.001.04.xsd	sr014104_setr_014_001_04
setr.015.001.04	setr.015.001.04.xsd	sr015104_setr_015_001_04
setr.016.001.04	setr.016.001.04.xsd	sr016104_setr_016_001_04
setr.016.002.01	setr.016.002.01.xsd	sr016201_setr_016_002_01
setr.017.001.04	setr.017.001.04.xsd	sr017104_setr_017_001_04
setr.018.001.04	setr.018.001.04.xsd	sr018104_setr_018_001_04
setr.027.001.04	setr.027.001.04.xsd	sr027104_setr_027_001_04
setr.029.001.02	setr.029.001.02.xsd	sr029102_setr_029_001_02
setr.030.001.02	setr.030.001.02.xsd	sr030102_setr_030_001_02
setr.044.001.03	setr.044.001.03.xsd	sr044103_setr_044_001_03
setr.047.001.02	setr.047.001.02.xsd	sr047102_setr_047_001_02
setr.049.001.02	setr.049.001.02.xsd	sr049102_setr_049_001_02
setr.051.001.02	setr.051.001.02.xsd	sr051102_setr_051_001_02
setr.053.001.02	setr.053.001.02.xsd	sr053102_setr_053_001_02
setr.055.001.02	setr.055.001.02.xsd	sr055102_setr_055_001_02
setr.057.001.02	setr.057.001.02.xsd	sr057102_setr_057_001_02
setr.058.001.02	setr.058.001.02.xsd	sr058102_setr_058_001_02
setr.059.001.01	setr.059.001.01.xsd	sr059101_setr_059_001_01
setr.060.001.01	setr.060.001.01.xsd	sr060101_setr_060_001_01
setr.061.001.01	setr.061.001.01.xsd	sr061101_setr_061_001_01
setr.062.001.01	setr.062.001.01.xsd	sr062101_setr_062_001_01
setr.064.001.01	setr.064.001.01.xsd	sr064101_setr_064_001_01
setr.065.001.01	setr.065.001.01.xsd	sr065101_setr_065_001_01
setr.066.001.01	setr.066.001.01.xsd	sr066101_setr_066_001_01

## Trade Services Management list of supported messages

The table shows the supported MX messages for Trade Services Management in the mx\_tsmt\_validation executable map.

Type trees	Schema name	Validation map name
tsmt.001.001.03	tsmt.001.001.03.xsd	tt001103_tsmt_001_001_03
tsmt.002.001.04	tsmt.002.001.04.xsd	tt002104_tsmt_002_001_04
tsmt.003.001.03	tsmt.003.001.03.xsd	tt003103_tsmt_003_001_03
tsmt.004.001.02	tsmt.004.001.02.xsd	tt004102_tsmt_004_001_02
tsmt.005.001.02	tsmt.005.001.02.xsd	tt005102_tsmt_005_001_02
tsmt.006.001.03	tsmt.006.001.03.xsd	tt006103_tsmt_006_001_03
tsmt.007.001.02	tsmt.007.001.02.xsd	tt007102_tsmt_007_001_02
tsmt.008.001.03	tsmt.008.001.03.xsd	tt008103_tsmt_008_001_03
tsmt.009.001.05	tsmt.009.001.05.xsd	tt009105_tsmt_009_001_05
tsmt.010.001.03	tsmt.010.001.03.xsd	tt010103_tsmt_010_001_03
tsmt.011.001.04	tsmt.011.001.04.xsd	tt011104_tsmt_011_001_04
tsmt.012.001.05	tsmt.012.001.05.xsd	tt012105_tsmt_012_001_05
tsmt.013.001.03	tsmt.013.001.03.xsd	tt013103_tsmt_013_001_03
tsmt.014.001.05	tsmt.014.001.05.xsd	tt014105_tsmt_014_001_05
tsmt.015.001.03	tsmt.015.001.03.xsd	tt015103_tsmt_015_001_03
tsmt.016.001.03	tsmt.016.001.03.xsd	tt016103_tsmt_016_001_03
tsmt.017.001.05	tsmt.017.001.05.xsd	tt017105_tsmt_017_001_05
tsmt.018.001.05	tsmt.018.001.05.xsd	tt018105_tsmt_018_001_05
tsmt.019.001.05	tsmt.019.001.05.xsd	tt019105_tsmt_019_001_05
tsmt.020.001.02	tsmt.020.001.02.xsd	tt020102_tsmt_020_001_02
tsmt.021.001.03	tsmt.021.001.03.xsd	tt021103_tsmt_021_001_03
tsmt.022.001.02	tsmt.022.001.02.xsd	tt022102_tsmt_022_001_02
tsmt.023.001.03	tsmt.023.001.03.xsd	tt023103_tsmt_023_001_03
tsmt.024.001.03	tsmt.024.001.03.xsd	tt024103_tsmt_024_001_03
tsmt.025.001.03	tsmt.025.001.03.xsd	tt025103_tsmt_025_001_03
tsmt.026.001.02	tsmt.026.001.02.xsd	tt026102_tsmt_026_001_02
tsmt.027.001.02	tsmt.027.001.02.xsd	tt027102_tsmt_027_001_02
tsmt.028.001.03	tsmt.028.001.03.xsd	tt028103_tsmt_028_001_03
tsmt.029.001.02	tsmt.029.001.02.xsd	tt029102_tsmt_029_001_02
tsmt.030.001.03	tsmt.030.001.03.xsd	tt030103_tsmt_030_001_03
tsmt.031.001.03	tsmt.031.001.03.xsd	tt031103_tsmt_031_001_03
tsmt.032.001.03	tsmt.032.001.03.xsd	tt032103_tsmt_032_001_03
tsmt.033.001.03	tsmt.033.001.03.xsd	tt033103_tsmt_033_001_03
tsmt.034.001.03	tsmt.034.001.03.xsd	tt034103_tsmt_034_001_03
tsmt.035.001.03	tsmt.035.001.03.xsd	tt035103_tsmt_035_001_03
tsmt.036.001.03	tsmt.036.001.03.xsd	tt036103_tsmt_036_001_03
tsmt.037.001.03	tsmt.037.001.03.xsd	tt037103_tsmt_037_001_03
tsmt.038.001.03	tsmt.038.001.03.xsd	tt038103_tsmt_038_001_03
tsmt.040.001.03	tsmt.040.001.03.xsd	tt040103_tsmt_040_001_03
tsmt.041.001.03	tsmt.041.001.03.xsd	tt041103_tsmt_041_001_03
tsmt.042.001.03	tsmt.042.001.03.xsd	tt042103_tsmt_042_001_03
tsmt.044.001.02	tsmt.044.001.02.xsd	tt044102_tsmt_044_001_02
tsmt.045.001.02	tsmt.045.001.02.xsd	tt045102_tsmt_045_001_02
tsmt.046.001.01	tsmt.046.001.01.xsd	tt046101_tsmt_046_001_01
tsmt.047.001.01	tsmt.047.001.01.xsd	tt047101_tsmt_047_001_01
tsmt.048.001.01	tsmt.048.001.01.xsd	tt048101_tsmt_048_001_01
tsmt.049.001.01	tsmt.049.001.01.xsd	tt049101_tsmt_049_001_01
tsmt.050.001.01	tsmt.050.001.01.xsd	tt050101_tsmt_050_001_01
tsmt.051.001.01	tsmt.051.001.01.xsd	tt051101_tsmt_051_001_01
tsmt.052.001.01	tsmt.052.001.01.xsd	tt052101_tsmt_052_001_01

## Error reporting in the MX validation framework

If errors are detected in the MX message, an error report is generated.

Two types of error messages are generated from the MX validation framework map, a schema validation error, and an extended validation error.

- A schema validation error indicates that the MX instance data does not match the schema.
- An extended validation error indicates that the MX instance data contains an invalid BIC, Country, Currency, IBAN, or incorrect decimal places.

## Schema validation error message

The error report contains the SAX parsing error message, or messages. An example of a schema validation error message is shown here:

```
Error (-1), "XMLParser: Input XML data is invalid." SAXParseException, Error [line: 24 column: 23] Datatype error: Type:InvalidDatatypeValueException, Message:Value
```

```
'99BADBIC' does not match regular expression facet '[A-Z]{6,6}[A-Z2-9][A-NP-ZO-9]([A-Z0-9]{3,3}){0,1}'.
```

## Extended validation error message

---

The extended validation error report contains an extended error message with the error code, text that explains the error, and an offset location of the error.

An example of an extended error message is shown here:

```
E80001 - UX - Invalid country code found at position 7362
```

This error shows that the value UX is not a valid country code, and the position of UX can be found on the 7,362nd position from the beginning of the file.

The resource map that is used for the extended error messages is called errorcodes.txt, and can be found in the mx/maps directory. These error codes are externalized so that you can easily update them for translation, or for wording changes. Translate only the text of the message and not the error code. Additional error messages can be added, if they follow the error code format and they are listed in ascending order by error message number.

---

## Adding maps to the MX validation framework

---

Additional XML messages can be added to the framework to perform XML schema validation and optionally your own extended validation rules.

Maps added to the framework must be consistent with existing validation maps in that they must have the same input and output cards, and map naming conventions, as described in [Individual MX validation maps](#). Perform these steps to add more messages:

1. Create additional validation maps. Copy an existing validation map and edit it according to your needs. Rename the map to specify the message type, bearing in mind the rules governing validation map names. See [Individual MX validation maps](#).
  2. Modify the input card in this map to use either the native XML schema, or create a Xerces type tree from the schema using the Transformation Extender Map Designer schema importer utility. The resulting type tree can be used as the input card source.
  3. Add the **RESTART** attribute to the Document Element, which is at the top level of the schema or the type tree. This ensures that all of the errors are captured.
  4. Use the existing map rules as a template to update the rules in Output Card 2, to collect all the elements that you want to validate, such as BICs and IBANs.
- 

## How to use different levels of validation in the MX validation framework

---

You can perform different levels of validation for different message types.

For example, you might not want to perform BIC validation for the camt messages, but you do want to perform this for the pacs messages. In this situation, you would use two separate mxconfig.xml files:

mxconfig_camt.xml	BIC validation=F
mxconfig_pacs.xml	BIC validation=T

How you use these files depends on whether you are using the framework map, or are running the individual MX validation maps independently of the framework. The examples here use the Command Server as the ITX map execution platform.

This example shows how to use the files when you use the framework map:

```
dtxcmdsv swiftval -IF1 camt.053.001.04.xml IF2 mxconfig_camt.xml
```

```
dtxcmdsv swiftval -IF1 pacs.008.001.04.xml IF2 mxconfig_pacs.xml
```

This example shows how to use the files when you use the individual MX validation maps:

```
dtxcmdsv ct053104_camt_053_001_04 -IF1 camt.053.001.04.xml IF2 mxconfig_camt.xml
```

```
dtxcmdsv ps008104_pacs_008_001_04 -IF1 pacs.008.001.04.xml IF2 mxconfig_pacs.xml
```

## Debugging the MX validation framework

---

The MX framework map is designed to record the steps it performs in parsing the data to determine the correct **RUN** map to use.

It captures the metadata information obtained from the configuration input and the message data. The map can, optionally, output all of this information for debugging purposes. To enable this feature see the [MX configuration file](#).

Error messages alert you that something went wrong in the process:

## E09005 - Input validation failure from map ct052102\_camt\_052\_001\_02

---

The E09005 message alerts you that the RUN map being called was unsuccessful in parsing the data and so did not produce any output. In order to determine what is wrong in this case, run the data directly through the map ct052102\_camt\_052\_001\_02 with the Transformation Extender Trace setting enabled.

## E09003 The WTX map failed due to invalid input card

---

The E09003 message usually occurs if the schema for the message data is not in the correct location, or if the message data is missing. Ensure that the schemas are deployed to the correct location. The default location the maps use is ..schemas.

## E09006 The XML schema name: badname.001.001.01 used as input to the map in the configuration file was not able to be identified.

The map as shipped supports a set of messages and specific versions. If data is passed to the framework, and this data is missing the schema name, or if the validation map name cannot be determined, then this message will be triggered. Ensure that the data being checked has the schema name, and that the schema is in place for the data to use. If the schema is not included, it can be added. See [Adding maps to the MX validation framework](#).

## Building and deploying SWIFT map systems

Batch scripts are included in the installation for the build and deployment of the SWIFT component MT and MX systems.

Additionally, ITX based systems (msd files) are also included for MX systems. These scripts make it simple to build all of the required maps, copy support files, and optionally FTP all of the files to their runtime destination directories.

- [Build Script Files](#)  
The location of the build script files, as well as a description of each, are included here.
- [Script Process Description](#)  
The documentation provided here describes script parameters, as well as the activities that the scripts perform.
- [Running the build and deployment scripts](#)  
Four individual systems can be built and deployed through use of the batch scripts.
- [How to enable SFTP to deploy the pack](#)  
By default, FTP is used to deploy the files to the target system.

## Build Script Files

The location of the build script files, as well as a description of each, are included here.

Script file name	Directory	Description
build_deploy_jvc.bat	../mt/jvc/maps	This script builds the SWIFT Java validation utility maps. Optionally, by use of FTP or SFTP, the script can deploy the maps, metadata, and the Java jar files to the target system.
build_deploy_lmf.bat	../mt/lmf	This script builds the LMF transformation maps. Optionally, by use of either FTP or SFTP, the script can deploy the maps.
build_deploy_mx_validation.bat	../mx/maps	This script builds the SWIFT MX validation maps, and copies all of the necessary support files. Optionally, by use of either FTP or SFTP, the script can deploy all of the maps and support files to the target system.

## Script Process Description

The documentation provided here describes script parameters, as well as the activities that the scripts perform.

- [Script parameters](#)  
The parameters that are listed here are used for all of the build and deploy scripts in the IBM Transformation Extender Pack for Financial Payments.
- [Actions performed by the build and deploy scripts](#)  
The build and deployment scripts perform the following actions, which are based on the parameters that are passed when they are invoked.

## Script parameters

The parameters that are listed here are used for all of the build and deploy scripts in the IBM Transformation Extender Pack for Financial Payments.

1. Mandatory - Operating system for map compile. This first parameter is always required. The parameter sets the operating system that the map step uses during compile.
2. TX installation directory.
3. FTP username - If no username is given, it is assumed that the deployment step is not required, and parameters 3 - 5 are not needed.
4. FTP user's password.
5. FTP server location - The FTP server location can be either a DHCP name, or the physical IP address.
6. Top-level target directory - The top-level target directory to deploy map files, schemas, and supporting files. This file must not contain spaces and must be the Transformation Extender home directory that contains the libs directory.

## Actions performed by the build and deploy scripts

The build and deployment scripts perform the following actions, which are based on the parameters that are passed when they are invoked.

1. The first parameter identifies the map compiler platform that the script is to use.
2. TX installation directory identifies which TX version will be used to compile the maps.
3. If the FTP username parameter is passed to the script, then the remaining parameters must also be passed (FTP password, FTP server, and the Transformation Extender target directory on the remote server).

4. A local deploy directory structure is created after parameter validation, if the directory structure does not already exist. This directory is created so that all of the files that need to be deployed are in a single directory.

The structure of this directory resembles the pack directory structure. The directory is created under the directory from which the batch file is running.

These corresponding directories are created:

- The build\_deploy\_jvc.bat file creates the directory deployment\_jvc under the mt/jvc/maps directory.
- The build\_deploy\_lmf.bat file creates the directory deployment\_lmf under the mt/lmf/maps directory.
- The build\_deploy\_mx\_validation.bat file creates the directory deployment\_mx\_validation under the mx/maps directory.

5. All maps for the system are compiled into the deploy directory for the operating system and are then passed to the script as the first parameter.

6. Any supporting files are then copied to the deploy directory.

7. FTP and SFTP deployment scripts are created from a template file (a file with an .scp extension). This file is in the same directory as the batch script at execution time. For security purposes, these scripts are deleted after the deployment completes.

Note: If you must keep these FTP scripts for debugging purposes, comment out the line 'del deploy.scp' in the batch script.

8. All of the files in the local deployment directory are transferred through use of FTP by default. They can, however, be modified by using SFTP.

9. When script execution is complete, a log file named *build script name.log* is available in the same directory as the script. The log file lists all of the actions that are performed by the script, and whether the action was successful

---

## Running the build and deployment scripts

Four individual systems can be built and deployed through use of the batch scripts.

See [Build Script Files](#) for details about these scripts.

To run the script, follow these steps:

1. Open a command prompt, and go to the directory that contains the script.

2. Call the script by passing in command-line parameters. The following are the available options: All supported OS by TX version

3. TX installation directory.

4. If the files must be copied to a remote system, add these parameters:

- a. FTP username - If a username is not given, it is assumed that the deployment step is not required. Parameters b - d are not needed.
- b. FTP user's password.
- c. FTP server location. This location can be identified as either a DHCP name or the physical IP address.
- d. Top-level target directory to which map files, schemas, and supporting files are deployed.

Note: The script is designed to use paths without spaces for parameter **d**. Ensure that your destination path does not contain spaces.

Note: For parameter **d**, under the top-level directory that is specified, the files are deployed to a subfolder called mt, or mx, depending on the deployment type, and distributed to other folders under this folder.

As an example, a call to a script in which all parameters pass is indicated by the following message:

```
<build script name>.bat AIX <TX install dir> user password UNIXMACHINE
/usr/apps/websphere_transformation_extender
```

---

## How to enable SFTP to deploy the pack

By default, FTP is used to deploy the files to the target system.

If your system uses SFTP, then the commented line in the following script can be edited:

```
REM psftp %USER%@%FTP_SERVER% -pw %PASSWORD% -bc -be -b deploy.scp >> %LOG%
```

The following is the default FTP line after it is commented to enable SFTP deployment:

```
psftp %USER%@%FTP_SERVER% -pw %PASSWORD% -bc -be -b deploy.scp >> %LOG% REM ftp -s:deploy.scp >> %LOG%
```

The default SFTP process uses the command from PuTTY. Modify the psftp line if you use other SFTP software.

---

## Implementing SWIFT component in IBM Sterling B2B Integrator (Deprecated)

The following are required in order to use the Pack for Financial Payments, SWIFT component with IBM Sterling B2B Integrator:

- IBm Transformation Extender Integration Server
- Transformation Extender Pack for Financial Payments
- IBM Sterling B2B Integrator

This documentation assumes familiarity with IBM Sterling B2B Integrator concepts and terminology.

IBM Sterling B2B Integrator is a transaction engine that runs business processes that you define, and manages them according to your business requirements. It supports high-volume electronic message exchange, complex routing, translation, and flexible interaction with multiple internal systems and external business partners. Using the IBm Transformation Extender Integration Server with IBM Sterling B2B Integrator, business process can derive benefit from the Transformation Extender transformation capabilities.

The following IBM Sterling B2B Integrator services are examples of services that provide Transformation Extender integration capabilities:

- Generic Enveloping Service
- Generic Deenveloping Service

Both of the services can be configured to allow you to choose:

- SWIFT component, JVC compliance checking for business process handling
- SWIFT component, MX validation for business process handling MX messages

Transformation Extender maps can run within the IBM Sterling B2B Integrator services on some, but not all, versions of the base product. See the system requirements for those versions of the base product that support IBM Sterling B2B Integrator.

- [Deploying the XML schemas to IBM Sterling B2B Integrator \(Deprecated\)](#)

The SWIFT component of the Pack for Financial Payments contains SWIFT MX schemas and other pack-specific XML schemas that are used for configuration, transformation, and validation.

- [Using JVC compliance checking on IBM Sterling B2B Integrator \(Deprecated\)](#)

The documentation provided here describes how to enable and deploy the SWIFT component of the Pack for Financial Payments compliance checking to IBM Sterling B2B Integrator.

- [Using MX validation for MX compliance checking on IBM Sterling B2B Integrator \(Deprecated\)](#)

The following procedures describe how to enable the Pack for Financial Payments compliance checking for SWIFT MX messages, and how to deploy its SWIFT MX compliance component to the IBM Sterling B2B Integrator server.

---

## Deploying the XML schemas to IBM Sterling B2B Integrator (Deprecated)

The SWIFT component of the Pack for Financial Payments contains SWIFT MX schemas and other pack-specific XML schemas that are used for configuration, transformation, and validation.

For these XML schemas to be used within the IBM Sterling B2B Integrator, they must be uploaded to the IBM Sterling B2B Integrator schema repository. The steps provided here describe how to check-in (upload) the XML schemas that come with the pack to the IBM Sterling B2B Integrator schema repository.

The Transformation Extender base product resolves the location of the XML schemas used within an Transformation Extender map that is executing within the IBM Sterling B2B Integrator in the following order:

1. The XML schema is resolved by searching the SI schema repository.
2. If the schema is not found in the SI repository, the Transformation Extender locates the schema based on any file path references associated with it.

To deploy XML schemas to the IBM Sterling B2B Integrator server:

1. Stop IBM Sterling B2B Integrator.
2. Navigate to `<si_install_dir>/bin`.
3. Load the XML schemas to the IBM Sterling B2B Integrator directory.

a. For Windows, enter the following at the command prompt or in the Run dialog box:

```
InstallService.cmd product short name/common/data/swiftnnnnxs4si.jar
```

Note: To obtain the product short name, open a command prompt, change the directory to C:/IBM, then enter dir /x. The product short name is displayed. For example, the product short name for IBM Transformation Extender version 8.4.1 is WEBSPH~1.1.

b. For Linux® and UNIX platforms, enter:

```
installService.sh <pack_install_dir>/common/data/swiftnnnnxs4si.jar
```

4. Restart IBM Sterling B2B Integrator.

In the directory locations, *product short name* indicates the short name of the installation directory; *<install\_dir>* indicates the installation location of the Pack for Financial Payments, and *n.n.n.n* indicates the version number of the Pack for Financial Payments.

---

## Using JVC compliance checking on IBM Sterling B2B Integrator (Deprecated)

The documentation provided here describes how to enable and deploy the SWIFT component of the Pack for Financial Payments compliance checking to IBM Sterling B2B Integrator.

- [Enabling JVC compliance checking on IBM Sterling B2B Integrator](#)

These are the steps required to enable the JVC compliance checking for the Generic Envelope and Generic Deenvelope services, using the IBM Sterling B2B Integrator Dashboard.

- [Deploying JVC compliance checking on IBM Sterling B2B Integrator](#)

You can deploy JVC compliance checking on IBM Sterling B2B Integrator.

---

## Enabling JVC compliance checking on IBM Sterling B2B Integrator

These are the steps required to enable the JVC compliance checking for the Generic Envelope and Generic Deenvelope services, using the IBM Sterling B2B Integrator Dashboard.

### About this task

If a Transformation Extender map is specified in the SWIFT envelope, the specified map is used only for translation, and compliance checking is handled separately.

### Procedure

1. Open the IBM Sterling B2B Integrator Dashboard.
2. On the SWIFT envelope, set the attribute Compliance Check message to **Yes**.
3. Set the attribute Use Transformation Extender compliance checking to **Yes**.
4. Set the WebSphere Transformation Extender Compliance Checking Options based upon your preferences.

- Validate BIC addresses.

Note: Validation for BIC, currency, and country codes is made against the IBM Sterling B2B Integrator SWIFT code lists. If the default SWIFT code lists versions have zero entries, the validation defaults against the following repository files that come with the SWIFT component: bic.xml and currencycode.xml

## Deploying JVC compliance checking on IBM Sterling B2B Integrator

You can deploy JVC compliance checking on IBM Sterling B2B Integrator.

### About this task

If the SWIFT component of the Transformation Extender Pack for Financial Payments is installed on the IBM Sterling B2B Integrator server, skip step 1.

If the Transformation Extender Pack for Financial Payments is not installed on the IBM Sterling B2B Integrator server, then you must deploy the components to the IBM Sterling B2B Integrator server using the procedures provided here.

In the following procedures, *<si\_install\_dir>* is the location where the IBM Sterling B2B Integrator server is installed.

In the following procedures, make certain that the permissions are set properly on the pack directory.

### Procedure

1. Deploy the pack to the IBM Sterling B2B Integrator server. See the pack release notes for deployment instructions.
2. After installing the Transformation Extender Pack for Financial Payments on Windows, copy jvalccyy.jar from the following location:  
`<install_dir>/packs/financial_payments_vn.n.n.n/UIProjectImports/swiftMTjvcConfig.tar.gz`  
to the following directory on the IBM Sterling B2B Integrator server:  
`<si_install_dir>/jar/SterlingTX/n_n_n`  
where *n\_n\_n* indicates the IBM Transformation Extender base product version.  
Note: The JVC must be set to verbose mode.
3. Install the jvalccyy.jar file by using the IBM Sterling B2B Integrator third party installer utility called install3rdParty under *<si\_install\_dir>/bin*  
For example:  
`<si_install_dir>/bin/install3rdParty.cmd<SP>SterlingTX<SP>n_n_n<SP>-j<SP><install_dir>/extjar/jvalccyy.jar -nodeploy`  
Where, <SP> is Space.  
Note: Make sure to depict appropriate spacing in the command.  
The above command is contained in one line.
4. Define customer\_overrides.properties under the following directory:  
`<si_install_dir>/properties`  
Include the following entry:  
`translator.SWIFTValidation=com.ibm.websphere.dtx.ip.swift.srgccyy.Validator`  
`translator.DTX_SWIFT_CONFIG_DIR=<install_dir>/packs/financial_payments_vn.n.n.n/swift/mt/jvc/maps`
5. Restart IBM Sterling B2B Integrator. This is required after changes are made to the customer\_overrides.properties file.
6. Verify the configuration by executing the MT Generic Envelope and Deenvelope example maps. The result should be consistent with the expected results as defined in [Expected results](#).

## Using MX validation for MX compliance checking on IBM Sterling B2B Integrator (Deprecated)

The following procedures describe how to enable the Pack for Financial Payments compliance checking for SWIFT MX messages, and how to deploy its SWIFT MX compliance component to the IBM Sterling B2B Integrator server.

- [Enabling MX compliance checking on IBM Sterling B2B Integrator](#)  
This documentation describes how to enable MX validation for the Generic Enveloping and Generic Deenveloping services.
- [Deploying MX compliance checking components](#)

## Enabling MX compliance checking on IBM Sterling B2B Integrator

This documentation describes how to enable MX validation for the Generic Enveloping and Generic Deenveloping services.

### About this task

If a Transformation Extender map is specified in the SWIFT envelope, the specified map is used only for translation. Compliance checking is handled separately.

### Procedure

1. On the SWIFT envelope, set the attribute Compliance Check message to Yes.
2. Set the attribute Use Transformation Extender compliance checking to Yes.
3. Set the Transformation Extender compliance checking options based upon your preferences:
  - Validate BIC addresses
  - Validate IBAN data

- Validate currency codes
- Validate amount and the number of decimal digits
- Validate country codes

Note: Validation for BIC, currency, and country codes are made against the Sterling SWIFT code lists. If the default SWIFT code list versions have zero entries, the validation defaults against the following repository files that come with the SWIFT component of the Pack for Financial Payments: bic.xml and currencycodedecimals.xml.

## Deploying MX compliance checking components

### About this task

If the SWIFT component of the Pack for Financial Payments is installed on the IBM Sterling B2B Integrator server, then you can skip step 1 in the following procedure, and you can continue with step 2.

If the SWIFT component of the Pack for Financial Payments is not installed on the IBM Sterling B2B Integrator server, then you must deploy the components to the IBM Sterling B2B Integrator server, using the following procedures:

In the following procedures, and throughout this documentation, `<si_install_dir>`, is the location where the IBM Sterling B2B Integrator is installed.

Note: Make certain that the permissions are set properly on the pack directory structure and subsequent sub-directories, including all of the compiled maps.

### Procedure

1. Deploy the Pack for Financial Payments to the IBM Sterling B2B Integrator server.
2. After installing the SWIFT component of the Pack for Financial Payments on Windows, define the `customer_overrides.properties` under the following directory:  
`<si_install_dir>/properties`  
Include the following entry:  
  
`translator.framework_driver_map_location=<install_dir>/financial_payments_vn.n.n.n/swift/mx/maps/swiftval.mmc`
3. Restart IBM Sterling B2B Integrator. This is required after changes are made to the `customer_overrides.properties` file.
4. Verify the configuration by executing the MX Generic Envelope and Deenvelope example maps. The result should be consistent with the expected results as defined in [Expected results](#).

## SWIFT component examples

- [Bank to XML example](#)
- [SWIFT to LMF example](#)
- [SWIFT from LMF example](#)
- [SWIFT MT Compliance example](#)
- [MT JVC Report example](#)  
This example takes a file in SWIFT Alliance Access (SAA) format containing a batch of SWIFT messages and uses the JVC to create a validation report.
- [MT950 Splitter Map example](#)  
The MT950 Splitter Map example contains 2 maps that demonstrate techniques for splitting a 5k MT950 Statement message into smaller MT950 messages.
- [MT Launcher Validation framework](#)  
This sample validation framework uses the Transformation Extender Launcher and the JVC to perform trigger-based MT validation.
- [MT - MX translation maps example \(Deprecated\)](#)  
The MT - MX translation maps example shows how the SWIFT component can be used to translate between MT and MX messages.
- [SWIFT MX Message Bundle and Unbundle map example using Design Server](#)  
This example demonstrates the use of a map that splits the MX message Business Application Header (BAH) from the MX message data, and also packages the BAH with the message data.
- [SWIFT MX validation example](#)  
This example demonstrates SWIFT MX validation.
- [IBM Sterling B2B Integrator examples \(Deprecated\)](#)  
The documentation provided here describes how to use the IBM Sterling B2B Integrator examples.
- [Universal Confirmation examples](#)

## Bank to XML example

- [How to run the example in Design Studio](#)  
Run instructions for the Bank to XML example.
- [How to run the example in Design Server](#)  
Run instructions for the Bank to XML example.

## How to run the example in Design Studio

Run instructions for the Bank to XML example.

## About this task

---

These steps describe how to run the Bank to XML example in Design Studio.

## Procedure

---

1. Use the Design Studio to open the following map source file: bankfiletoxml.mms.
2. Build all the maps in the source file, and then run the following transformation maps:
  - bictoxml
  - currencytoxml

## How to run the example in Design Server

---

Run instructions for the Bank to XML example.

## About this task

---

These steps describe how to run the Bank to XML example in Design Server.

## Procedure

---

1. Import the swiftBankFileXML.zip file into the Design Server UI. See the release notes for import instructions.
2. Create a package for the project and build all the maps.
3. Open the swiftBankFileXML project in the Design Server UI.
4. Run the following maps:
  - currencytoxml (maps/bankfiletoxml)
  - bictoxml (maps/bankfiletoxml)
5. View the output. Go to the map Diagram-Result, and hover the cursor over Target, then select View Data.
6. To download the output file, refer to [How to download output files](#).
  - [How to download output files](#)  
After you build/run maps, download the output files.

## How to download output files

---

After you build/run maps, download the output files.

## About this task

---

To download the output file, go to **Files** in the Design Server UI and download the desired file using these steps:

## Procedure

---

1. Click the output Card Settings that you want to download.
2. From the Command text box, copy the name of the file that you want to download. Make certain that you copy only the file name.  
For example, if the Command text box displays:  
  
`data/ach_contested_dishonored_return.out`  
copy only:  
  
`ach_contested_dishonored_return.out`
3. From Files, click the New icon to open the New File dialog box.
4. Paste the file name in the Name field.
5. Click the Folder field drop down and select data.
6. Click Ok.
7. Build and re-run the map.
8. Go to Files and select Download to view the data.

## SWIFT to LMF example

---

- [How to run the example in Design Studio](#)  
Run instructions for the SWIFT to LMF example.
- [How to run the example in Design Server](#)  
Run instructions for the SWIFT to LMF example.

# How to run the example in Design Studio

Run instructions for the SWIFT to LMF example.

## About this task

These steps describe how to run the SWIFT to LMF example in Design Studio.

## Procedure

1. Use the Design Studio to open the following map source files:

- swift\_mt101\_tolmf.mms
- swift\_mt102\_tolmf.mms
- swift\_mt103\_tolmf.mms
- swift\_mt202\_COV\_tolmf.mms
- swift\_mt202\_tolmf.mms
- swift\_mt205\_COV\_tolmf.mms
- swift\_mt205\_tolmf.mms
- swift\_mt210\_tolmf.mms
- swift\_mt300\_tolmf.mms
- swift\_mt304\_tolmf.mms
- swift\_mt305\_tolmf.mms
- swift\_mt320\_tolmf.mms
- swift\_mt502\_tolmf.mms
- swift\_mt509\_tolmf.mms
- swift\_mt515\_tolmf.mms
- swift\_mt535\_tolmf.mms
- swift\_mt536\_tolmf.mms
- swift\_mt537\_tolmf.mms
- swift\_mt548\_tolmf.mms
- swift\_mt549\_tolmf.mms
- swift\_mt564\_tolmf.mms
- swift\_mt565\_tolmf.mms
- swift\_mt566\_tolmf.mms
- swift\_mt567\_tolmf.mms
- swift\_mt568\_tolmf.mms
- swift\_mt578\_tolmf.mms
- swift\_mt900\_tolmf.mms
- swift\_mt910\_tolmf.mms
- swift\_mt940\_tolmf.mms
- swift\_mt942\_tolmf.mms
- swift\_mt950\_tolmf.mms
- swift\_mtn90\_tolmf.mms
- swift\_mtn91\_tolmf.mms
- swift\_mtn92\_tolmf.mms
- swift\_mtn95\_tolmf.mms
- swift\_mtn96\_tolmf.mms
- swift\_settconf\_mt54x\_tolmf.mms
- swift\_settinst\_mt54x\_tolmf.mms

2. Build all the maps in the source files, and then run the maps. See [SWIFT To LMF maps](#).

# How to run the example in Design Server

Run instructions for the SWIFT to LMF example.

## About this task

These steps describe how to run the SWIFT to LMF example in Design Server.

## Procedure

1. Import the swiftToLMF.zip file into the Design Server UI. See the release notes for import instructions.
2. Create a package for the project and build all the maps.
3. Open the swiftToLMF project in the Design Server UI.
4. Run the maps, see [SWIFT To LMF maps](#).
5. View the output. Go to the map Diagram-Result, and hover the cursor over Target, then select View Data.
6. To download the output file, see [How to download output files](#).

- [SWIFT To LMF maps](#)

The maps to be run are listed below:

- [How to download output files](#)

After you build/run maps, download the output files.

## SWIFT To LMF maps

The maps to be run are listed below:

- swift\_mt537\_tolmf (maps/swift\_mt537\_tolmf)
- swift\_settconf\_mt54x\_tolmf (maps/swift\_settconf\_mt54x\_tolmf)
- swift\_mt568\_tolmf (maps/swift\_mt568\_tolmf)
- tolmf\_packet\_payment\_101 (maps/swift\_mt101\_tolmf)
- tolmf\_packet\_tradeorder\_509 (maps/swift\_mt509\_tolmf)
- tolmf\_packet\_transfer\_mt210 (maps/swift\_mt210\_tolmf)
- swift\_mt548\_tolmf (maps/swift\_mt548\_tolmf)
- tolmf\_swift\_mt515 (maps/swift\_mt515\_tolmf)
- swift\_mt535\_tolmf (maps/swift\_mt535\_tolmf)
- tolmf\_packet\_common\_mtn91 (maps/swift\_mtn91\_tolmf)
- tolmf\_packet\_confirmation\_mt910 (maps/swift\_mt910\_tolmf)
- tolmf\_packet\_payment\_102 (maps/swift\_mt102\_tolmf)
- swift\_settinst\_mt54x\_tolmf (maps/swift\_settinst\_mt54x\_tolmf)
- tolmf\_packet\_confirmation\_mt900 (maps/swift\_mt900\_tolmf)
- tolmf\_packet\_transfer\_mt202 (maps/swift\_mt202\_tolmf)
- tolmf\_packet\_transfer\_mt202\_COV (maps/swift\_mt202\_COV\_tolmf)
- tolmf\_packet\_statement\_mt942 (maps/swift\_mt942\_tolmf)
- tolmf\_packet\_fx\_304 (maps/swift\_mt304\_tolmf)
- swift\_mt565\_tolmf (maps/swift\_mt565\_tolmf)
- tolmf\_packet\_fx\_305 (maps/swift\_mt305\_tolmf)
- swift\_mt564\_tolmf (maps/swift\_mt564\_tolmf)
- tolmf\_packet\_payment\_103 (maps/swift\_mt103\_tolmf)
- tolmf\_packet\_common\_mtn95 (maps/swift\_mtn95\_tolmf)
- tolmf\_packet\_320 (maps/swift\_mt320\_tolmf)
- tolmf\_packet\_transfer\_mt205 (maps/swift\_mt205\_tolmf)
- tolmf\_packet\_fx\_300 (maps/swift\_mt300\_tolmf)
- tolmf\_packet\_statement\_mt950 (maps/swift\_mt950\_tolmf)
- tolmf\_packet\_common\_mtn90 (maps/swift\_mtn90\_tolmf)
- swift\_mt536\_tolmf (maps/swift\_mt536\_tolmf)
- swift\_mt549\_tolmf (maps/swift\_mt549\_tolmf)
- tolmf\_packet\_common\_mtn96 (maps/swift\_mtn96\_tolmf)
- swift\_mt567\_tolmf (maps/swift\_mt567\_tolmf)
- swift\_mt566\_tolmf (maps/swift\_mt566\_tolmf)
- tolmf\_swift\_mt578 (maps/swift\_mt578\_tolmf)
- tolmf\_packet\_common\_mtn92 (maps/swift\_mtn92\_tolmf)
- tolmf\_packet\_transfer\_mt205\_COV (maps/swift\_mt205\_COV\_tolmf)
- tolmf\_packet\_order\_502 (maps/swift\_mt502\_tolmf)
- tolmf\_packet\_statement\_mt940 (maps/swift\_mt940\_tolmf)

## How to download output files

After you build/run maps, download the output files.

### About this task

To download the output file, go to **Files** in the Design Server UI and download the desired file using these steps:

### Procedure

1. Click the output Card Settings that you want to download.
2. From the Command text box, copy the name of the file that you want to download. Make certain that you copy only the file name.  
For example, if the Command text box displays:  

```
data/ach_contested_dishonored_return.out
```

copy only:  

```
ach_contested_dishonored_return.out
```
3. From Files, click the New icon to open the New File dialog box.
4. Paste the file name in the Name field.
5. Click the Folder field drop down and select data.
6. Click Ok.
7. Build and re-run the map.
8. Go to Files and select Download to view the data.

## SWIFT from LMF example

- [How to run the example in Design Studio](#)  
Run instructions for the SWIFT from LMF example.
- [How to run the example in Design Server](#)  
Run instructions for the SWIFT from LMF example.

## How to run the example in Design Studio

Run instructions for the SWIFT from LMF example.

### About this task

These instructions describe how to run the SWIFT from LMF example in Design Studio.

### Procedure

1. Use the Design Studio to open the following map source files:
  - swift\_ca\_event\_fromlmf.mms
  - swift\_mt101\_fromlmf.mms
  - swift\_mt102\_fromlmf.mms
  - swift\_mt103\_fromlmf.mms
  - swift\_mt202\_COV\_fromlmf.mms
  - swift\_mt202\_fromlmf.mms
  - swift\_mt205\_COV\_fromlmf.mms
  - swift\_mt205\_fromlmf.mms
  - swift\_mt210\_fromlmf.mms
  - swift\_mt300\_fromlmf.mms
  - swift\_mt304\_fromlmf.mms
  - swift\_mt305\_fromlmf.mms
  - swift\_mt320\_fromlmf.mms
  - swift\_mt502\_fromlmf.mms
  - swift\_mt509\_fromlmf.mms
  - swift\_mt515\_fromlmf.mms
  - swift\_mt535\_fromlmf.mms
  - swift\_mt536\_fromlmf.mms
  - swift\_mt537\_fromlmf.mms
  - swift\_mt548\_fromlmf.mms
  - swift\_mt549\_fromlmf.mms
  - swift\_mt578\_fromlmf.mms
  - swift\_mt900\_fromlmf.mms
  - swift\_mt910\_fromlmf.mms
  - swift\_mt940\_fromlmf.mms
  - swift\_mt942\_fromlmf.mms
  - swift\_mt950\_fromlmf.mms
  - swift\_mtn90\_fromlmf.mms
  - swift\_mtn91\_fromlmf.mms
  - swift\_mtn92\_fromlmf.mms
  - swift\_mtn95\_fromlmf.mms
  - swift\_mtn96\_fromlmf.mms
  - swift\_settconf\_mt54x\_fromlmf.mms
  - swift\_settinst\_mt54x\_fromlmf.mms
2. Build all the maps in the source files, and then run the maps. See [SWIFT from LMF maps](#).

## How to run the example in Design Server

Run instructions for the SWIFT from LMF example.

### About this task

These instructions describe how to run the SWIFT from LMF example in Design Server.

### Procedure

1. Import the swiftFromLMF.zip file into the Design Server UI. See the release notes for import instructions.
  2. Create a package for the project and build all the maps.
  3. Open the swiftFromLMF project in the Design Server UI.
  4. Run the maps, see [SWIFT from LMF maps](#).
  5. View the output. Go to the map Diagram-Result, and hover the cursor over Target, then select View Data.
  6. To download the output file, see [How to download output files](#).
- [SWIFT from LMF maps](#)  
The maps to be run are listed below:

- [How to download output files](#)

After you build/run maps, download the output files.

---

## SWIFT from LMF maps

The maps to be run are listed below:

- fromlmf\_packet\_transfer\_mt210 (maps/swift\_mt210\_fromlmf)
- fromlmf\_packet\_payment\_102 (maps/swift\_mt102\_fromlmf)
- fromlmf\_packet\_confirmation\_910 (maps/swift\_mt910\_fromlmf)
- fromlmf\_packet\_common\_mtn96 (maps/swift\_mtn96\_fromlmf)
- swift\_settconf\_mt54x\_fromlmf (maps/swift\_mt54x\_fromlmf)
- fromlmf\_packet\_statement\_950 (maps/swift\_mt950\_fromlmf)
- fromlmf\_packet\_fx\_305 (maps/swift\_mt305\_fromlmf)
- swift\_mt549\_fromlmf (swift\_mt549\_fromlmf (maps/swift\_mt549\_fromlmf))
- fromlmf\_packet\_common\_mtn95 (maps/swift\_mtn95\_fromlmf)
- from\_packet\_fx\_300 (maps/swift\_mt300\_fromlmf)
- fromlmf\_packet\_transfer\_mt205 (maps/swift\_mt205\_fromlmf)
- fromlmf\_swift\_mt509 (maps/swift\_mt509\_fromlmf)
- fromlmf\_packet\_statement\_942 (maps/swift\_mt942\_fromlmf)
- fromlmf\_packet\_payment\_101 (maps/swift\_mt101\_fromlmf)
- swift\_stttinst\_mt54x\_fromlmf (maps/swift\_mt54x\_fromlmf)
- fromlmf\_packet\_common\_mtn91 (maps/swift\_mtn91\_fromlmf)
- fromlmf\_packet\_statement\_940 (maps/swift\_mt940\_fromlmf)
- fromlmf\_packet\_confirmation\_900 (maps/swift\_mt900\_fromlmf)
- swift\_mt548\_fromlmf (maps/swift\_mt548\_fromlmf)
- fromlmf\_packet\_order\_502 (maps/swift\_mt502\_fromlmf)
- fromlmf\_packet\_common\_mtn92 (maps/swift\_mtn92\_fromlmf)
- swift\_ca\_event\_fromlmf (maps/swift\_ca\_event\_fromlmf)
- fromlmf\_packet\_fx\_304 (maps/swift\_mt304\_fromlmf)
- swift\_mt535\_fromlmf (maps/swift\_mt535\_fromlmf)
- swift\_mt537\_fromlmf (maps/swift\_mt537\_fromlmf)
- fromlmf\_packet\_trans\_mt202\_COV (maps/swift\_mt202\_fromlmf)
- fromlmf\_packet\_common\_mtn90 (maps/swift\_mtn90\_fromlmf)
- fromlmf\_swift\_mt578 (maps\_mt578\_fromlmf)
- fromlmf\_swift\_mt515 (maps/swift\_mt515\_fromlmf)
- swift\_mt536\_fromlmf (maps/swift\_mt536\_fromlmf)
- fromlmf\_packet\_trans\_mt205\_COV (maps/swift\_mt205\_fromlmf)
- fromlmf\_packet\_transfer\_mt202 (maps/swift\_mt202\_fromlmf)
- fromlmf\_packet\_loandepo\_320 (maps/swift\_mt320\_fromlmf)
- from\_lmf\_payment\_packet\_103 (maps/swift\_mt103\_fromlmf)

---

## How to download output files

After you build/run maps, download the output files.

### About this task

To download the output file, go to **Files** in the Design Server UI and download the desired file using these steps:

### Procedure

1. Click the output Card Settings that you want to download.
2. From the Command text box, copy the name of the file that you want to download. Make certain that you copy only the file name.  
For example, if the Command text box displays:  

```
data/ach_contested_dishonored_return.out
```

  
copy only:  

```
ach_contested_dishonored_return.out
```
3. From Files, click the New icon to open the New File dialog box.
4. Paste the file name in the Name field.
5. Click the Folder field drop down and select data.
6. Click Ok.
7. Build and re-run the map.
8. Go to Files and select Download to view the data.

---

## SWIFT MT Compliance example

- [How to run the example in Design Studio](#)  
Run instructions for the SWIFT MT Compliance Example.
  - [How to run the example in Design Server](#)  
Run instructions for the SWIFT MT Compliance Example.
- 

## How to run the example in Design Studio

Run instructions for the SWIFT MT Compliance Example.

### About this task

These steps describe how to run the SWIFT MT Compliance example in Design Studio.

### Procedure

1. Use the Design Studio to open the following map source file: validate\_messages.mms.
  2. Build the maps in the source file, and then run the map:
    - validate\_message
- 

## How to run the example in Design Server

Run instructions for the SWIFT MT Compliance Example.

### About this task

These steps describe how to run the SWIFT MT Compliance example in Design Server.

### Procedure

1. Import the swiftMTCompliance.zip file into the Design Server UI. See the release notes for import instructions.
  2. Create a package for the project and build all the maps.
  3. Open the swiftMTCompliance project in the Design Server UI.
  4. Run the validate\_messages map.
  5. View the output. Go to the map Diagram-Result, and hover the cursor over Target, then select View Data.
  6. To download the output file, see [How to download output files](#).
    - [How to download output files](#)  
After you build/run maps, download the output files.
- 

## How to download output files

After you build/run maps, download the output files.

### About this task

To download the output file, go to **Files** in the Design Server UI and download the desired file using these steps:

### Procedure

1. Click the output Card Settings that you want to download.
  2. From the Command text box, copy the name of the file that you want to download. Make certain that you copy only the file name.  
For example, if the Command text box displays:

```
data/ach_contested_dishonored_return.out
```

copy only:

```
ach_contested_dishonored_return.out
```
  3. From Files, click the New icon to open the New File dialog box.
  4. Paste the file name in the Name field.
  5. Click the Folder field drop down and select data.
  6. Click Ok.
  7. Build and re-run the map.
  8. Go to Files and select Download to view the data.
- 

## MT JVC Report example

This example takes a file in SWIFT Alliance Access (SAA) format containing a batch of SWIFT messages and uses the JVC to create a validation report.

The report shows the validation status of each message. The example data file contains both valid and invalid messages, so that both validation passes and failures are demonstrated.

- [Configuration of SWIFT MT JVC Report example in Design Studio](#)  
This topic documents how to configure the MT JVC Report example in Design Studio.
- [Configuration of SWIFT MT JVC Report example in Design Server](#)

---

## Configuration of SWIFT MT JVC Report example in Design Studio

This topic documents how to configure the MT JVC Report example in Design Studio.

### Before you begin

The jvalccyy.prop is installed in `<install_dir>/packs/financial_payments_vn.n.n.n/swift/mt/jvc/maps` for all versions of ITX.

For Transformation Extender V8.n, the jvalccyy.prop file can be in any directory, however the map calling the JVC must be in the same directory as jvalccyy.prop.

### Procedure

Move the jvalccyy.prop in the following location for ITX V.9.n and higher versions:

- Windows: `<tx_install_dir>`
- Linux: `<tx_install_dir>/bin`
- [Additional configuration for MT JVC validation](#)
- [How to run the example in Design Studio](#)  
The MT JVC Report example map creates a JVC validation report from a batch of SWIFT messages.

---

## Additional configuration for MT JVC validation

### Pre-requisites

The swiftMTjvcConfig.tar.gz file is contained within UIProjectImports directory.

The UIProjectImports directory is located in the following location:

`<install_dir>/packs/financial_payments_vn.n.n.n/UIProjectImports`

Where, `<install_dir>` is the location where the pack zip distro was extracted and n.n.n.n is the current version of the pack.

Extract the following from swiftMTjvcConfig.tar.gz file:

- jvcwrap.jar
- jvalccyy.jar

---

## How to run the example in Design Studio

The MT JVC Report example map creates a JVC validation report from a batch of SWIFT messages.

### Before you begin

Refer to [Additional configuration for MT JVC validation](#).

For Design Studio, will require administrative rights:

Copy jars to `<TX_install_dir>/extjar`.

For Design Server installation:

See [Configuration files for Design Server](#) for the steps to perform before you import `<project>.zip` file.

where `<project>.zip` can be any of the following:

- swiftJVCReport.zip

### About this task

These steps describe how to run the MT JVC Report example in Design Studio:

### Procedure

1. Copy the following file into <install\_dir>/packs/financial\_payments\_vn.n.n.n/swift/mt/examples/jvc\_report:
  - For versions of the base product, prior to version 9.0.0, copy file:  
<install\_dir>/packs/financial\_payments\_vn.n.n.n/swift/mt/jvc/maps/jvalccyy.prop
  - For versions of the base product, version 9.0.0 and later:  
Copy <install\_dir>/packs/financial\_payments\_vn.n.n.n/swift/mt/jvc/maps/jvalccyy.prop to the base product installation directory.
2. Edit the copied jvalccyy.prop file ensure that the Network Rule validation is ON, BIC validation is OFF and terse mode is ON:
  - rules.networkvalidation.enabled=true
  - bic.file.enabled=false
  - ReportingMode=terse
3. Use the Design Studio to open the following map source file <install\_dir>/packs/financial\_payments\_vn.n.n.n/swift/mt/jvc/maps/validate\_messages.
  - a. Compile the executable map "validate\_messages".
  - b. Copy the resultant validate\_messages.mmc file to the jvc\_report example folder.
4. Use the Design Studio to open the following map source file <install\_dir>/packs/financial\_payments\_vn.n.n.n/swift/mt/examples/jvc\_report/jvc\_report and compile the executable map jvc\_report.
5. Run the jvc\_report. The JVC validation results for each MT message in the input file are reported in the output file, jvc\_report.txt, located in the same folder as the example.

## Configuration of SWIFT MT JVC Report example in Design Server

### Configuration files for Design Server

The swiftMTjvcConfig.tar.gz file is contained within UIPackageImports directory.

The UIPackageImports directory is located in the following location:

<install\_dir>/packs/financial\_payments\_vn.n.n.n/UIPackageImports

Where, <install\_dir> is the location where the pack zip distro was extracted and n.n.n.n is the current version of the pack.

Use these steps to run this file:

Extract the swiftMTjvcConfig.tar.gz file.

Follow the below steps, before you import <project>.zip file.

Perform the steps in the same directory location as the Design Server is installed:

1. Create a temp directory called packs.  
Example: /home/user/packs
2. For UNIX environment, FTP (in binary mode) the swiftMTjvcConfig.tar.gz to the pack's directory defined in Step 1.
3. Untar the swiftMTjvcConfig.tar.gz file.  
Example: \$ tar -zxf swiftMTjvcConfig.tar.gz
4. For Container based Design Server installation:
  - a. Run the **jvcsetup.sh** script.  
Example: \$ ./jvcsetup.sh
  - b. The script copies the following to the ITX server:
    - jvcwrap.jar
    - jvalccyy.jar
 packs/swift folder and sub-folders includes the jvalccyy.prop and XML metadata.

#### Optional:

- a. Run the **cleanjvc.sh** script.  
Example: \$ ./cleanjvc.sh
  - b. The script removes the following to the ITX server:
    - jvcwrap.jar
    - jvalccyy.jar
 packs/swift folder and sub-folders includes the jvalccyy.prop and XML metadata.
5. For Native based installation:
    - a. Copy the jvalccyy.jar and jvcwrap.jar into the directory defined in config.yaml server.persistence.files, by default, this is set to /opt/tlxlibs.
    - b. Restart the application running ./ITX stop and then ./ITX start.
  6. After importing <project>.zip project:
    - a. Copy the jvalccyy.prop file to the workdir directory under directory set in the config.yaml server.persistence.files, the default settings is /opt/tlxfiles/workdir.
    - b. On the project, update the map rule on the validate\_message map, on output card #1 item JvalPropPath. Change the map rule from =NONE to the location of the jvalccyy.prop.

Example:

    - On UNIX  
="DS\_DATA\_DIR/workdir"
    - On WINDOWS  
="DS\_DATA\_DIR\\workdir"

where DS\_DATA\_DIR refers to the Design Server data directory, set in the config.yamlserver.persistence.files previously TX\_FILE\_DIR

7. Restart the Design Server: ./ITX stop and ./ITX start.

Note: To reflect any jvalccyy.prop updates, repeat step 7.

- [How to run the example in Design Server](#)

Run instructions for the SWIFT MT JVC Report example.

## How to run the example in Design Server

Run instructions for the SWIFT MT JVC Report example.

### About this task

Follow these steps to run the SWIFT MT JVC Report example in Design Server.

### Procedure

1. Import the swiftJVCReport.zip file into the Design Server UI. See the release notes for import instructions.
2. Create a package for the project and build all the maps.
3. Open the swiftJVCReport project in the Design Server UI.
4. Run the maps:
  - jvc\_report (maps/jvc\_report)
  - validate\_messages (maps/validate\_messages)
5. View the output. Go to the map Diagram-Result, and hover the cursor over Target, then select View Data.
6. To download the output file, see [How to download output files](#).
  - [How to download output files](#)  
After you build/run maps, download the output files.

## How to download output files

After you build/run maps, download the output files.

### About this task

To download the output file, go to **Files** in the Design Server UI and download the desired file using these steps:

### Procedure

1. Click the output Card Settings you want to download.
2. From the Command text box, copy the name of the file that you want to download. Make certain that you copy only the file name.  
For example, if the Command text box displays:  
  
data/ach\_contested\_dishonored\_return.out  
  
copy only:  
  
ach\_contested\_dishonored\_return.out
3. From Files, click the New icon to open the New File dialog box.
4. Paste the file name in the Name field.
5. Click the Folder field drop down and select data.
6. Click Ok.
7. Build and re-run the map.
8. Go to Files and select Download to view the data.

## MT950 Splitter Map example

The MT950 Splitter Map example contains 2 maps that demonstrate techniques for splitting a 5k MT950 Statement message into smaller MT950 messages.

These are the maps and the technique used by each:

### mt950\_split

This map uses recursion (that is, the map calls itself recursively) to perform the split. This method requires only one data pass to split the MT950 into smaller messages and to recalculate opening and closing balances on each split message. However, this method does not scale well due to the resources consumed during recursion, and can exceed the thread stack size limit.

### mt950\_split\_v2

This map uses the CLONE function to perform the split. This method scales better than the recursive map. The disadvantage is that two data passes are required to split the MT950 into smaller messages and to recalculate opening and closing balances on each split message.

- [MT 950 Splitter Map example overview](#)

The example maps split a large MT950 Statement message and generate multiple, smaller MT950 files.

## MT 950 Splitter Map example overview

The example maps split a large MT950 Statement message and generate multiple, smaller MT950 files.

For each of the smaller files the example maps recalculate running balances, and maintain the message sequence number within the overall statement number. Both example maps work on the following algorithm:

Variables names match the map input card names:

- n - Indicates the number of statement lines in each target message. The default is 30, but this parameter can be adjusted as necessary, based on the average expected size of a statement line.
- t - Indicates the number of lines in the source message. This value is calculated by the map using COUNT.
- T - Indicates total number of target messages. Calculated as INT(t/n)+1
- x - Indicates current target message number iteration (incremented by the map).

As an example: n=30, t=70, then T=INT(70/30)+1=3

therefore, three target messages are needed

The Target lines are built for each iteration using EXTRACT on the source message statement line's INDEX:

```
=EXTRACT (StatementLine #61 General Field WHERE
INDEX (StatementLine #61 General Field) >= (n*(x-1)+1 &
INDEX (StatementLine #61 General Field) <= x*n)
mt950_split is executed as a recursive map until x > T
mt950_split_v2 is executed T times using CLONE to control the iteration.
```

- [How to run the example in Design Studio](#)  
Run instructions for the MT950 Splitter Map example.
- [How to run the example in Design Server](#)  
Run instructions for the MT950 Splitter Map example.

## How to run the example in Design Studio

Run instructions for the MT950 Splitter Map example.

### About this task

These steps describe how to run the MT950 Splitter Map example in Design Studio.

### Procedure

1. Use the Design Studio to open the following map source files:
  - mt950\_split.mms
  - mt950\_split\_v2.mms
2. Build the maps in the source files, and then run the following maps:
  - mt950\_control (maps/mt950\_split)
  - v2\_mt950\_recalc (maps/mt950\_split\_v2)
  - v2\_mt950\_control (maps/mt950\_split\_v2)

## How to run the example in Design Server

Run instructions for the MT950 Splitter Map example.

### About this task

These steps describe how to run the MT950 Splitter Map example in Design Server.

### Procedure

1. Import the swiftMT950Split.zip file into the Design Server UI. See the release notes for import instructions.
2. Create a package for the project and build all the maps.
3. Open the swiftMT950Split project in the Design Server UI.
4. Run the maps.
  - mt950\_control (maps/mt950\_split)
  - v2\_mt950\_recalc (maps/mt950\_split\_v2)
  - v2\_mt950\_control (maps/mt950\_split\_v2)
5. View the output. Go to the map Diagram-Result, and hover the cursor over Target, then select View Data.

6. To download the output file, see [How to download output files](#).

- [How to download output files](#)

After you build/run maps, download the output files.

- [How to run mt950\\_split\\_v2](#)

---

## How to download output files

After you build/run maps, download the output files.

### About this task

To download the output file, go to **Files** in the Design Server UI and download the desired file using these steps:

### Procedure

1. Click the output Card Settings that you want to download.
2. From the Command text box, copy the name of the file that you want to download. Make certain that you copy only the file name.

For example, if the Command text box displays:

```
data/ach_contested_dishonored_return.out
```

copy only:

```
ach_contested_dishonored_return.out
```

3. From Files, click the New icon to open the New File dialog box.
4. Paste the file name in the Name field.
5. Click the Folder field drop down and select data.
6. Click Ok.
7. Build and re-run the map.
8. Go to Files and select Download to view the data.

---

## How to run mt950\_split\_v2

### About this task

These steps describe how to run the mt950\_split\_v2 map.

### Procedure

1. Use the Map Designer to open mt950\_split\_v2 . It contains these executable maps: v2\_mt950\_control, v2\_mt950\_runmap\_split, v2\_mt950\_recalc and v2\_mt950\_runmap\_balance.
  - a. v2\_mt950\_control counts the number of statement lines (t), supplies the default value for n (30), calculates T and calls v2\_mt950\_runmap\_split.
  - b. v2\_mt950\_runmap\_split generates smaller messages based on the parameters supplied by the calling map. The split messages are all contained in a single file called mt950\_split.txt. This is an intermediate file, where the opening and closing balances have not been recalculated
  - c. v2\_mt950\_recalc calls v2\_mt950\_runmap\_balance for each split message in mt950\_split.txt
  - d. v2\_mt950\_runmap\_balance recalculates the opening and closing balance for each split message. The closing balance from one message is carried over into the following message using a work file called ClosingBal.dat.
2. Build all four maps.
3. Run the v2\_mt950\_control map. This will create the intermediate file, mt950\_split.txt.
4. Run the v2\_mt950\_recalc map. This will recalculate balances and create a separate file for each split MT950 using the following naming convention:  
mt950\_recalc\_StatementNo(from Tag 28C)\_SequenceNo\_TimeStamp.txt

These files are located in the same folder as this example.

---

## MT Launcher Validation framework

This sample validation framework uses the Transformation Extender Launcher and the JVC to perform trigger-based MT validation.

The Transformation Extender Launcher Edition is required to run this example.

The framework for this sample is triggered by placing a file containing a single MT message into a folder, which causes the framework to be invoked to validate the message. Depending on the results of the validation, the framework either creates an error report file, or passes the message through unchanged. After being processed, the file is deleted from the input folder. Using this method it is possible to validate numerous separate files by placing them into the input folder. The framework, therefore, consists of a folder structure for processing the data, a map source file for performing the validation, and a Launcher System Definition file for deploying to the Launcher. The framework is installed into this directory location: <install\_dir>/packs/financial\_payments\_vn.n.n.n/swift/mt/examples/validation

Where *install\_dir* is the installed location of the installation location of the Pack for Financial Payments and *n.n.n.n* is the version number of the Pack for Financial Payments.

The structure of this top-level folder is as follows:

```

data/input
Placeholder for files to be validated
data/output/error/
Placeholder for error reports.
data/output/passed/
Placeholder for files that pass validation.

maps/execution/
Executable maps are deployed here.
maps/source/validate_mt
Where validate_mt is the source map file for deployment.
msds/validate_mt.msd
Where validate_mt.msd is the system definition file for deployment.

```

The following instructions are for deploying and running the framework on a Windows platform. Refer to the Transformation Extender documentation for instructions on running the Launcher on other platforms.

- [\*\*Deploying the Launcher Validation Framework\*\*](#)  
These procedures describe how to deploy the Launcher Validation Framework.
- [\*\*Running the Launcher Validation Framework\*\*](#)  
Perform the following procedure to run the Launcher Validation Framework:

## Deploying the Launcher Validation Framework

These procedures describe how to deploy the Launcher Validation Framework.

### Procedure

1. Open validate\_mt.msd using the Integration Flow Designer. The Integration Flow Designer is part of the IBM Transformation Extender Design Studio.
2. Right-click on validate\_mt system, and select Deploy/Local. The maps are then compiled, and a pop-up box will state "The Deploy operation has completed successfully. Would you like to see the results?"
3. Click Yes, and view the MSEDploy.TXT file. There should be no errors, and a message should state "Deploy has completed successfully". If the deploy has failed the most likely explanation is an error with the installation.
4. Review the MSEDploy.TXT to locate the source of the error. If successful, the Launcher execution file validate\_mt.msl is placed in the <tx\_install\_dir>/systems folder.

### Results

If successful, the jvalccyy.prop file is copied to the maps/execution folder. Review this file to ensure it has the required parameter settings. See the JVC documentation for details on configuring the JVC.

## Running the Launcher Validation Framework

Perform the following procedure to run the Launcher Validation Framework:

1. Using the Launcher Administration tool Administration tool, create a Launcher System.  
Note: The Launcher Administration tool installs with the Transformation Extender Launcher Edition
2. Start the Launcher processes.
3. Start the Management Console, and connect it to your Launcher System. The system is now ready to process files.  
Note: Refer to the Transformation Extender Launcher documentation for details about how to set up and run a Launcher System.
4. Place a file containing a single MT message in the data/input folder.
  - If it passes validation, it is copied to the data/output/passed folder, and deleted from the data/input folder.
  - If it fails validation, an error report is placed in the data/output/error folder, and the file is deleted from the data/input folder. To ensure uniqueness, each filename is appended with a date-time stamp when placed in the passed folder.

Error files are in a simple XML format (there is schema for this), and are also appended with a date-time stamp. The actual format of the error report depends upon the following factors:

- Whether the JVC is run in verbose mode, or in terse mode.
- The type of error - for example, a low-level syntax error versus a conditional rule error.
- Whether the data in error was multi-line or single line.

## MT - MX translation maps example (Deprecated)

The MT - MX translation maps example shows how the SWIFT component can be used to translate between MT and MX messages.

The MT - MX translation maps example provides examples for the Payments, Funds, and Corporate Actions business areas.

The example can be found in the following location:

<install\_dir>/packs/financial\_payments\_vn.n.n.n/swift/mx/examples/mt\_mx\_translation

- [How to run the example in Design Studio](#)

This example describes how to run the MT / MX Translation Maps example.

- [How to run the example in Design Server](#)

---

## How to run the example in Design Studio

This example describes how to run the MT / MX Translation Maps example.

### About this task

---

Follow these steps to run the example.

### Procedure

---

1. Run the build script build\_mt\_mx\_translation.bat.
2. This script can be found in the maps sub-directory. This will compile all of the maps in the example.
3. Check the log produced by the script to check there are no compilation errors. The log is called build\_deploy\_mt\_mx\_translation.log. The log is located in the maps sub-folder.
4. Use the Design Studio to open the following map source files:
  - mt101\_mx\_pain\_001.mms
  - mt103\_mx\_pacs\_008.mms
  - mt202\_COV\_mx\_pacs\_009.mms
  - mt202\_mx\_pacs\_009.mms
  - mt502\_mx\_setr\_004.mms
  - mt502\_mx\_setr\_010.mms
  - mt509\_mx\_setr\_016.mms
  - mt515\_mx\_setr\_006.mms
  - mt515\_mx\_setr\_012.mms
  - mt564\_mx\_seev\_031.mms
  - mt565\_mx\_seev\_033.mms
  - mt566\_mx\_seev\_036.mms
  - mt900\_mx\_camt\_054.mms
  - mt910\_mx\_camt\_054.mms
  - mt940\_mx\_camt\_053.mms
  - mt941\_mx\_camt\_052.mms
  - mt942\_mx\_camt\_052.mms
  - mt950\_mx\_camt\_053.mms
  - mt\_funds\_xtnsn.mms
  - mt\_mx\_runmaps.mms
  - mx\_funds\_preconversion.mms
  - mx\_pacs\_008\_mt\_103core.mms
  - mx\_pacs\_009\_mt202.mms
  - mx\_pacs\_009\_mt202\_cov.mms
  - mx\_pain\_001\_mt101.mms
  - mx\_seev\_031\_mt564.mms
  - mx\_seev\_033\_mt565.mms
  - mx\_seev\_036\_mt566.mms
  - mx\_setr\_004\_mt502.mms
  - mx\_setr\_006\_mt515.mms
  - mx\_setr\_010\_mt\_502.mms
  - mx\_setr\_012\_mt515.mms
  - mx\_setr\_016\_mt509.mms
5. Run the following maps:
  - cor15001\_mt564\_seev\_031
  - cor15004\_mt565\_seev\_033
  - cor15007\_mt566\_seev\_036
  - cor25001\_seev\_031\_mt564
  - cor25004\_seev\_033\_mt565
  - cor25007\_seev\_036\_mt566
  - pym11001\_mt101\_pain1
  - pym11002\_pain1\_mt101
  - pym51001\_mt103core\_pacs8
  - pym51002\_mt103plus\_pacs8
  - pym51003\_mt202\_pacs9
  - pym51004\_mt202\_cov\_pacs9
  - pym61001\_pacs8\_mt103core
  - pym61002\_pacs9\_mt202
  - pym61003\_pacs9\_mt202\_cov
  - pym91001\_mt900\_camt54
  - pym91002\_mt910\_camt54
  - pym91003\_mt940\_camt53
  - pym91004\_mt941\_camt52
  - pym91005\_mt942\_camt52
  - pym91006\_mt950\_camt53
  - snf51003\_mt502\_setr\_004
  - snf51007\_mt502\_setr\_010

- snf51011\_mt509\_setr\_016
  - snf51015\_mt515\_setr\_006
  - snf51016\_mt515\_setr\_012
  - snf61003\_setr\_004\_mt502
  - snf61007\_setr\_010\_mt502
  - snf61011\_setr\_016\_mt509
  - snf61015\_setr\_006\_mt515
  - snf61016\_setr\_012\_mt515
- 

## How to run the example in Design Server

### How to run the MT to MX Translation maps example in Design Server:

1. Import the swiftMTmxTranslation.zip file into the Design Server UI. See the release notes for import instructions.
2. Create a package for the project and build all the maps.
3. Open the swiftMTmxTranslation project in the Design Server UI.
4. Run the following maps:
  - cor15001\_mt564\_seev\_031 (maps/mt564\_mx\_seev\_031)
  - cor15004\_mt565\_seev\_033 (maps/mt565\_mx\_seev\_033)
  - cor15007\_mt566\_seev\_036 (maps/mt566\_mx\_seev\_036)
  - pym11001\_mt101\_pain1 (maps/mt101\_mx\_pain\_001)
  - pym51001\_mt103core\_pacs8 (maps/mt103\_mx\_pacs\_008)
  - pym51002\_mt103plus\_pacs8 (maps/mt103\_mx\_pacs\_008)
  - pym51003\_mt202\_pacs9 (maps/mt202\_mx\_pacs\_009)
  - pym51004\_mt202\_cov\_pacs9 (maps/mt202\_COV\_mx\_pacs\_009)
  - pym91001\_mt900\_camt54 (maps/mt900\_mx\_camt\_054)
  - pym91002\_mt910\_camt54 (maps/mt910\_mx\_camt\_054)
  - pym91003\_mt940\_camt53 (maps/mt940\_mx\_camt\_053)
  - pym91004\_mt941\_camt52 (maps/mt941\_mx\_camt\_052)
  - pym91005\_mt942\_camt52 (maps/mt942\_mx\_camt\_052)
  - pym91006\_mt950\_camt53 (maps/mt950\_mx\_camt\_053)
  - snf51003\_mt502\_setr\_004 (maps/mt502\_mx\_setr\_004)
  - snf51007\_mt502\_setr\_010 (maps/mt502\_mx\_setr\_010)
  - snf51011\_mt509\_setr\_016 (maps/mt509\_mx\_setr\_016)
  - snf51015\_mt515\_setr\_006 (maps/mt515\_mx\_setr\_006)
  - snf51016\_mt515\_setr\_012 (maps/mt515\_mx\_setr\_012)
5. View the output. Go to the map Diagram-Result, and hover the cursor over Target, then select View Data.
6. To download the output file, see [How to download output files](#).

### How to run the MX to MT Translation maps example in Design Server:

1. Import the swiftMXmtTranslation.zip file into the Design Server UI. See the release notes for import instructions.
2. Create a package for the project and build all the maps.
3. Open the swiftMTmxTranslation project in the Design Server UI.
4. Run the following maps:
  - cor25001\_seev\_031\_002\_08\_mt564 (maps/mx\_seev\_031\_mt564)
  - cor25004\_seev\_033\_mt565 (maps/mx\_seev\_033\_mt565)
  - cor25007\_seev\_036\_mt566 (maps/mx\_seev\_036\_mt566)
  - pym11002\_pain1\_mt101 (maps/mx\_pain\_001\_mt101)
  - pym61001\_pacs8\_mt103core (maps/mx\_pacs\_008\_mt\_103core)
  - pym61002\_pacs9\_mt202 (maps/mx\_pacs\_009\_mt202)
  - pym61003\_pacs9\_mt202\_cov (maps/mx\_pacs\_009\_mt202\_cov)
  - snf61003\_setr\_004\_mt502 (maps/mx\_setr\_004\_mt502)
  - snf61007\_setr\_010\_mt502 (maps/mx\_setr\_010\_mt502)
  - snf61011\_setr\_016\_mt509 (maps/mx\_setr\_016\_mt509)
  - snf61015\_setr\_006\_mt515 (maps/mx\_setr\_006\_mt515)
  - snf61016\_setr\_012\_mt515 (maps/mx\_setr\_012\_mt515)
5. View the output. Go to the map Diagram-Result, and hover the cursor over Target, then select View Data.
6. To download the output file, see [How to download output files](#).
- **Building multiple maps using Deploy**  
To deploy large numbers of maps (generally, when a project contains more than 10 maps), use the instructions provided here.

---

## Building multiple maps using Deploy

To deploy large numbers of maps (generally, when a project contains more than 10 maps), use the instructions provided here.

### About this task

Follow these steps to deploy large numbers of maps:

### Procedure

1. From the Design Server UI, click on the Deploy drop down, and click Packages.
2. Define a new Package, then click Add icon.
3. Define a Name field on the Edit Package dialog.
4. From the Project drop down, select the imported project.
5. Check top level box on Maps and Files drop down, this select all maps and files in the project
6. Save the Package.
7. Return to Design Server UI, main page, click Deploy drop down, scroll down and select Deploy from the list on the left.
8. Select the desired server (see the Server definition on Deployment > Deployment overview) and select the package.
9. Click Deploy.
10. Return to the Design Server UI, main page, click Design, select imported project (one just deployed).
11. Open the map, see [How to run the example in Design Server](#).

---

## SWIFT MX Message Bundle and Unbundle map example using Design Server

This example demonstrates the use of a map that splits the MX message Business Application Header (BAH) from the MX message data, and also packages the BAH with the message data.

Any MX message type requiring packaging with a BAH can be used with this map.

- [How to run the example in Design Studio](#)  
Run instructions for the Swift MX Message Bundle example.
- [How to run the example in Design Server](#)  
Run instructions for the SWIFT MX Message Bundle and Unbundle map example.

---

## How to run the example in Design Studio

Run instructions for the Swift MX Message Bundle example.

### About this task

These steps describe how to run the Swift MX Message Bundle example in Design Studio.

### Procedure

1. Use the Design Studio to open the following map source file: mxmessage.mms
2. Build all the maps in the source file, and then run the following transformation maps:
  - mxpack
  - mxunpack

---

## How to run the example in Design Server

Run instructions for the SWIFT MX Message Bundle and Unbundle map example.

### About this task

Follow these steps to run the SWIFT MX Message Bundle and Unbundle map example in Design Server.

### Procedure

1. Import the swiftMXmsgPackage.zip file into the Design Server UI. See the release notes for import instructions.
2. Create a package for the project and build all the maps.
3. Open the **swiftMXmsgPackage** project in the Design Server UI.
4. Run the following maps:
  - mxpack
  - mxunpack
5. View the output. Go to the map Diagram-Result, and hover the cursor over Target, then select View Data.
6. To download the output file, see [How to download output files](#).
  - [How to download output files](#)  
After you build/run maps, download the output files.
  - [Example executable maps](#)  
There are two example executable maps in the MX Message Bundle and Unbundle example:

---

## How to download output files

After you build/run maps, download the output files.

## About this task

---

To download the output file, go to **Files** in the Design Server UI and download the desired file using these steps:

## Procedure

---

1. Click the output Card Settings you want to download.
2. From the Command text box, copy the name of the file that you want to download. Make certain that you copy only the file name.

For example, if the Command text box displays:

data/ach\_contested\_dishonored\_return.out

copy only:

ach\_contested\_dishonored\_return.out

3. From Files, click the New icon to open the New File dialog box.
4. Paste the file name in the Name field.
5. Click the Folder field drop down and select data.
6. Click Ok.
7. Build and re-run the map.
8. Go to Files and select Download to view the data.

---

## Example executable maps

There are two example executable maps in the MX Message Bundle and Unbundle example:

- mxpack - Executable map that bundles the SWIFT message data with the required Business Application Header (BAH). This map takes the BAH and the message data and packages them into a single output with a wrapper around the existing pieces of data. The map uses the mxpackage type tree, with the BAH and the MX data being treated as blob item text elements. The BAH input data is separated from the namespace information, and the rest of the message is treated as a text blob and packaged with the message document data which is treated the same as the BAH data.
- mxunpack - Executable map that unbundles the SWIFT message data and the BAH, creating two output files. The input uses the mxpackage type tree to determine the BAH group and the MX message group. The first output file contains the BAH data mapped directly from the input. The second output file contains the MX message.

---

## SWIFT MX validation example

This example demonstrates SWIFT MX validation.

- [How to run the example in Design Studio](#)  
Run instructions for the Swift MX Validation example.
- [How to run the example in Design Server](#)  
Run instructions for the SWIFT MX Validation example.

---

## How to run the example in Design Studio

Run instructions for the Swift MX Validation example.

## About this task

---

These steps describe how to run the Swift MX Validation example in Design Studio.

## Procedure

---

1. Use the Design Studio to open the following map source file: swiftval.mms.
2. Build the map in the source file, and then run the map:
  - swiftval

---

## How to run the example in Design Server

Run instructions for the SWIFT MX Validation example.

## About this task

---

Follow these steps to run the SWIFT MX Validation example in Design Server.

## Procedure

---

1. Import the swiftMXCompliance.zip file into the Design Server UI. See the release notes for import instructions.
  2. Create a package for the project and build all the maps.
  3. Open the swiftMXCompliance project in the Design Server UI.
  4. Run the swiftval map.
  5. View the output. Go to the map Diagram-Result, and hover the cursor over Target, then select View Data.
  6. To download the output file, see [How to download output files](#).
- [How to download output files](#)  
After you build/run maps, download the output files.
- 

## How to download output files

After you build/run maps, download the output files.

## About this task

---

To download the output file, go to **Files** in the Design Server UI and download the desired file using these steps:

## Procedure

---

1. Click the output Card Settings you want to download.
  2. From the Command text box, copy the name of the file that you want to download. Make certain that you copy only the file name.  
For example, if the Command text box displays:  
  
data/ach\_contested\_dishonored\_return.out  
  
copy only:  
  
ach\_contested\_dishonored\_return.out
  3. From File, click the New icon to open the New File dialog box.
  4. Paste the file name in the Name field.
  5. Click the Folder field drop down and select data.
  6. Click Ok.
  7. Build and re-run the map.
  8. Go to File and select Download to view the data.
- 

## IBM Sterling B2B Integrator examples (Deprecated)

The documentation provided here describes how to use the IBM Sterling B2B Integrator examples.

- [IBM Sterling B2B Integrator examples overview](#)  
In IBM Sterling B2B Integrator, a *service* is a resource that you can configure to carry out an activity.
  - [Directory structure of the IBM Sterling B2B Integrator examples](#)  
The SWIFT component examples for IBM Sterling B2B Integrator include transformation maps, corresponding type trees, test data, and exported IBM Sterling B2B Integrator artifacts.
  - [IBM Sterling B2B Integrator Generic Envelope service MT example](#)  
The Generic Envelope example demonstrates how to transform a proprietary single credit transfer positional (flat) file into MT103 STP messages.
  - [IBM Sterling B2B Integrator Generic Deenvelope service MT example](#)  
The Generic Deenvelope example demonstrates how to transform a MT103 STP message into a single credit transfer positional (flat) file.
  - [IBM Sterling B2B Integrator Generic Envelope service MX example](#)
  - [IBM Sterling B2B Integrator Generic Deenvelope MX example](#)  
The Generic Deenvelope service MX example demonstrates how to transform an XML SWIFT MX pacs.008.nnn.nn document to a proprietary customer credit transfer positional (flat) file.
  - [Design considerations for the MT and MX Generic Envelope and Deenvelope examples](#)  
This documentation describes design features and concepts that were used in building the example transformation maps invoked by the Generic Deenvelope and Generic Envelope services.
  - [Deploying and executing the MT and MX Generic Envelope and Deenvelope examples](#)  
These procedures describe how to run the Generic Deenvelope and Generic Envelope services examples from the IBM Transformation Extender and from the IBM Sterling B2B Integrator.
- 

## IBM Sterling B2B Integrator examples overview

In IBM Sterling B2B Integrator, a *service* is a resource that you can configure to carry out an activity.

A typical activity scenario is one where SWIFT data must be sent to a trading partner. In preparation for this, the data must be enveloped in order to provide identifying batch and interchange data. The SWIFT Envelope business process calls Generic Envelope to provide these enveloping services, while the SWIFT Deenvelope business process calls Generic Deenvelope to provide de-enveloping services.

A document envelope consists of control information that enables organizations to effectively exchange documents. The document envelope takes the original document, assigns a control number, and packages header and trailer information with it, prior to submitting it to a trading partner. Creating document envelopes is necessary if you wish to do electronic exchange of data with your trading partners.

The SWIFT envelopes can be configured to specify Transformation Extender maps for any data transformation requirements. These maps are invoked by the Generic Envelope and Deenvelope services.

The pack examples for IBM Sterling B2B Integrator demonstrate, on both SWIFT MT and MX formats, the following:

- Creation of fully enveloped SWIFT outbound messages.
- Successfully process inbound SWIFT messages.
- Use of Transformation Extender maps for data transformation.
- Use of SWIFT compliance checking that is provided by the Pack for Financial Payments.

Transformation Extender maps can run within the IBM Sterling B2B Integrator services on some, but not all, versions of the Transformation Extender base product. See the product release notes for those versions of the base product that support IBM Sterling B2B Integrator.

## Directory structure of the IBM Sterling B2B Integrator examples

The SWIFT component examples for IBM Sterling B2B Integrator include transformation maps, corresponding type trees, test data, and exported IBM Sterling B2B Integrator artifacts.

The IBM Sterling B2B Integrator artifacts include trading partner envelopes, and business processes, which are relevant when executing the Transformation Extender map within IBM Sterling B2B Integrator.

The table shows the directory structure used for processing SWIFT MT data, using both IBM Sterling B2B Integrator Generic Envelope and Generic Deenvelope services.

Header	Header
mt/examples/sterling	
mt/examples/sterling/services	
mt/examples/sterling/services/generic_deenvelope	
mt/examples/sterling/services/generic_deenvelope/data	Placeholder for input test data for the Generic De-envelope service
mt/examples/sterling/services/generic_deenvelope/maps	Placeholder for the SWIFT MT transformation map invoked by SI's Generic De-envelope service
mt/examples/sterling/services/generic_deenvelope/platform_support	Placeholder for IBM Sterling B2B Integrator export (.xml) files
mt/examples/sterling/services/generic_envelope	Placeholder for the SWIFT MT transformation scenario using SI's Generic Envelope service
mt/examples/sterling/services/generic_envelope/data	Placeholder for input test data for the Generic Envelope service
mt/examples/sterling/services/generic_envelope/maps	Placeholder for the SWIFT MT transformation map invoked by SI's Generic Envelope service
mt/examples/sterling/services/generic_envelope/platform_support	Placeholder for IBM Sterling B2B Integrator export (.xml) files
mt/examples/sterling/services/trees	Placeholder for type trees used in both example scenarios

The table shows the directory structure used for processing SWIFT MX data using both IBM Sterling B2B Integrator Generic Envelope, and Generic Deenvelope services.

Directory structure	Description
mx/examples/sterling	
mx/examples/sterling/services	
mx/examples/sterling/services/generic_deenvelope	
mx/examples/sterling/services/generic_deenvelope/data	Placeholder for input test data for the Generic De-envelope service
mx/examples/sterling/services/generic_deenvelope/maps	Placeholder for the SWIFT MX transformation map invoked by SI's Generic Deenvelope service
mx/examples/sterling/services/generic_deenvelope/platform_support	Placeholder for IBM Sterling B2B Integrator export (.xml) files
mx/examples/sterling/services/generic_envelope	Placeholder for the transformation examples used in SI's Generic Envelope service
mx/examples/sterling/services/generic_envelope/data	Placeholder for input test data for the Generic Envelope service
mx/examples/sterling/services/generic_envelope/maps	Placeholder for the SWIFT MX transformation map invoked by SI's Generic Envelope service
mx/examples/sterling/services/generic_envelope/platform_support	Placeholder for IBM Sterling B2B Integrator export (.xml) files
mx/examples/sterling/services/trees	Placeholder for type trees used in both example scenarios

## IBM Sterling B2B Integrator Generic Envelope service MT example

The Generic Envelope example demonstrates how to transform a proprietary single credit transfer positional (flat) file into MT103 STP messages.

The positional file has a 'one line' per record format and terminates with a newLine. Each record consists of data elements which are delimited by horizontal tabs HEX"09". The first record in the file is reserved as the header record, and each record is separated by a 45 character broken line. Each data element within a record was designed to match corresponding SWIFT FIN MT tag fields that makes up a MT103 STP message body.

The example file consists of three single credit transfer records.

The Generic Envelope service MT example files install into the following directory:

<install\_dir>/packs/financial\_payments\_vn.n.n.n/swift/mt/examples/sterling/services/generic\_envelope

The source for this type of map is the credit\_transfer\_to\_mt103+, and the map is called crdt\_xfer\_to\_mt103\_plus.

- Input 1 (single\_credit\_transfer\_flat) – This card is used to represent the credit transfer positional data file.
- Output 1 (mt103\_plus\_block4) – This card retrieves all data elements that match corresponding SWIFT FIN MT tag fields that make up a MT103 STP message body.

The resulting MT103 STP message from the translation map is limited to the block 4 message body and does not include envelopes. The envelopes, for example, block 1, block 2 and block 3, are populated as part of the Generic Envelope service.

## IBM Sterling B2B Integrator Generic Deenvelope service MT example

The Generic Deenvelope example demonstrates how to transform a MT103 STP message into a single credit transfer positional (flat) file.

The positional file has a 'one line' per record format and terminates with a newLine. Each record consists of data elements which are delimited by horizontal tabs HEX"09". The first record in the file is reserved as the header record, and each record is separated by a 45 character broken line. Each data element within a record was designed to match corresponding SWIFT FIN MT tag fields that make up a MT103 STP message body.

The Generic Deenvelope service MT example files install into the following directory:

<install\_dir>/packs/financial\_payments\_vn.n.n.n/swift/mt/examples/sterling/services/generic\_deenvelope.

The source for this type of map is the mt103+\_to\_credit\_transfer, and the map is called mt103\_plus\_to\_crdt\_xfer\_flat.

- Input 1 (mt103\_plus) – This card is used to represent the SWIFT MT103 STP data file.
- Output 1 (single\_credit\_transfer\_flat) – This card retrieves all data elements that match corresponding positional credit transfer element fields.

## IBM Sterling B2B Integrator Generic Envelope service MX example

The Generic Envelope service MX example demonstrates how to transform a proprietary customer credit transfer positional (flat) file into an XML SWIFT MX pacs.008.nnn.nn document.

The positional file follows a 'tag value' format, and terminates with a newLine. Each tag and value is separated by a colon ':', example MsgId:XXXX123456789. Each record starts with a tag MsgId and ends with a tag CdtrTwnNm. Each data element within a record was designed to match corresponding SWIFT MX pacs.008.nnn.nn element fields.

The example file consists of three customer credit transfer records. The Generic Envelope service MX example files install into the following directory:

<install\_dir>/packs/financial\_payments\_vn.n.n.n/swift/mx/examples/sterling/services/generic\_envelope. The source for this type of map is the credit\_transfer\_to\_mx\_pacs8 , and the map is called cstmr\_cdtrrf\_to\_pacs812.

- Input 1 (cstmrCrdtTrfFlat) – This card is used to represent the credit transfer positional data file.
- Output 1 (pacs\_008\_nnn\_nn) – This card retrieves all data elements that match corresponding SWIFT MX tag fields that make up a pacs.008.nnn.nn document.

The resulting message from the translation map is limited to the SWIFT MX pacs.008.nnn.nn document payload and does not include envelopes. The envelopes, for example, SWIFT Alliance Access Protocol Data Units (PDUs) and DataPDUs are populated as part of the Generic Envelope service.

## IBM Sterling B2B Integrator Generic Deenvelope MX example

The Generic Deenvelope service MX example demonstrates how to transform an XML SWIFT MX pacs.008.nnn.nn document to a proprietary customer credit transfer positional (flat) file.

The positional file follows a 'tag value' format, and terminates with a newLine. Each tag and value is separated by a colon ':', example MsgId:XXXX123456789. Each record starts with a tag MsgId and ends with a tag CdtrTwnNm. Each data element within a record was designed to match corresponding SWIFT MX pacs.008.nnn.nn element fields.

The Generic Deenvelope service MX example files install into the following directory:

<install\_dir>/packs/financial\_payments\_vn.n.n.n/swift/mx/examples/sterling/services/generic\_deenvelope.

The source for this type of map is the mx\_pacs8\_to\_credit\_transfer , and the map is called pacs812\_to\_cstmr\_cdtrrf

- Input 1 (pacs\_008\_nnn\_nn) – This card is used to represent the SWIFT MX pacs.008.nnn.nn data file.
- Output 1 (cstmrCrdtTrfFlat) – This card retrieves all data elements that match corresponding positional credit transfer element fields.

Note: This example IBM Transformation Extender map expects a fully enveloped MX pacs.008.nnn.nn message as input when executed within the IBM Sterling B2B Integrator Generic Deenvelope service. The IBM Sterling B2B Integrator Generic Deenvelope service only passes the actual MX instance document without the SWIFT Alliance Access (SAA) envelopes. For example, the SAA PDU and DataPDU information are stripped off by the Generic Deenvelope service, and only the MX payload pacs.008.nnn.nn instance document is passed to the example IBM Transformation Extender map.

## Design considerations for the MT and MX Generic Envelope and Deenvelope examples

This documentation describes design features and concepts that were used in building the example transformation maps invoked by the Generic Deenvelope and Generic Envelope services.

- **IBM Sterling B2B Integrator data harness**

In some cases, IBM Transformation Extender maps must share information with the IBM Sterling B2B Integrator services and other IBM Sterling B2B Integrator processes.

- **Thread safety**

The Generic Deenvelope service and the Generic Envelope service example maps are 'thread-safe' when they are deployed in the IBM Sterling B2B Integrator environment.

- **Specific map and type tree settings**

Important map and type tree settings that are used in the examples are of particular importance for running with the MT and MX Generic Envelope service.

---

## IBM Sterling B2B Integrator data harness

In some cases, IBM Transformation Extender maps must share information with the IBM Sterling B2B Integrator services and other IBM Sterling B2B Integrator processes.

IBM Sterling B2B Integrator

For example, the maps can perform one of the following activities:

- Store information about the data, for example, a document ID or other key field that can be used by Sterling B2B Integrator to track the data.
- Get information from Sterling B2B Integrator tables, such as trading partner information, or code list table data. The maps read and write this information to Sterling B2B Integrator database tables using the Sterling B2B Integrator adapter. This is referred to as the Sterling B2B Integrator Data Harness which provides access to these tables.

The Sterling B2B Integrator adapter is used to get and to set certain values that are maintained in the Sterling B2B Integrator adapter database tables.

- The **setProcessData** call is used to set values such as reference ids in the **processData** table. This value can be used by other services in the Business Process flow.
- The **setCorrelation** call is used to set the values such as reference ids in the correlation data table. This value can be used to help search for documents processed by the map.

The Generic Deenvelope service and Generic Envelope service example maps use PUT calls to the Sterling B2B Integrator adapter in order to set the transaction reference number for MT scenarios, and instruction id, for MX scenarios in the Sterling B2B Integrator database called **ProcessData** and **Correlation** tables.

---

## Thread safety

The Generic Deenvelope service and the Generic Envelope service example maps are 'thread-safe' when they are deployed in the IBM Sterling B2B Integrator environment.

A thread-safe map can run in a multi-threaded environment. One of the attributes a map needs in order to run in a multi-threaded environment is that when multiple occurrences of the map are instantiated, none of the resources used by an individual instance of the map are used by another instance of the map.

For example, the map work files, audit logs, backup files, and trace logs are individually named; that is, they are unique within each instance of the map.

For more information about thread safety, including how you can verify that the map meets at least the basic thread-safe checks, see the Transformation Extender Map Designer documentation.

---

## Specific map and type tree settings

Important map and type tree settings that are used in the examples are of particular importance for running with the MT and MX Generic Envelope service.

- **Burst**

The *burst* concept deals with the scope of the data that is processed in one run of the map. Generic Envelope service example maps are set to execute in burst mode.

- **Audit settings**

Audit settings enable a map to generate data audit information that can be used by the Generic service in business process reports.

- **Restart**

The use of a restart flag allows a IBM Transformation Extender map to continue to attempt to process remaining transactions after an invalid transaction is found in the input file.

- **Custom MT type tree**

On the Generic Envelope service MT example, the Generic Envelope service expects that the resulting file of the transformation map contains only SWIFT MT block 4 content, without any header or trailer information.

---

## Burst

The *burst* concept deals with the scope of the data that is processed in one run of the map. Generic Envelope service example maps are set to execute in burst mode.

The Generic Envelope service runs a portion of the input with each iteration of the map. This is expected behavior in several IBM Sterling B2B Integrator services.

In the Generic Envelope service example, in the map input card, the FetchAs property is set to Burst with a FetchUnit of 1. This allows the IBM Transformation Extender map to process a single transaction at a time, as expected by the Generic Envelope service.

---

## Audit settings

Audit settings enable a map to generate data audit information that can be used by the Generic service in business process reports.

In the map settings, the MapAudit switch is set to ON, the BurstAudit and SummaryAudit settings are set to Always, and the AuditLocation is set to Memory.

In the Generic Deenvelope MT example, the Data Audit Settings for the map are set as follows:

Audit	Message MT103_Plus Category 1 Text Block:mt103_plus
Track	Occurrence
Details	Occurrence
Item Data	Occurrence

In the Generic Envelope service MT example, the Data Audit Settings for the map are set as follows:

Audit	Record:single_credit_transfer_flat
Track	Occurrence
Details	Occurrence
Item Data	Occurrence

In the Generic Deenvelope MX example, the Data Audit Settings for the map are set as follows:

Audit	FIToFICstmrCdtTrfsequence:packs_008_001_02
Track	Occurrence
Details	Occurrence
Item Data	Occurrence

In the Generic Envelope service MX example, the Data Audit Settings for the map are set as follows:

Audit	Record:cstrmCrdtTrfFlat
Track	Occurrence
Details	Occurrence
Item Data	Occurrence

## Restart

The use of a restart flag allows a IBM Transformation Extender map to continue to attempt to process remaining transactions after an invalid transaction is found in the input file.

### Generic Deenvelope service MT example

In the Generic Deenvelope service MT example, the swift\_mt103\_plus\_nohdrs.mtt type tree File group specifies the Restart flag on the following component:

- Message MT103\_Plus Category 1 TextBlock:mt103\_plus

### Generic Envelope service MT example

In the Generic Envelope service MT example, the single\_credit\_transfer\_flat.mtt type tree in the File group specifies the Restart flag on the following component:

- Record:single\_credit\_transfer\_flat

### Generic Deenvelope service MX example

In the Generic Deenvelope MX service example, the pacs.008.nnn.nn.xsd schema File group specifies the Restart flag on the following components:

- FIToFICstmrCdtTrf:sequence:pacs\_008\_nnn\_nn

### Generic Envelope service MX example

In the generic envelope service MX example, the cstmr\_cdttrf\_flat.mtt type tree in the File group specifies the Restart flag on the following component:

- Record:cstmrCrdtTrfFlat

## Custom MT type tree

On the Generic Envelope service MT example, the Generic Envelope service expects that the resulting file of the transformation map contains only SWIFT MT block 4 content, without any header or trailer information.

For example, blocks 1, 2, and ,3 are not populated by the map. This information is provided by the Generic Envelope service. The Transformation Extender Pack for Financial Payments ships type trees that represent SWIFT MT message syntax. That includes SWIFT MT headers and trailers, such as block 1, 2, 3, 4 and 5. This documentation provides the steps needed to build a custom type tree based from the provided type trees without any header or trailer information.

The type tree used in the examples, swift\_mt103\_plus\_nohdrs.mtt, is a subset of the swift\_iso7775\_ccyy.mtt type tree that is included with the SWIFT component of the Transformation Extender Pack for Financial Payments . This type tree is shared between the Generic Deenvelope service and the Generic Envelope service examples.

In the above file name, ccyy indicates a four-digit year value; for example, 2012.

- [Building a custom MT type tree](#)

This procedure describes how to create a custom MT103 STP type tree.

## Building a custom MT type tree

This procedure describes how to create a custom MT103 STP type tree.

### About this task

Perform the following steps to create a custom MT103 STP type tree:

### Procedure

1. Open the **swift\_iso7775\_ccyy.mtt** type tree that installs with the SWIFT component of the IBM Transformation Extender Pack for Financial Payments.
2. Select **Save As** and define a new type tree named **swift\_mt103\_plus\_nohdrs.mtt**.
3. Under SWIFT > Batch, remove all groups except **Category1**.
4. Under SWIFT > Batch > Category1 group, remove all groups except for MT 103.
5. Under the SWIFT > Block > Text, remove all categories except for **Category1**.
6. Under the SWIFT > Block > Text > Category1 category, remove all category groups except for MT103\_Plus.
7. Under the SWIFT > Block > Text > Category1 > MT103\_Plus category, create a copy of the **Message** group called **MessageContentOnly** in the same **MT103\_Plus** category
8. On the SWIFT > Block > Text > Category1 > MT103\_Plus > MessageContentOnly group, update the following **Type Syntax** properties to **None**:
  - a. Initiator
  - b. Terminator
9. Under the SWIFT > Block > Text > Category1 > MT103\_Plus category, create a new **block4ContentOnly** group with the following properties:
  - a. Set the **Class** to **Group**.
  - b. Set **Group Subclass** to **Sequence**
  - c. Set Group Subclass > Format to **Implicit**.
  - d. Include the **MessageContentOnly** group as a component of **block4ContentOnly** with a range of (1:s).
10. Under the SWIFT > Message category, remove all **Category** groups except for **Category1**.
11. Under the SWIFT > Message > Category1 group, remove the all groups except for **MT103**.
12. Under the SWIFT > Message > Category1 > MT103 group, remove **Core** group.
13. Under the SWIFT > Message > Input category, remove all groups except for **Category1**.
14. Under the SWIFT > Message > Input > Category1 group, remove all groups except for **MT103**.
15. Under SWIFT > Message > Input > Category1 > MT103 group, remove the **Core** group.
16. Under SWIFT > Message > Output category, remove all groups except for **Category1**.
17. Under the SWIFT > Message > Output > Category1 group, remove all groups except for **MT103**.
18. Under the SWIFT > Message > Output > Category1 > MT103 group, remove the **Core** group.
19. Under the SWIFT > Message > LiveMsgs group remove all components except for **MT103 Category1**.
20. Under the SWIFT > Subfield > Block category, remove the **Tag 270** and **Tag 280** groups.
21. Analyze and save the type tree. Warnings will occur. Errors will not exist.

## Deploying and executing the MT and MX Generic Envelope and Deenvelope examples

These procedures describe how to run the Generic Deenvelope and Generic Envelope services examples from the IBM Transformation Extender and from the IBM Sterling B2B Integrator.

The example maps call the IBM Sterling B2B Integrator adapter to write values to IBM Sterling B2B Integrator tables. Because of this, the map fails if you attempt to run the map outside of IBM Sterling B2B Integrator.

The map can also be tested by using the Run on Sterling B2B Integrator option in the Design Studio. Test the map to make certain that the map is built to run properly on IBM Sterling B2B Integrator even though the Generic Envelope and Deenvelope services are not invoked. The Design Studio preferences must be configured to connect to the IBM Sterling B2B Integrator. See the Design Studio documentation for more information.

- [Deploying the example maps to IBM Sterling B2B Integrator](#)  
The map is deployed using either IBM Transformation Extender Design Studio or the IBM Sterling B2B Integrator dashboard.
- [Running the MT examples](#)  
This documentation describes how to configure and run the Generic Deenvelope service and Generic Envelope service examples on MT.
- [Running the MX examples](#)  
The documentation provided here describes how to configure and run the Generic Deenvelope service and Generic Envelope service examples on MX.
- [Expected results](#)

## Deploying the example maps to IBM Sterling B2B Integrator

The map is deployed using either IBM Transformation Extender Design Studio or the IBM Sterling B2B Integrator dashboard.

- [Deploying the map using Transformation Extender](#)  
The documentation provided here describes how to deploy the map to IBM Sterling B2B Integrator using IBM Transformation Extender Design Studio.
- [Deploying the map using IBM Sterling B2B Integrator](#)  
The following procedures describe an alternate method of checking in the map by using the IBM Sterling B2B Integrator dashboard.

## Deploying the map using Transformation Extender

The documentation provided here describes how to deploy the map to IBM Sterling B2B Integrator using IBM Transformation Extender Design Studio.

## Procedure

---

1. Configure IBM Transformation Extender Design Studio preferences before you attempt to connect to the IBM Sterling B2B Integrator server. See the IBM Transformation Extender Design Studio documentation for more information.
  2. Open the source map in IBM Transformation Extender Design Studio.
  3. Right click on the map, then choose Deploy to Sterling B2B Integrator. Enter your ID and password if you are prompted.
- 

## Deploying the map using IBM Sterling B2B Integrator

The following procedures describe an alternate method of checking in the map by using the IBM Sterling B2B Integrator dashboard.

## Procedure

---

1. Compile the source maps for the correct platform on which the IBM Sterling B2B Integrator is installed.
  2. From the IBM Sterling B2B Integrator dashboard, go to Deployment > Maps, then select the **Check in new Map from Map Editor** option.  
If the map has already been deployed to the IBM Sterling B2B Integrator, you can update it by first searching for the map, and then using the Map Source Manager to check out the map and then check in the new version.
- 

## Running the MT examples

This documentation describes how to configure and run the Generic Deenvelope service and Generic Envelope service examples on MT.

1. Set the SYSTEM environment variables to contain the installation directory.
  2. Configure the jvalccyy.prop file. See the documentation that is titled [How to configure the JVC](#)
  3. Copy and install the SWIFT JAR files from IBM Transformation Extender to IBM Sterling B2B Integrator. See [Deploying JVC compliance checking on IBM Sterling B2B Integrator](#). Follow the instructions for the install3rdParty.cmd file.
  4. Create a customer\_overrides\_properties.cmd file to include translator statements for IBM Sterling B2B Integrator. Follow the steps in [Deploying JVC compliance checking on IBM Sterling B2B Integrator](#).
  5. Build the maps for Windows by using the build\_deploy\_jvc.bat file. The build\_deploy\_jvc.bat file creates the directory, deployment\_jvc, under the mt/jvc/maps directory. Copy the files to the correct Windows folders.
  6. Import IBM Sterling B2B Integrator artifacts. From the IBM Sterling B2B Integrator dashboard, import one of the following files from the platform\_support directory of the example: ExportSWIFTmtDeenvSIOBJECTS.xml or ExportSWIFTmtEnvSIOBJECTS.xml, then proceed as described here:
    - a. Log in to the IBM Sterling B2B Integrator dashboard.
    - b. From the Administration menu, expand the Deployment and Resource Manager sections, and then select the Import/Export option.
    - c. Choose the Import Resources option.
    - d. Select a file name:
      - For the Generic Deenvelope services example for MT, select the ExportSWIFTmtDeenvSIOBJECTS.xml file name.
      - For the Generic Envelope services example for MT, select the ExportSWIFTmtEnvSIOBJECTS.xml file name.
    - e. Enter the default SI B2Bi account passphrase and then check the Import All Resources check box.
- [Deploying the example maps to IBM Sterling B2B Integrator](#)  
The example maps are deployed using either IBM Transformation Extender Design Studio, or through use of the IBM Sterling B2B Integrator dashboard.
  - [Deploying maps from within IBM Transformation Extender](#)  
This documentation describes how to deploy the example maps to IBM Sterling B2B Integrator using Design Studio.
  - [Deploying maps using IBM Sterling B2B Integrator](#)
  - [How to run MT examples from IBM Sterling B2B Integrator](#)  
The SWIFT MT examples can be run from the Generic Deenvelope service and the Generic Envelope service.
- 

## Deploying the example maps to IBM Sterling B2B Integrator

The example maps are deployed using either IBM Transformation Extender Design Studio, or through use of the IBM Sterling B2B Integrator dashboard.

## Deploying maps from within IBM Transformation ExtenderIBM®

This documentation describes how to deploy the example maps to IBM Sterling B2B Integrator using Design Studio.

## Procedure

---

1. Configure Design Studio preferences before you attempt to connect to the IBM Sterling B2B Integrator server.
  2. Open the source map in Design Studio.
    - a. If you are running the Generic Deenvelope service example, open the mt103\_plus\_to\_crdt\_xfer\_flat file from source map, mt103+\_to\_credit\_transfer .
    - b. If you are running the Generic Envelope service example, open the crdt\_xfer\_to\_mt103\_plus file from source map credit\_transfer\_to\_mt103+ .
  3. Right click on the map, then choose Deploy to Sterling B2B Integrator. Enter your ID and your password if you are prompted.
-

# Deploying maps using IBM Sterling B2B Integrator

## About this task

These procedures describe how to use the IBM Sterling B2B Integrator dashboard to check in the map.

## Procedure

1. Use IBM Transformation Extender to compile the source maps for the correct platform on which IBM Sterling B2B Integrator is installed.
  - a. Open the source map in the IBM Transformation Extender Design Studio.
    - If you are running the Generic Deenvelope service example, open the mt103\_plus\_to\_crdt\_xfer\_flat file from the mt103+\_to\_credit\_transfer source map.
    - If you are running the Generic Envelope service example, open the crdt\_xfer\_to\_mt103\_plus file from the credit\_transfer\_to\_mt103+ source map.
2. From the IBM Sterling B2B Integrator dashboard, go to Deployment > Maps, then select the Check in a new Map from the Map Editor option.  
Note: If the map has already been deployed to the IBM Sterling B2B Integrator you can update it by first searching for the map, and then using the Map Source Manager to check out the map. Then check in the new version.

# How to run MT examples from IBM Sterling B2B Integrator

The SWIFT MT examples can be run from the Generic Deenvelope service and the Generic Envelope service.

## About this task

These procedures describe how to run the deenvelope and envelope example maps from the IBM Sterling B2B Integrator dashboard.

## Procedure

1. From the Administration menu, go to the Business Process manager page.
2. Search for a Business Process:
  - If you are running the IBM Sterling B2B Integrator Generic Deenvelope service example, search for the EDIDeenvelope Business Process.
  - If you are running the IBM Sterling B2B Integrator Generic Envelope service example, search for the SWIFT\_MT\_103STP\_EnvelopeUsingWTX Business Process.
3. Select the execution manager for the Business Process.

# Running the MX examples

The documentation provided here describes how to configure and run the Generic Deenvelope service and Generic Envelope service examples on MX.

To run the examples, you must first perform the steps listed here.

1. Set the SYSTEM environment variables to contain the IBM Transformation Extender installation directory.
  2. Deploy the XML schemas by using the InstallService.cmd file. See the procedures in [Deploying the XML schemas to IBM Sterling B2B Integrator \(Deprecated\)](#) to accomplish this activity.
  3. Create a customer\_overrides\_properties.cmd file to include translator statements for IBM Sterling B2B Integrator. Follow the procedures provided in [Deploying MX compliance checking components](#).
  4. Build the maps for Windows using the build\_deploy\_mx\_validation.bat file. The build\_deploy\_mx\_validation.bat file creates the deployment\_mx\_validation directory under the mx/maps directory. Copy the files to the proper Windows folders.  
Note: The swiftval map must be deployed to IBM Sterling B2B Integrator along with the example maps.
  5. Import Sterling B2B Integrator artifacts (for MX). From the Sterling B2B Integrator dashboard, import one of the following files from the platform\_support directory of the example: ExportSWIFTmxDeenvSIOBJECTS.xml and ExportSWIFTmxEnvSIOBJECTS.xml.
    - a. Login to the Sterling B2B Integrator dashboard.
    - b. From the Administration menu, expand the Deployment and Resource Manager sections, then select the Import/Export option.
    - c. Choose the Import Resources option.
      - For the Generic Deenvelope services example for MX, select the file name: ExportSWIFTmxDeenvSIOBJECTS.xml
      - For the Generic Envelope services example for MX, select the file name: ExportSWIFTmxEnvSIOBJECTS.xml
  6. Enter the default SI B2Bi account passphrase and then check the Import All Resources check box.
- [Importing IBM Sterling B2B Integrator](#)  
The map is deployed by using either IBM Transformation Extender Design Studio or the IBM Sterling B2B Integrator dashboard.
  - [Deploying the map from within IBM Transformation Extender](#)  
These steps describe how to deploy the map to IBM Sterling B2B Integrator using IBM Transformation Extender Design Studio.
  - [Deploying the maps using IBM Sterling B2B Integrator](#)  
The following procedures describe the method of checking in the map by using the IBM Sterling B2B Integrator dashboard.
  - [How to run the MX examples from IBM Sterling B2B Integrator](#)  
These procedures describe how to run the Generic Deenvelope service and the Generic Envelope service example maps from the IBM Sterling B2B Integrator dashboard:

# Importing IBM Sterling B2B Integrator

The map is deployed by using either IBM Transformation Extender Design Studio or the IBM Sterling B2B Integrator dashboard.

## Deploying the map from within IBM Transformation Extender

These steps describe how to deploy the map to IBM Sterling B2B Integrator using IBM Transformation Extender Design Studio.

### Procedure

1. Configure Design Studio preferences before you attempt to connect to the IBM Sterling B2B Integratorserver. See the Design Studio documentation for more information.
2. Open the source map in Design Studio.
  - If you are running the Generic Deenvelope service example, open the pacs812\_to\_cstmr\_cdtrrf to pacs8nn\_to\_cstmr\_cdtrrf file from source map: mx\_pacs8\_to\_credit\_transfer .
  - If you are running the Generic Envelope service example, open the cstmr\_cdtrrf\_to\_pacs812 to cstmr\_cdtrrf\_to\_pacs8nn file from source map: credit\_transfer\_to\_mx\_pacs8
3. Right click on the map, then choose Deploy to Sterling B2B Integrator. Enter your id and your password if you are prompted.
4. Repeat steps 2 and 3 to deploy the map for MX validation, swiftval file from source map: mx\_validation\_framework .

## Deploying the maps using IBM Sterling B2B Integrator

The following procedures describe the method of checking in the map by using the IBM Sterling B2B Integrator dashboard.

### Procedure

1. Use the IBM Transformation Extender Design Studio to compile the source maps for the correct platform on which IBM Sterling B2B Integrator is installed.
2. Open the source map in IBM Transformation Extender Design Studio.
  - If you are running the Generic Deenvelope service example, open the pacs812\_to\_cstmr\_cdtrrf to pacs8nn\_to\_cstmr\_cdtrrf file from source map: mx\_pacs8\_to\_credit\_transfer
  - If you are running the Generic Envelope service example, open the cstmr\_cdtrrf\_to\_pacs812 to cstmr\_cdtrrf\_to\_pacs8nn file from source map: credit\_transfer\_to\_mx\_pacs8
3. From the IBM Sterling B2B Integrator dashboard, go to Deployment > Maps, then select the Check in new Map from Map Editor option. Compiled maps will include: pacs812\_to\_cstmr\_cdtrrf to pacs8nn\_to\_cstmr\_cdtrrf, cstmr\_cdtrrf\_to\_pacs812 to cstmr\_cdtrrf\_to\_pacs8nn and swiftval (compiled from running the build\_deploy\_mx\_validation.bat) If the map has already been deployed to the IBM Sterling B2B Integrator, you can update it by first searching for the map, and then using the Map Source Manager to check out the map and then check in the new version.

## How to run the MX examples from IBM Sterling B2B Integrator

These procedures describe how to run the Generic Deenvelope service and the Generic Envelope service example maps from the IBM Sterling B2B Integrator dashboard:

### Procedure

1. Import IBM Sterling B2B Integrator artifacts. See [Importing IBM Sterling B2B Integrator](#) for instructions.
2. Deploy the example maps. See [Deploying the example maps to IBM Sterling B2B Integrator](#) for instructions.
3. From the Administration menu, go to the Business Process Manager page.
4. Search for a business process:
  - If you are running the IBM Sterling B2B Integrator Generic Deenvelope service example, search for the EDIDeenvlope Business Process.
  - If you are running the IBM Sterling B2B Integrator Generic Envelope service example, search for the SWIFT\_MX\_PACS8\_EnvelopeUsingWTX Business Process.
5. Select the execution manager for the Business Process, then select the Execute option.
6. On the Local Desktop file name, browse, and then select, the appropriate input file from the data folder.
  - If you are running the Generic Service Deenvelope example, select the pacs\_008\_nnn\_nn\_pdu.in input file.
  - If you are running the Generic Service Envelope example, select the cstmr\_cdtrrf\_flat.dat input file.Note: If the Sterling server is running on UNIX, use cstmr\_cdtrrf\_flat\_LNX.dat as the input file.
7. Select the Go option.
8. The Execute Business Process results show a status of Success for all of the steps in the business process.

You can view more details, such as Status Report, document, and Instance Data by going to this location:

Business Process > Monitor > Current Process

The translated output of the examples can be viewed by going to this location:

Business > Monitor > Current Documents

The document information shows the translated output of the Generic Deenvelope service and the Generic Envelope service. The Document Instance Data shows the Process data values that are set by the service, including the instrID values set by the map.

## Expected results

The expected results vary, depending upon which example is run. Results for the MT generic services de-envelope and envelope services, as well as for the MX generic services de-envelope and envelope service are described in the following documentation.

- [\*\*Generic Deenvelope service MT example: expected results\*\*](#)  
Expect these results from the Generic Deenvelope service MT example:
  - [\*\*Generic Envelope service MT example: expected results\*\*](#)  
Expect the following results from the Generic Envelope service MT example:
  - [\*\*Generic Deenvelope service MX example, expected results\*\*](#)  
Expect the following results from the Generic Deenvelope service MX example.
  - [\*\*Generic Envelope service MX example: expected results\*\*](#)  
Expect the following results from the Generic Envelope service MX example:
- 

## Generic Deenvelope service MT example: expected results

Expect these results from the Generic Deenvelope service MT example:

- When analyzing the swift\_mt103\_plus\_nohdrs.mtt type tree, warnings are expected. Errors are not expected.
  - The translation map generates one credit transfer transaction.
  - The result of the translation map is the same format as the creditTransfer\_flat.dat input file used for the sample IBM Transformation Extender map for the IBM Sterling B2B Integrator Generic Envelope service for MT.
  - The entry for the transaction reference number, transRef, exists in both the ProcessData and correlation tables.
- 

## Generic Envelope service MT example: expected results

Expect the following results from the Generic Envelope service MT example:

- When analyzing the single\_credit\_transfer\_flat.mtt type tree, no error or warning is expected.
  - The translation map generates three MT103+ SWIFT FIN messages.
  - When a compliance check is run against the result of the translation map, two valid, and one invalid (first of the three transactions), MT103+ SWIFT FIN messages are generated.
  - The failure is due to the absence of required tag 121 (UETR) on block 3.
  - The entry for transaction reference number, transRef, exists in both the ProcessData and Correlation tables.
- 

## Generic Deenvelope service MX example, expected results

Expect the following results from the Generic Deenvelope service MX example.

- The translation map generates three customer credit transfer transactions.
  - The result of the translation map is the same format as the cstmr\_cdttrf\_flat.dat input file used for the sample Transformation Extender map for the IBM Sterling B2B Integrator Generic Envelope service for MX.
  - Compliance check performed against the MX document prior to execution of the translation map indicates two valid and one invalid (first of the three transactions), within the SWIFT MX message.
  - The failure is due to an invalid currency code on element IntrBkSttlmAmt currency attribute.
  - The entry for transaction instruction identification, instrID, exists in both the ProcessData and Correlation tables.
- 

## Generic Envelope service MX example: expected results

Expect the following results from the Generic Envelope service MX example:

- The entry for the transaction instruction identification (instrID) exists in both the ProcessData and Correlation tables.
  - When analyzing the cstmr\_cdttrf\_flat.mtt type tree, no error or warning is expected.
  - The translation map generates three FIToFICstmrCdtTrf (pacs.008.nnn.nn) SWIFT MX documents.
- 

## Universal Confirmation examples

Examples in Universal Confirmation component:

- Universal Confirmation API Example
  - Universal Confirmation CSV Example
  - Universal Confirmation FIN Example
  - [\*\*Configuration of SWIFT Universal Confirmation examples in Design Studio\*\*](#)  
This topic documents how to configure the Universal Confirmation examples in Design Studio.
  - [\*\*Configuration of SWIFT Universal Confirmation examples in Design Server\*\*](#)
- 

## Configuration of SWIFT Universal Confirmation examples in Design Studio

This topic documents how to configure the Universal Confirmation examples in Design Studio.

## Before you begin

---

The jvalccyy.prop is installed in <install\_dir>/packs/financial\_payments\_yn.n.n.n/swift/mt/jvc/maps for all versions of ITX.

## Procedure

---

Move the jvalccyy.prop in the following location for ITX V.9.n and higher versions:

- Windows: <tx\_install\_dir>
- Linux: <tx\_install\_dir>/bin
- [How to run the Universal Confirmation API example in Design Studio](#)
- [How to run the Universal Confirmation CSV example in Design Studio](#)
- [How to run the Universal Confirmation FIN example in Design Studio](#)

## How to run the Universal Confirmation API example in Design Studio

---

### About this task

---

This topic describes how to run the Universal Confirmation API example in Design Studio.

## Procedure

---

1. Use the Design Studio to open and build all maps in uc\_api\_validation.mms, uc\_translation.mms, and validate\_messages.mms maps.
2. Run the map uctrxapi in uc\_translation.mms for MT103 to api translation.
3. Run the map uctrx199 in uc\_translation.mms for api to MT199 translation.

Expected results:

- a. Valid Scenarios:  
In case of successful pre-validation, translation, and post validation, following files appear in the data folder:
  - api.json: For MT103 to api translation.
  - MT199.out: For api to MT199 translation.
- b. Invalid Scenarios:  
An error file prevalidation\_failed.json file appears in the data folder in following scenarios, but not limited to:
  - Any of the pre-validation executable map \*.mmc is not found .
  - Input source file is invalid and not compliant with pre-validation rules.
  - The translation executable map is not found, or translation is failed.
  - Unsupported input provided for translation.An error file postvalidation\_failed.json file appears in the data folder in following scenarios:
  - Any of the post validation executable map \*.mmc is not found.
  - The outputfile is invalid and not compliant with post validation rules.

## How to run the Universal Confirmation CSV example in Design Studio

---

### About this task

---

This topic describes how to run the Universal Confirmation CSV example in Design Studio.

## Procedure

---

1. Use the Design Studio to open and build all maps in uc\_csv\_validation.mms, uc\_translation.mms, and validate\_messages.mms maps.
2. Run the map ucrtcsv in uc\_translation.mms for MT103 to csv translation.
3. Run the map uctrx199 in uc\_translation.mms for csv to MT199 translation.

Expected results:

- a. Valid Scenarios:  
In case of successful pre-validation, translation, and post validation, following files appear in the data folder:
  - CSV.csv: For MT103 to csv translation.
  - MT199.out: For csv to MT199 translation.
- b. Invalid Scenarios:  
An error file prevalidation\_failed.json file appears in the data folder in following scenarios, but not limited to:
  - Any of the pre-validation executable map \*.mmc is not found .
  - Input source file is invalid and not compliant with pre-validation rules.
  - The translation executable map is not found, or translation is failed.
  - Unsupported input provided for translation.An error file postvalidation\_failed.json file appears in the data folder in following scenarios:
  - Any of the post validation executable map \*.mmc is not found.
  - The outputfile is invalid and not compliant with post validation rules.

If the reference of MT103 does not match with the instruction\_reference in the translation configuration file, an error file MT103\_config\_mismatch.out appears in the data folder.

# How to run the Universal Confirmation FIN example in Design Studio

## Before you begin

Refer to [Additional configuration for MT JVC validation](#).

For Design Studio, will require administrative rights:

Copy jars to <TX\_install\_dir>/extjar.

For Design Server installation:

See [Configuration files for Design Server](#) for the steps to perform before you import <project>.zip file.

where <project>.zip can be any of the following:

- swiftUCapi.zip
- swiftUCcsv.zip
- swiftUCfin.zip

## About this task

This topic describes how to run the Universal Confirmation FIN example in Design Studio.

## Procedure

1. Use the Design Studio to open and build all maps in uc\_translation.mms, and validate\_messages.mms maps.
2. Run the map uctrx199 in uc\_translation.mms for fin to MT199 translation.
3. Check for the output file prevalidation\_failed.json, postvalidation\_failed.json, or MT199.out in the data folder.

Note: Any one of the three files will appear in the data folder.

Expected results:

- a. Valid Scenarios:  
In case of successful pre-validation, translation, and post validation, the MT199.out (For fin to MT199 translation) file appears in the data folder.

- b. Invalid Scenarios:

An error file prevalidation\_failed.json file appears in the data folder in following scenarios, but not limited to:

- Any of the pre-validation executable map \*.mmc is not found.
- Input source file is invalid and not compliant with prevalidation rules.
- The translation executable map is not found, or translation is failed.
- Unsupported input provided for translation.

An error file postvalidation\_failed.json file appears in the data folder in following scenarios:

- Any of the post validation executable map \*.mmc is not found.
- The output file is invalid and not compliant with post validation rules.

# Configuration of SWIFT Universal Confirmation examples in Design Server

## Configuration files for Design Server

The swiftMTjvcConfig.tar.gz file is contained within UIPackageImports directory.

The UIPackageImports directory is located in the following location:

<install\_dir>/packs/financial\_payments\_vn.n.n.n/UIProjectImports

Where, <install\_dir> is the location where the pack zip distro was extracted and n.n.n.n is the current version of the pack..

Use these steps to run this file:

Extract the swiftMTjvcConfig.tar.gz file.

Follow the below steps, before you import <project>.zip file.

Perform the steps in the same directory location as the Design Server is installed:

1. Create a temp directory called packs.  
Example: /home/user/packs
2. For UNIX environment, FTP (in binary mode) the swiftMTjvcConfig.tar.gz to the pack's directory defined in Step 1.
3. Untar the swiftMTjvcConfig.tar.gz file.  
Example: \$ tar -zxf swiftMTjvcConfig.tar.gz
4. For Container based Design Server installation:
  - a. Run the **jvcsetup.sh** script.  
Example: \$ ./jvcsetup.sh
  - b. The script copies the following to the ITX server:
    - jcwrap.jar

- jvalccyy.jar
- packs/swift folder and sub-folders includes the jvalccyy.prop and XML metadata.

**Optional:**

- a. Run the `cleanjvc.sh` script.  
Example: `$ ./cleanjvc.sh`

- b. The script removes the following to the ITX server:

- jcwrap.jar
- jvalccyy.jar

packs/swift folder and sub-folders includes the jvalccyy.prop and XML metadata.

5. For Native based installation:

- a. Copy the jvalccyy.jar and jcwrap.jar into the directory defined in config.yaml server.persistence.files, by default, this is set to /opt/txlibs.
- b. Restart the application running `./ITX stop` and then `./ITX start`.

6. After importing `<project>.zip` project:

- a. Copy the jvalccyy.prop file to the workdir directory under directory set in the config.yaml server.persistence.files, the default settings is /opt/txfiles/workdir.
- b. On the project, update the map rule on the validate\_message map, on output card #1 item JvalPropPath. Change the map rule from =NONE to the location of the jvalccyy.prop.

Example:

- On UNIX  
= "DS\_DATA\_DIR/workdir"
- On WINDOWS  
= "DS\_DATA\_DIR\\workdir"

where DS\_DATA\_DIR refers to the Design Server data directory, set in the config.yamlserver.persistence.files previously TX\_FILE\_DIR

7. Restart the Design Server: `./ITX stop` and `./ITX start`.

Note: To reflect any jvalccyy.prop updates, repeat step 7.

- [How to run the Universal Confirmation API example in Design Server](#)
- [How to run the Universal Confirmation CSV example in Design Server](#)
- [How to run the Universal Confirmation FIN example in Design Server](#)

## How to run the Universal Confirmation API example in Design Server

### About this task

This topic describes how to run the Universal Confirmation API in Design Server.

To configure the Design Server, see [Configuration of SWIFT Universal Confirmation examples in Design Server](#).

### Procedure

1. Import the swiftUCApi.zip file into the Design Server UI. See the release notes for import instructions.
2. Create a package for the project and build all the maps.
3. Open swiftUCApi project in the Design Server UI.  
Note:  
For version 10.0.0.0, do the following steps.
  - Update the Input Card #2 Config file path.
    - For all run map using the input card #2 Config, update the path references on the input card #2 from data/..config to data/config.
4. Run any of the two router maps in the uc\_translation.mms:
  - ucrtxapi: Router map for MT103 to api translation.
  - ucrtx199: Router map for api to MT199 translation.
5. View the output. Go to the map Diagram-Result, and hover the cursor over Target, select View Data.
6. To download the output file, see [How to download output files of Universal Confirmation API example](#).
  - [How to download output files of Universal Confirmation API example](#)

## How to download output files of Universal Confirmation API example

### Before you begin

Maps are built/run.

### About this task

To download the output file, go to Files in the Design Server UI and download the desired file as follows.

## Procedure

---

1. Click the output Card Settings that you want to download.
2. From the Command text box, copy the name of the file that you want to download. Make sure that you copy only the file name.  
Example 1:

For MT103 to api translation, the Command text box displays: data/api.json.

copy only: api.json.

Example 2:

For api to MT199 translation, the Command text box displays: data/MT199.out.

copy only: MT199.out.

3. From Files, click the New icon to open the New File dialog box.
4. Paste the file name in the Name field.
5. Drop down the Folder field and select data.
6. Click Ok.
7. Build and re-run the map.
8. Go to Files and select Download to view the data.

---

## How to run the Universal Confirmation CSV example in Design Server

### About this task

---

This topic describes how to run the Universal Confirmation CSV in Design Server.

## Procedure

---

1. Import the swiftUCcsv.zip file into the Design Server UI. See the release notes for import instructions.
2. Create a package for the project and build all the maps.
3. Open swiftUCcsv project in the Design Server UI.  
Note:  
For version 10.0.0.0, do the following steps.
  - Update the Input Card #2 Config file path.
    - For all run map using the input card #2 Config, update the path references on the input card #2 from data/..config to data/config.
4. Run any of the two router maps in the uc\_translation.mms:
  - uc csvfile: Router map for MT103 to csv translation.
  - uc trx199: Router map for csv to MT199 translation.
5. View the output. Go to the map Diagram-Result, and hover the cursor over Target, select View Data.
6. To download the output file, see [How to download output files of Universal Confirmation CSV example](#).
  - [How to download output files of Universal Confirmation CSV example](#)

---

## How to download output files of Universal Confirmation CSV example

### Before you begin

---

Maps are built/run.

### About this task

---

To download the output file, go to Files in the Design Server UI and download the desired file as follows.

## Procedure

---

1. Click the output Card Settings that you want to download.
2. From the Command text box, copy the name of the file that you want to download. Make sure that you copy only the file name.  
Example 1:

For csv to MT199 translation, the Command text box displays: data/MT199.out.

copy only: MT199.out.

Example 2:

For MT103 to csv translation, the Command text box displays: data/CSV.csv.

copy only: CSV.csv.

3. From Files, click the New icon to open the New File dialog box.
4. Paste the file name in the Name field.
5. Drop down the Folder field and select data.
6. Click Ok.
7. Build and re-run the map.
8. Go to Files and select Download to view the data.

---

## How to run the Universal Confirmation FIN example in Design Server

### About this task

This topic describes how to run the Universal Confirmation FIN in Design Server.

### Procedure

1. Import the swiftUCfin.zip file into the Design Server UI. See the release notes for import instructions.
2. Create a package for the project and build all the maps.
3. Open swiftUCfin project in the Design Server UI.  
Note:  
For version 10.0.0.0, do the following steps.
  - Update the Input Card #2 Config file path.
    - For all run map using the input card #2 Config, update the path references on the input card #2 from data/..config to data/config.
4. Run the main router map in the uc\_translation.mms:
  - uctrx199: Router map for MT103 to MT199 translation.
5. View the output. Go to the map Diagram-Result, and hover the cursor over Target, select View Data.
6. To download the output file, see [How to download output files of Universal Confirmation FIN example](#).
  - [How to download output files of Universal Confirmation FIN example](#)

---

## How to download output files of Universal Confirmation FIN example

### Before you begin

Maps are built/run.

### About this task

To download the output file, go to Files in the Design Server UI and download the desired file as follows.

### Procedure

1. Click the outputCard Settings that you want to download.
2. From the Command text box, copy the name of the file that you want to download. Make sure that you copy only the file name.  
Example:  
For fin to MT199 translation, the Command text box displays: data/MT199.out.  
copy only: MT199.out.
3. From Files, click the New icon to open the New File dialog box.
4. Paste the file name in the Name field.
5. Drop down the Folder field and select data.
6. Click Ok.
7. Build and re-run the map.
8. Go to Files and select Download to view the data.

---

## Financial Payments Plus standards

The IBM Transformation Extender Pack for Financial Payments Plus, uses the strength of the data definitions and transformation capabilities from various key financial services standards.

The following industry standards are supported:

- CBPR+
- EBA EURO1/STEP1
- NBOR ReGIS
- TARGET2 RTGS/CLM/CoCo/Securities
- CHAPS L4L/ENH
- SWIFT GPI/UC

- SIC RTGS
- [Overview](#)  
IBM Transformation Extender Pack for Financial Payments Plus provides support for clients that need to create, validate, or transform ISO 20022 messages based on various financial market infrastructure standards.
- [Glossary](#)  
The **Cross-Border Payments and Reporting Plus (CBPR+)** working group formulates and refines the market practice guidelines. They develop ISO20022 usage guidelines that cover migration of MT categories 1, 2 and 9 to lessen the impact to correspondent banks as well as other users of these messages, including security players.

## Overview

IBM Transformation Extender Pack for Financial Payments Plus provides support for clients that need to create, validate, or transform ISO 20022 messages based on various financial market infrastructure standards.

The pack provides different levels of validation:

- XML schema only
- Enhanced extended validation
  - Extended validation
    - BIC lookup
    - Ccy lookup
    - Ctry lookup
    - IBAN format check
    - Maximum allowed decimal places
  - Market Infrastructure usage guidelines rules
    - CBPR+
    - CHAPS L4L/ENH
    - EBA Clearing EURO1/STEP1
    - NBOR ReGIS
    - TARGET2 CLM/RTGS/CoCo/Securities
    - SWIFT GPI/UC
    - SIC RTGS

It also supports MX translation based on CBPR+ translation guides.

This Pack can be configured to leverage the MT validation component of the Pack for Financial Payments.

## Glossary

The **Cross-Border Payments and Reporting Plus (CBPR+)** working group formulates and refines the market practice guidelines. They develop ISO20022 usage guidelines that cover migration of MT categories 1, 2 and 9 to lessen the impact to correspondent banks as well as other users of these messages, including security players.

The **European Banking Association Clearing (EBA) EURO1/STEP1 (E1S1)** system processes transactions of high priority and urgency, and primarily of large amount, both at a domestic and at a cross-border level. EURO1 is the only private sector large-value payment system for single same-day euro transactions at a pan-European level. STEP1 is a same-day single payment service and processes retail payment orders and bank-to-bank transfers related to such commercial transactions.

The **National Bank of Romania (NBOR) ReGIS** is the national funds transfer system with real-time gross settlement (RTGS) for payments in LEI currency owned and operated by the National Bank of Romania, used for the settlement of central bank's monetary policy operations, interbank transfers, operations of other financial market infrastructures that operate at the national level, as well as payments initiated by participants on behalf of their customers, regardless of value.

The **TARGET2 (T2)** is the **Real-Time Gross Settlement (RTGS)** system owned and operated by the Eurosystem. Central banks and commercial banks can submit payment orders in euro to TARGET2, where they are processed and settled in central bank money, i.e., money held in an account with a central bank.

The **TARGET2 (T2) Central Liquidity Management (CLM)** component enables its participants to optimize their liquidity management, to group their accounts and to pool the available intraday liquidity for the benefit of all members of the group. It allows participants to control their liquidity outflow by setting limits, actively managing payment queues or prioritizing certain payments.

The **TARGET2 (T2) Common Components (CoCo)** are a set of extracted Common Reference Data Management (CRDM) messages related to common components. CRDM provides common reference data management features that allows participants to create and maintain common reference data for configuration of data related parties, cash accounts, rules and parameters.

The **TARGET2 Securities (T2S)** is a European platform designed to centralize and harmonize the settlement of securities transactions. Managed by the European Central Bank (ECB), T2S enables the seamless settlement of securities transactions in central bank money across various European markets.

The **Bank of England Clearing House Automated Payment System (CHAPS)** is a sterling same-day system that is used to settle the high-value wholesale payments as well as time-critical lower-value payments, like buying or paying a deposit on a property. **CHAPS** migration to ISO20022 will be done in:

- Phase 1 all DP or direct participants will use like-for-like (L4L) ISO20022 messaging, easy to translate back to existing MT messaging, since there will be no additional data requirements.
- Phase 2 all DP or direct participants will use enhanced (ENH) ISO20022 messaging, with data that is above that is required or supported in existing MT message formats where possible. Includes fields which will support transmission of more information about the identities of those involved in the payments and the purpose of the payment.

The **SWIFT Universal Confirmation (UC)** provides an ISO 20022-compliant XML message that can be used to confirm both MT103 or pacs.008 by providing confirmations to the SWIFT GPI Tracker. The Tracker enables participants to track payments status in real time via end-to-end tracking database that is hosted by SWIFT.

The **SWIFT Global Payment Innovation (GPI)** tracker messages enables member banks to track payment statuses in real time through an end-to-end tracking database, incorporated by SWIFT. Includes ISO 20022 XML compliant messages used for payment status updates, reporting, trace and alert notifications.

- SWIFT gpi Customer Credit Transfer (**gCCT**), first service implemented as part of the gpi initiative. Offers enhanced payment services thru faster, same day use of funds, (within the time zone of the receiving gpi customer), transparency of deducts, end-to-end payments tracking, remittance information transferred unaltered.
- SWIFT gpi Instant Payments (**gCCT Inst**) is a gCCT transaction used for cross-border real-time payments.
- SWIFT gpi for Corporates (**g4C**) service allow g4C corporates receive standardized and consistent payment statuses from their g4C banks that can easily be centralized and integrated into their Treasury Management Systems (TMS) or Enterprise Resource Planning (ERP) applications.
- SWIFT gpi Cover Payments (**gCOV**) which is part of the gCCT service. The gCOV service is designed to further increase the timely processing of gCCT transactions by expediting credit to the beneficiary of the cover and facilitating its reconciliation with the underlying credit transfer.
- SWIFT gpi Financial Institution Transfer (**gFIT**) service focuses on financial institution transfers and was designed to support financial institutions with monitoring of their settlement risk and counterparty exposure, intraday liquidity management and forecasting an improved operational efficiency.

The **SWIFT Go** is a service dedicated to low-value, cross-border transactions.

The **SIC (SIX Interbank Clearing) RTGS** is the **Real-Time Gross Settlement (RTGS)** system for clearing and settlement of interbank payments in Swiss francs (CHF). It is operated by SIX Interbank Clearing under the oversight of Swiss National Bank (SNB). SIC processes both large-value and retail payments, ensuring safe and efficient transactions in central bank money. There are two main RTGS services within the SIC framework: SIC for Swiss franc payments and euroSIC for euro payments as well as SIC IP for instant payments.

## Components

The following lists the components of the Pack for Financial Payment Plus:

- Below is a list of translation scenarios supported:

MT - MX translation

Based on CBPR+ Translation guidelines

MT -> MX

- MT103 COR/STP/SWIFTGo/RETN -> pacs.008.001.08 or pacs.004.001.09
- MT196/MT296 -> camt.029.001.09
- MT192/MT292 -> camt.056.001.08
- MT202 COR/COV/RETN -> pacs.009.001.08 ADV/COR/COV or pacs.004.001.09
- MT205 COR/COV/RETN -> pacs.009.001.08 COR/COV or pacs.004.001.09
- MT900/MT910 -> camt.054.001.08

MX -> MT

- pacs.002.001.10 -> MT199/MT299
- pacs.004.001.09 -> MT103 RETN/MT202 RETN/MT205 RETN
- pacs.008.001.08 COR/STP/SWIFTGo -> MT103 COR/SWIFTGo
- pacs.009.001.08 ADV/COR/COV -> MT202 COR/COV
- pacs.009.001.08 COR/COV -> MT205 COR/COV
- camt.029.001.09 -> MT196/MT296
- camt.052.001.08 -> MT942
- camt.053.001.08 -> MT940
- camt.054.001.08 -> MT900/MT910
- camt.056.001.08 -> MT192/MT292
- camt.057.001.06 -> MT210
- camt.058.001.08 -> MT292
- camt.107.001.01 -> MT110
- camt.108.001.01 -> MT111
- camt.109.001.01 -> MT112

- MX validation

Below is a table of supported messages:

ISO 20022 message	CBPR +	CHAPS L4L	CHAPS ENH	EBA E1S1	NBOR ReGIS	TARGET2 CLM	TARGET2 RTGS	TARGET2 CoCo	SWIFT UC	SWIFT GPI	SIC RTGS	TARGET2 Securities
acmt.007.001.02								X				
acmt.010.001.02								X				
acmt.010.001.03.ch.01											X	
acmt.011.001.02								X				
acmt.011.001.03.ch.01											X	
acmt.015.001.02								X				
acmt.015.001.03.ch.01											X	
acmt.019.001.02								X				
acmt.025.001.02								X				
acmt.026.001.02								X				
admi.004.001.02				X	X	X						
admi.004.001.02_fqsm	X	X										
admi.004.001.02_rsr	X	X										
admi.005.001.01						X	X					X
admi.006.001.01												X
admi.007.001.01			X		X	X	X	X				X
camt.003.001.07			X		X	X	X					X

ISO 20022 message	CBPR +	CHAPS L4L	CHAPS ENH	EBA E1S1	NBOR ReGIS	TARGET2 CLM	TARGET2 RTGS	TARGET2 CoCo	SWIFT UC	SWIFT GPI	SIC RTGS	TARGET2 Securities
camt.003.001.07.ch.02											X	
camt.004.001.08				X		X	X					X
camt.004.001.08.ch.02											X	
camt.005.001.08				X	X	X	X					X
camt.005.001.08.ch.01											X	
camt.006.001.08				X	X	X	X					X
camt.006.001.08.ch.02											X	
camt.007.001.08					X		X					
camt.007.001.08.ch.01											X	
camt.008.001.08					X							
camt.008.001.08.ch.01											X	
camt.009.001.07				X			X	X				X
camt.010.001.08				X			X	X				X
camt.011.001.07				X			X	X				
camt.011.001.07.ch.01											X	
camt.012.001.07							X	X				
camt.018.001.05						X	X	X				
camt.019.001.07						X	X	X				X
camt.019.001.07.ch.02											X	
camt.021.001.06								X				
camt.024.001.06									X			
camt.025.001.05					X	X	X	X				X
camt.025.001.05.ch.02											X	
camt.027.001.07.ch.01											X	
camt.029.001.09	X			X	X	X	X					
camt.029.001.09.ch.03											X	
camt.046.001.05					X	X	X					
camt.047.001.06					X	X	X					
camt.048.001.05						X	X	X				
camt.048.001.05.ch.01											X	
camt.049.001.05						X	X					
camt.050.001.05						X	X					X
camt.050.001.05.ch.01											X	
camt.051.001.05												X
camt.052.001.08	X	X	X	X	X							X
camt.052.001.08.ch.02											X	
camt.053.001.08	X	X	X	X	X		X					X
camt.053.001.08_btcs							X					
camt.053.001.08_tsgl							X					
camt.054.001.08	X			X	X	X	X					X
camt.054.001.08.ch.02											X	
camt.054.001.08_ca		X	X									
camt.054.001.08_cld	X	X										
camt.054.001.08_cli		X	X									
camt.054.001.08_lpa		X	X									
camt.055.001.08	X											
camt.056.001.08	X			X	X	X	X					
camt.056.001.08.ch.04											X	
camt.057.001.06	X											
camt.058.001.08	X											
camt.060.001.05	X	X	X		X							
camt.064.001.01											X	
camt.065.001.01											X	
camt.066.001.01											X	
camt.067.001.01											X	
camt.068.001.01											X	
camt.069.001.03								X			X	
camt.070.001.04								X			X	
camt.071.001.03								X				
camt.072.001.01											X	
camt.073.001.01											X	
camt.074.001.01											X	
camt.075.001.01											X	
camt.076.001.01								X				
camt.077.001.01								X				
camt.078.001.01											X	

ISO 20022 message	CBPR +	CHAPS L4L	CHAPS ENH	EBA E1S1	NBOR ReGIS	TARGET2 CLM	TARGET2 RTGS	TARGET2 CoCo	SWIFT UC	SWIFT GPI	SIC RTGS	TARGET2 Securities
camt.079.001.01												X
camt.080.001.01												X
camt.081.001.01												X
camt.082.001.01												X
camt.083.001.01												X
camt.084.001.01												X
camt.085.001.01												X
camt.087.001.06.ch.01												X
camt.099.001.01								X				
camt.100.001.01								X				
camt.105.001.02	X											
camt.105.001.02_mlp	X											
camt.106.001.02	X											
camt.106.001.02_mlp	X											
camt.107.001.01	X											
camt.108.001.01	X											
camt.109.001.01	X											
camt.998.001.03_apmr						X						
camt.998.001.03_gar				X								
camt.998.001.03_gpmr						X						
camt.998.001.03_ibmr							X					
camt.998.001.03_ivmr							X					
camt.998.001.03_mcl						X						
camt.998.001.03_rar				X								
camt.998.001.03_rpimr						X						
camt.998.001.03_rpmr						X						
colr.001.001.01												X
colr.002.001.01												X
head.001.001.01				X		X	X	X				
head.001.001.02	X	X							X	X		
head.002.001.01						X						
mirs.095.001.01	X	X										
pacs.002.001.10	X	X	X	X	X	X	X					
pacs.002.001.10.ch.02												X
pacs.003.001.08	X											
pacs.004.001.09	X	X	X	X	X		X					
pacs.004.001.09.ch.02												X
pacs.008.001.08	X	X	X	X	X		X					
pacs.008.001.08.ch.02												X
pacs.008.001.08_stp	X											
pacs.008.001.08_TREZ					X							
ROBU												
pacs.009.001.08	X	X	X	X	X	X	X					
pacs.009.001.08.ch.03												X
pacs.009.001.08_adv	X											
pacs.009.001.08_cov		X	X		X							
pacs.010.001.03	X			X		X	X					
pacs.010.001.03_mgn	X											
pacs.028.001.03.ch.01												X
pacs.029.001.01					X							
pain.001.001.09	X											
pain.002.001.10	X											
pain.008.001.08	X											
pain.998.001.01_ais							X					
pain.998.001.01_ati								X				
pain.998.001.01_atn								X				
reda.014.001.01									X			
reda.015.001.01									X			
reda.015.001.01.ch.01												X
reda.016.001.01									X			
reda.017.001.01									X			
reda.017.001.01.ch.02												X
reda.022.001.01								X				
reda.031.001.01									X			
reda.039.001.01								X				

ISO 20022 message	CBPR +	CHAPS L4L	CHAPS ENH	EBA E1S1	NBOR ReGIS	TARGET2 CLM	TARGET2 RTGS	TARGET2 CoCo	SWIFT UC	SWIFT GPI	SIC RTGS	TARGET2 Securities
reda.040.001.01								X				
reda.042.001.01								X				
reda.043.001.01								X				
reda.064.001.01								X				
reda.065.001.01								X				
semt.002.001.10												X
semt.013.001.04												X
semt.014.001.06												X
semt.015.001.07												X
semt.016.001.07												X
semt.017.001.09												X
semt.018.001.10												X
semt.019.001.08												X
semt.020.001.05												X
semt.022.001.04												X
semt.025.001.01												X
semt.026.001.01												X
semt.027.001.01												X
semt.028.001.01												X
semt.029.001.01												X
semt.030.001.01												X
semt.031.001.01												X
semt.032.001.01												X
semt.033.001.01												X
semt.034.001.01												X
semt.040.001.01												X
semt.044.001.01												X
sese.020.001.06												X
sese.021.001.05												X
sese.022.001.05												X
sese.023.001.09												X
sese.024.001.10												X
sese.025.001.09												X
sese.027.001.05												X
sese.028.001.08												X
sese.029.001.04												X
sese.030.001.08												X
sese.031.001.08												X
sese.032.001.09												X
sup_camt.998.001.03_gar				X								
sup_camt.998.001.03_rar				X								
supl.021.001.01												X
trck.001.001.02_gCCTI_nst										X		
trck.001.001.02_uc									X			
trck.001.001.03_gCCT										X		
trck.001.001.03_gCOV										X		
trck.001.001.03_gFIT										X		
trck.001.001.03_swiftG_o										X		
trck.002.001.02_gCCT										X		
trck.002.001.02_gCCTI_nst										X		
trck.002.001.02_gCOV										X		
trck.002.001.02_gFIT										X		
trck.002.001.02_swiftG_o										X		
trck.003.001.02										X		
trck.004.001.01_g4C										X		
trck.005.001.02										X		
xmldsig-core-schema					X	X	X					X

## Examples

This documentation lists and describes the examples of Financial Payments Plus.

- [CBPR+ Examples:](#)
  - [CHAPS L4L/ENH Examples:](#)
  - [EBA EURO1/STEP1 Examples:](#)
  - [NBOR ReGIS Examples:](#)
  - [TARGET2 CLM/RTGS/CoCo/Securities Examples:](#)
  - [SWIFT GPI/UC Examples:](#)
  - [SIC \(SIX Interbank Clearing\) Validation Examples:](#)
- 

## CBPR+ Examples:

- [CBPR+ Validation Examples:](#)
  - [CBPR+ Translation Examples:](#)
- 

## CBPR+ Validation Examples:

- [CBPR+ enhanced MX validation \(using flows\) Example](#)  
This example demonstrates the CBPR+ enhanced MX validation for CBPR+ messages using flow server.
  - [CBPR+ enhanced MX validation \(using maps\) Example](#)  
This example demonstrates the CBPR+ enhanced MX validation for CBPR+ messages using maps only.
  - [CBPR+ schema validation \(using maps\) Example](#)  
This example demonstrates the CBPR+ schema validation for CBPR+ messages using maps only.
- 

## CBPR+ enhanced MX validation (using flows) Example

This example demonstrates the CBPR+ enhanced MX validation for CBPR+ messages using flow server.

The flow can perform following level of validation:

- MX extended validation includes following checks:
    - BIC lookup
    - Country code lookup
    - Currency code lookup
    - IBAN format
    - Allowed maximum fractional digits per currency
    - Usage guideline rules, see mxconfig.xml for list of rules
  - Schema validation
  - [What the example contains](#)  
Files included in this example are as follows:
  - [How to run the example](#)  
This CBPR+ MX Validation will use the sample files to demonstrate the generation of validation reports as output from a CBPR+ XML message.
  - [How to customize the mxconfig.xml file](#)  
Rules enforced by the MX validation framework can be customized by editing the mxconfig.xml located under the validation/mx\_extended/data folder.
- 

## What the example contains

Files included in this example are as follows:

- cbpr\_validation.zip
  - Test Data Files:
    - cbpr\_pacs\_009\_adv.xml
    - cbpr\_pacs\_009\_adv\_bad\_rule.xml
    - cbpr\_pacs\_009\_adv\_bad\_schema.xml
  - Validation Files:
    - mxconfig.xml
    - bic.xml
    - currencycodedecimals.xml
  - Schemas:
    - bic.xsd  
Metadata that represents the bic.xml repository file structure.
    - ccy.xsd  
Metadata that represents the currencycodedecimals.xml repository file structure.
    - mxconfig.xsd  
Metadata that represents the mxconfig.xml configuration file structure.
    - cbprErrorReport.mtt  
Metadata that represent the xml based structure of the validation report.

- swiftroute\_funds.mtt  
Metadata that is used as an internal element placeholder.

(Based from SWIFT MyStandards Readiness CBPR+ portal)

- camt.029.001.09.xsd
- camt.052.001.08.xsd
- camt.053.001.08.xsd
- camt.054.001.08.xsd
- camt.055.001.08.xsd
- camt.056.001.08.xsd
- camt.057.001.06.xsd
- camt.058.001.08.xsd
- camt.060.001.05.xsd
- camt.105.001.02.xsd
- camt.105.001.02\_mlp.xsd
- camt.106.001.02.xsd
- camt.106.001.02\_mlp.xsd
- camt.107.001.01.xsd
- camt.108.001.01.xsd
- camt.109.001.01.xsd
- head.001.001.02\_camt\_029.xsd
- head.001.001.02\_camt\_052.xsd
- head.001.001.02\_camt\_053.xsd
- head.001.001.02\_camt\_054.xsd
- head.001.001.02\_camt\_055.xsd
- head.001.001.02\_camt\_056.xsd
- head.001.001.02\_camt\_057.xsd
- head.001.001.02\_camt\_058.xsd
- head.001.001.02\_camt\_060.xsd
- head.001.001.02\_camt\_105.xsd
- head.001.001.02\_camt\_105\_mlp.xsd
- head.001.001.02\_camt\_106.xsd
- head.001.001.02\_camt\_106\_mlp.xsd
- head.001.001.02\_camt\_107.xsd
- head.001.001.02\_camt\_108.xsd
- head.001.001.02\_camt\_109.xsd
- head.001.001.02\_pacs\_002.xsd
- head.001.001.02\_pacs\_003.xsd
- head.001.001.02\_pacs\_004.xsd
- head.001.001.02\_pacs\_008.xsd
- head.001.001.02\_pacs\_008\_stp.xsd
- head.001.001.02\_pacs\_009.xsd
- head.001.001.02\_pacs\_009\_cov.xsd
- head.001.001.02\_pacs\_009\_adv.xsd
- head.001.001.02\_pacs\_010.xsd
- head.001.001.02\_pacs\_010\_mgn.xsd
- head.001.001.02\_pain\_001.xsd
- head.001.001.02\_pain\_002.xsd
- head.001.001.02\_pain\_008.xsd
- pacs.002.001.10.xsd
- pacs.003.001.08.xsd
- pacs.004.001.09.xsd
- pacs.008.001.08.xsd
- pacs.008.001.08\_stp.xsd
- pacs.009.001.08.xsd
- pacs.009.001.08\_adv.xsd
- pacs.009.001.08\_cov.xsd
- pacs.010.001.03.xsd
- pacs.010.001.03\_mgn.xsd
- pain.001.001.09.xsd
- pain.002.001.10.xsd
- pain.008.001.08.xsd
- Maps:
  - For Extended Validation:
    - cbpr6200\_val
    - cbpr6005\_camt\_057\_001\_06
    - cbpr6007\_camt\_060\_001\_05
    - cbpr6001\_camt\_029\_001\_09
    - cbpr6002\_camt\_052\_001\_08
    - cbpr6003\_camt\_053\_001\_08
    - cbpr6006\_camt\_056\_001\_08
    - cbpr6004\_camt\_054\_001\_08
    - cbpr6008\_camt\_055\_001\_08
    - cbpr6009\_camt\_058\_001\_08
    - cbpr6010\_camt\_107\_001\_01
    - cbpr6011\_camt\_108\_001\_01
    - cbpr6012\_camt\_109\_001\_01
    - cbpr6013\_camt\_105\_001\_02

- cbpr6014\_camt\_105\_001\_02\_mlp
- cbpr6015\_camt\_106\_001\_02
- cbpr6016\_camt\_106\_001\_02\_mlp
- cbpr6051\_head112\_camt029
- cbpr6052\_head112\_camt052
- cbpr6053\_head112\_camt053
- cbpr6054\_head112\_camt054
- cbpr6055\_head112\_camt057
- cbpr6056\_head112\_camt056
- cbpr6057\_head112\_camt060
- cbpr6058\_head112\_camt055
- cbpr6059\_head112\_camt058
- cbpr6060\_head112\_camt107
- cbpr6061\_head112\_camt108
- cbpr6062\_head112\_camt109
- cbpr6063\_head112\_camt105
- cbpr6064\_head112\_camt105\_mlp
- cbpr6065\_head112\_camt106
- cbpr6066\_head112\_camt106\_mlp
- cbpr6101\_pacs\_002\_001\_10
- cbpr6102\_pacs\_004\_001\_09
- cbpr6103\_pacs\_008\_001\_08
- cbpr6104\_pacs\_009\_001\_08
- cbpr6105\_pacs\_009\_001\_08\_cov
- cbpr6106\_pacs\_010\_001\_03\_mgn
- cbpr6110\_pacs\_010\_001\_03
- cbpr6107\_pacs\_008\_001\_08\_stp
- cbpr6108\_pacs\_009\_001\_08\_adv
- cbpr6109\_pacs\_003\_001\_08
- cbpr6151\_head112\_pacs002
- cbpr6152\_head112\_pacs004
- cbpr6153\_head112\_pacs008
- cbpr6154\_head112\_pacs009
- cbpr6155\_head112\_pacs009\_cov
- cbpr6156\_head112\_pacs010\_mgn
- cbpr6160\_head112\_pacs010
- cbpr6157\_head112\_pacs008\_stp
- cbpr6158\_head112\_pacs009\_adv
- cbpr6159\_head112\_pacs003
- cbpr6401\_pain\_001\_001\_09
- cbpr6402\_pain\_002\_001\_10
- cbpr6403\_pain\_008\_001\_08
- cbpr6451\_head112\_pain001
- cbpr6452\_head112\_pain002
- cbpr6453\_head112\_pain008

- For Schema Validation:

- cbpr6100\_val
- cbpr6201\_camt\_029\_001\_09
- cbpr6202\_camt\_052\_001\_08
- cbpr6203\_camt\_053\_001\_08
- cbpr6204\_camt\_054\_001\_08
- cbpr6205\_camt\_056\_001\_08
- cbpr6206\_camt\_057\_001\_06
- cbpr6207\_camt\_060\_001\_05
- cbpr6208\_camt\_055\_001\_08
- cbpr6209\_camt\_058\_001\_08
- cbpr6210\_camt\_107\_001\_01
- cbpr6211\_camt\_108\_001\_01
- cbpr6212\_camt\_109\_001\_01
- cbpr6213\_camt\_105\_001\_02
- cbpr6214\_camt\_105\_001\_02\_mlp
- cbpr6215\_camt\_106\_001\_02
- cbpr6216\_camt\_106\_001\_02\_mlp
- cbpr6251\_head112\_camt029
- cbpr6252\_head112\_camt052
- cbpr6253\_head112\_camt053
- cbpr6254\_head112\_camt054
- cbpr6255\_head112\_camt056
- cbpr6256\_head112\_camt057
- cbpr6257\_head112\_camt060
- cbpr6258\_head112\_camt055
- cbpr6259\_head112\_camt058
- cbpr6260\_head112\_camt107
- cbpr6261\_head112\_camt108
- cbpr6262\_head112\_camt109
- cbpr6263\_head112\_camt105
- cbpr6264\_head112\_camt105\_mlp
- cbpr6265\_head112\_camt106
- cbpr6266\_head112\_camt106\_mlp

- cbpr6301\_pacs\_002\_001\_10
  - cbpr6302\_pacs\_004\_001\_09
  - cbpr6303\_pacs\_008\_001\_08
  - cbpr6304\_pacs\_009\_001\_08
  - cbpr6305\_pacs\_009\_001\_08\_cov
  - cbpr6306\_pacs\_010\_001\_03\_mgn
  - cbpr6310\_pacs\_010\_001\_03
  - cbpr6307\_pacs\_008\_001\_08\_stp
  - cbpr6308\_pacs\_009\_001\_08\_adv
  - cbpr6309\_pacs\_003\_001\_08
  - cbpr6351\_head112\_pacs002
  - cbpr6352\_head112\_pacs004
  - cbpr6353\_head112\_pacs008
  - cbpr6354\_head112\_pacs009
  - cbpr6355\_head112\_pacs009\_cov
  - cbpr6356\_head112\_pacs010\_mgn
  - cbpr6360\_head112\_pacs010
  - cbpr6357\_head112\_pacs008\_stp
  - cbpr6358\_head112\_pacs009\_adv
  - cbpr6359\_head112\_pacs003
  - cbpr6501\_pain\_001\_001\_09
  - cbpr6502\_pain\_002\_001\_10
  - cbpr6503\_pain\_008\_001\_08
  - cbpr6551\_head112\_pain001
  - cbpr6552\_head112\_pain002
  - cbpr6553\_head112\_pain008
- Common:
    - mxut1001\_bizsvc\_cbpr
  - Flows:
    - cbpr\_validation

## How to run the example

This CBPR+ MX Validation will use the sample files to demonstrate the generation of validation reports as output from a CBPR+ XML message.

The stopValidation.json file will report the validation failure due to the pre-conversion checks:

**If input file is other than any of the supported CBPR+ messages as per schema list indicated above.**

1. Import the cbpr\_validation.zip project into the Design Server.

2. Open the cbpr\_validation\_flow project in Design Server and view the cbpr\_validation flow.

a. The Flow Description points to all the maps to be executed during the flow process, which will be called from within some of the map nodes in the flow. The following is a list of maps invoked using RUN built-in function during the flow process:

- @packagemap=cbpr\_plus/validation/mx\_extended/maps/cbpr\_mx\_camt\_validation\_enh/cbpr6005\_camt\_057\_001\_06
- @packagemap=cbpr\_plus/validation/mx\_extended/maps/cbpr\_mx\_camt\_validation\_enh/cbpr6007\_camt\_060\_001\_05
- @packagemap=cbpr\_plus/validation/mx\_extended/maps/cbpr\_mx\_camt\_validation\_enh/cbpr6009\_camt\_058\_001\_08
- @packagemap=cbpr\_plus/validation/mx\_extended/maps/cbpr\_mx\_camt\_validation\_enh/cbpr6010\_camt\_107\_001\_01
- @packagemap=cbpr\_plus/validation/mx\_extended/maps/cbpr\_mx\_camt\_validation\_enh/cbpr6011\_camt\_108\_001\_01
- @packagemap=cbpr\_plus/validation/mx\_extended/maps/cbpr\_mx\_camt\_validation\_enh/cbpr6012\_camt\_109\_001\_01
- @packagemap=cbpr\_plus/validation/mx\_extended/maps/cbpr\_mx\_camt\_validation\_enh/cbpr6013\_camt\_105\_001\_02
- @packagemap=cbpr\_plus/validation/mx\_extended/maps/cbpr\_mx\_camt\_validation\_enh/cbpr6014\_camt\_105\_001\_02\_mlp
- @packagemap=cbpr\_plus/validation/mx\_extended/maps/cbpr\_mx\_camt\_validation\_enh/cbpr6015\_camt\_106\_001\_02
- @packagemap=cbpr\_plus/validation/mx\_extended/maps/cbpr\_mx\_camt\_validation\_enh/cbpr6016\_camt\_106\_001\_02\_mlp
- @packagemap=cbpr\_plus/validation/mx\_extended/maps/cbpr\_mx\_camt\_validation\_enh/cbpr6051\_head112\_camt029
- @packagemap=cbpr\_plus/validation/mx\_extended/maps/cbpr\_mx\_camt\_validation\_enh/cbpr6052\_head112\_camt052
- @packagemap=cbpr\_plus/validation/mx\_extended/maps/cbpr\_mx\_camt\_validation\_enh/cbpr6053\_head112\_camt053
- @packagemap=cbpr\_plus/validation/mx\_extended/maps/cbpr\_mx\_camt\_validation\_enh/cbpr6054\_head112\_camt054
- @packagemap=cbpr\_plus/validation/mx\_extended/maps/cbpr\_mx\_camt\_validation\_enh/cbpr6055\_head112\_camt057
- @packagemap=cbpr\_plus/validation/mx\_extended/maps/cbpr\_mx\_camt\_validation\_enh/cbpr6056\_head112\_camt056
- @packagemap=cbpr\_plus/validation/mx\_extended/maps/cbpr\_mx\_camt\_validation\_enh/cbpr6057\_head112\_camt060
- @packagemap=cbpr\_plus/validation/mx\_extended/maps/cbpr\_mx\_camt\_validation\_enh/cbpr6058\_head112\_camt055
- @packagemap=cbpr\_plus/validation/mx\_extended/maps/cbpr\_mx\_camt\_validation\_enh/cbpr6059\_head112\_camt058
- @packagemap=cbpr\_plus/validation/mx\_extended/maps/cbpr\_mx\_camt\_validation\_enh/cbpr6060\_head112\_camt107
- @packagemap=cbpr\_plus/validation/mx\_extended/maps/cbpr\_mx\_camt\_validation\_enh/cbpr6061\_head112\_camt108
- @packagemap=cbpr\_plus/validation/mx\_extended/maps/cbpr\_mx\_camt\_validation\_enh/cbpr6062\_head112\_camt109
- @packagemap=cbpr\_plus/validation/mx\_extended/maps/cbpr\_mx\_camt\_validation\_enh/cbpr6063\_head112\_camt105
- @packagemap=cbpr\_plus/validation/mx\_extended/maps/cbpr\_mx\_camt\_validation\_enh/cbpr6064\_head112\_camt105\_mlp
- @packagemap=cbpr\_plus/validation/mx\_extended/maps/cbpr\_mx\_camt\_validation\_enh/cbpr6065\_head112\_camt106
- @packagemap=cbpr\_plus/validation/mx\_extended/maps/cbpr\_mx\_camt\_validation\_enh/cbpr6066\_head112\_camt106\_mlp
- @packagemap=cbpr\_plus/validation/mx\_extended/maps/cbpr\_mx\_pacs\_validation\_enh/cbpr6151\_head112\_pacs002
- @packagemap=cbpr\_plus/validation/mx\_extended/maps/cbpr\_mx\_pacs\_validation\_enh/cbpr6152\_head112\_pacs004
- @packagemap=cbpr\_plus/validation/mx\_extended/maps/cbpr\_mx\_pacs\_validation\_enh/cbpr6153\_head112\_pacs008
- @packagemap=cbpr\_plus/validation/mx\_extended/maps/cbpr\_mx\_pacs\_validation\_enh/cbpr6154\_head112\_pacs009
- @packagemap=cbpr\_plus/validation/mx\_extended/maps/cbpr\_mx\_pacs\_validation\_enh/cbpr6155\_head112\_pacs009\_cov
- @packagemap=cbpr\_plus/validation/mx\_extended/maps/cbpr\_mx\_pacs\_validation\_enh/cbpr6156\_head112\_pacs010\_mgn
- @packagemap=cbpr\_plus/validation/mx\_extended/maps/cbpr\_mx\_pacs\_validation\_enh/cbpr6160\_head112\_pacs010
- @packagemap=cbpr\_plus/validation/mx\_extended/maps/cbpr\_mx\_pacs\_validation\_enh/cbpr6157\_head112\_pacs008\_stp
- @packagemap=cbpr\_plus/validation/mx\_extended/maps/cbpr\_mx\_pacs\_validation\_enh/cbpr6158\_head112\_pacs009\_adv

- @packagemap=cbpr\_plus/validation/mx\_extended/maps/cbpr\_mx\_pacs\_validation\_enh/cbpr6159\_head112\_pacs003
- @packagemap=cbpr\_plus/validation/mx\_extended/maps/cbpr\_mx\_pain\_validation\_enh/cbpr6401\_pain\_001\_001\_09
- @packagemap=cbpr\_plus/validation/mx\_extended/maps/cbpr\_mx\_pain\_validation\_enh/cbpr6402\_pain\_002\_001\_10
- @packagemap=cbpr\_plus/validation/mx\_extended/maps/cbpr\_mx\_pain\_validation\_enh/cbpr6403\_pain\_008\_001\_08
- @packagemap=cbpr\_plus/validation/mx\_extended/maps/cbpr\_mx\_pain\_validation\_enh/cbpr6451\_head112\_pain001
- @packagemap=cbpr\_plus/validation/mx\_extended/maps/cbpr\_mx\_pain\_validation\_enh/cbpr6452\_head112\_pain002
- @packagemap=cbpr\_plus/validation/mx\_extended/maps/cbpr\_mx\_pain\_validation\_enh/cbpr6453\_head112\_pain008
- @packagemap=cbpr\_plus/validation/schema\_only/maps/cbpr\_mx\_schema\_validation\_xsd/cbpr6201\_camt\_029\_001\_09
- @packagemap=cbpr\_plus/validation/schema\_only/maps/cbpr\_mx\_schema\_validation\_xsd/cbpr6202\_camt\_052\_001\_08
- @packagemap=cbpr\_plus/validation/schema\_only/maps/cbpr\_mx\_schema\_validation\_xsd/cbpr6203\_camt\_053\_001\_08
- @packagemap=cbpr\_plus/validation/schema\_only/maps/cbpr\_mx\_schema\_validation\_xsd/cbpr6204\_camt\_054\_001\_08
- @packagemap=cbpr\_plus/validation/schema\_only/maps/cbpr\_mx\_schema\_validation\_xsd/cbpr6205\_camt\_056\_001\_08
- @packagemap=cbpr\_plus/validation/schema\_only/maps/cbpr\_mx\_schema\_validation\_xsd/cbpr6206\_camt\_057\_001\_06
- @packagemap=cbpr\_plus/validation/schema\_only/maps/cbpr\_mx\_schema\_validation\_xsd/cbpr6207\_camt\_060\_001\_05
- @packagemap=cbpr\_plus/validation/schema\_only/maps/cbpr\_mx\_schema\_validation\_xsd/cbpr6208\_camt\_055\_001\_08
- @packagemap=cbpr\_plus/validation/schema\_only/maps/cbpr\_mx\_schema\_validation\_xsd/cbpr6209\_camt\_058\_001\_08
- @packagemap=cbpr\_plus/validation/schema\_only/maps/cbpr\_mx\_schema\_validation\_xsd/cbpr6210\_camt\_107\_001\_01
- @packagemap=cbpr\_plus/validation/schema\_only/maps/cbpr\_mx\_schema\_validation\_xsd/cbpr6211\_camt\_108\_001\_01
- @packagemap=cbpr\_plus/validation/schema\_only/maps/cbpr\_mx\_schema\_validation\_xsd/cbpr6212\_camt\_109\_001\_01
- @packagemap=cbpr\_plus/validation/schema\_only/maps/cbpr\_mx\_schema\_validation\_xsd/cbpr6213\_camt\_105\_001\_02
- @packagemap=cbpr\_plus/validation/schema\_only/maps/cbpr\_mx\_schema\_validation\_xsd/cbpr6214\_camt\_105\_001\_02\_mlp
- @packagemap=cbpr\_plus/validation/schema\_only/maps/cbpr\_mx\_schema\_validation\_xsd/cbpr6215\_camt\_106\_001\_02
- @packagemap=cbpr\_plus/validation/schema\_only/maps/cbpr\_mx\_schema\_validation\_xsd/cbpr6216\_camt\_106\_001\_02\_mlp
- @packagemap=cbpr\_plus/validation/schema\_only/maps/cbpr\_mx\_schema\_validation\_xsd/cbpr6251\_head112\_camt029
- @packagemap=cbpr\_plus/validation/schema\_only/maps/cbpr\_mx\_schema\_validation\_xsd/cbpr6252\_head112\_camt052
- @packagemap=cbpr\_plus/validation/schema\_only/maps/cbpr\_mx\_schema\_validation\_xsd/cbpr6253\_head112\_camt053
- @packagemap=cbpr\_plus/validation/schema\_only/maps/cbpr\_mx\_schema\_validation\_xsd/cbpr6254\_head112\_camt054
- @packagemap=cbpr\_plus/validation/schema\_only/maps/cbpr\_mx\_schema\_validation\_xsd/cbpr6255\_head112\_camt056
- @packagemap=cbpr\_plus/validation/schema\_only/maps/cbpr\_mx\_schema\_validation\_xsd/cbpr6256\_head112\_camt057
- @packagemap=cbpr\_plus/validation/schema\_only/maps/cbpr\_mx\_schema\_validation\_xsd/cbpr6257\_head112\_camt060
- @packagemap=cbpr\_plus/validation/schema\_only/maps/cbpr\_mx\_schema\_validation\_xsd/cbpr6258\_head112\_camt055
- @packagemap=cbpr\_plus/validation/schema\_only/maps/cbpr\_mx\_schema\_validation\_xsd/cbpr6259\_head112\_camt058
- @packagemap=cbpr\_plus/validation/schema\_only/maps/cbpr\_mx\_schema\_validation\_xsd/cbpr6260\_head112\_camt107
- @packagemap=cbpr\_plus/validation/schema\_only/maps/cbpr\_mx\_schema\_validation\_xsd/cbpr6261\_head112\_camt108
- @packagemap=cbpr\_plus/validation/schema\_only/maps/cbpr\_mx\_schema\_validation\_xsd/cbpr6262\_head112\_camt109
- @packagemap=cbpr\_plus/validation/schema\_only/maps/cbpr\_mx\_schema\_validation\_xsd/cbpr6263\_head112\_camt105
- @packagemap=cbpr\_plus/validation/schema\_only/maps/cbpr\_mx\_schema\_validation\_xsd/cbpr6264\_head112\_camt105\_mlp
- @packagemap=cbpr\_plus/validation/schema\_only/maps/cbpr\_mx\_schema\_validation\_xsd/cbpr6265\_head112\_camt106
- @packagemap=cbpr\_plus/validation/schema\_only/maps/cbpr\_mx\_schema\_validation\_xsd/cbpr6266\_head112\_camt106\_mlp
- @packagemap=cbpr\_plus/validation/schema\_only/maps/cbpr\_mx\_schema\_validation\_xsd/cbpr6301\_pacs\_002\_001\_10
- @packagemap=cbpr\_plus/validation/schema\_only/maps/cbpr\_mx\_schema\_validation\_xsd/cbpr6302\_pacs\_004\_001\_09
- @packagemap=cbpr\_plus/validation/schema\_only/maps/cbpr\_mx\_schema\_validation\_xsd/cbpr6303\_pacs\_008\_001\_08
- @packagemap=cbpr\_plus/validation/schema\_only/maps/cbpr\_mx\_schema\_validation\_xsd/cbpr6304\_pacs\_009\_001\_08
- @packagemap=cbpr\_plus/validation/schema\_only/maps/cbpr\_mx\_schema\_validation\_xsd/cbpr6305\_pacs\_009\_001\_08\_cov
- @packagemap=cbpr\_plus/validation/schema\_only/maps/cbpr\_mx\_schema\_validation\_xsd/cbpr6306\_pacs\_010\_001\_03\_mgn
- @packagemap=cbpr\_plus/validation/schema\_only/maps/cbpr\_mx\_schema\_validation\_xsd/cbpr6310\_pacs\_010\_001\_03
- @packagemap=cbpr\_plus/validation/schema\_only/maps/cbpr\_mx\_schema\_validation\_xsd/cbpr6307\_pacs\_008\_001\_08\_stp
- @packagemap=cbpr\_plus/validation/schema\_only/maps/cbpr\_mx\_schema\_validation\_xsd/cbpr6308\_pacs\_009\_001\_08\_adv
- @packagemap=cbpr\_plus/validation/schema\_only/maps/cbpr\_mx\_schema\_validation\_xsd/cbpr6309\_pacs\_003\_001\_08
- @packagemap=cbpr\_plus/validation/schema\_only/maps/cbpr\_mx\_schema\_validation\_xsd/cbpr6351\_head112\_pacs002
- @packagemap=cbpr\_plus/validation/schema\_only/maps/cbpr\_mx\_schema\_validation\_xsd/cbpr6352\_head112\_pacs004
- @packagemap=cbpr\_plus/validation/schema\_only/maps/cbpr\_mx\_schema\_validation\_xsd/cbpr6353\_head112\_pacs008
- @packagemap=cbpr\_plus/validation/schema\_only/maps/cbpr\_mx\_schema\_validation\_xsd/cbpr6354\_head112\_pacs009
- @packagemap=cbpr\_plus/validation/schema\_only/maps/cbpr\_mx\_schema\_validation\_xsd/cbpr6355\_head112\_pacs009\_cov
- @packagemap=cbpr\_plus/validation/schema\_only/maps/cbpr\_mx\_schema\_validation\_xsd/cbpr6356\_head112\_pacs010\_mgn
- @packagemap=cbpr\_plus/validation/schema\_only/maps/cbpr\_mx\_schema\_validation\_xsd/cbpr6360\_head112\_pacs010
- @packagemap=cbpr\_plus/validation/schema\_only/maps/cbpr\_mx\_schema\_validation\_xsd/cbpr6357\_head112\_pacs008\_stp
- @packagemap=cbpr\_plus/validation/schema\_only/maps/cbpr\_mx\_schema\_validation\_xsd/cbpr6358\_head112\_pacs009\_adv
- @packagemap=cbpr\_plus/validation/schema\_only/maps/cbpr\_mx\_schema\_validation\_xsd/cbpr6359\_head112\_pacs003
- @packagemap=cbpr\_plus/validation/schema\_only/maps/cbpr\_mx\_schema\_validation\_xsd/cbpr6501\_pain\_001\_001\_09
- @packagemap=cbpr\_plus/validation/schema\_only/maps/cbpr\_mx\_schema\_validation\_xsd/cbpr6502\_pain\_002\_001\_10
- @packagemap=cbpr\_plus/validation/schema\_only/maps/cbpr\_mx\_schema\_validation\_xsd/cbpr6503\_pain\_008\_001\_08
- @packagemap=cbpr\_plus/validation/schema\_only/maps/cbpr\_mx\_schema\_validation\_xsd/cbpr6551\_head112\_pain001
- @packagemap=cbpr\_plus/validation/schema\_only/maps/cbpr\_mx\_schema\_validation\_xsd/cbpr6552\_head112\_pain002
- @packagemap=cbpr\_plus/validation/schema\_only/maps/cbpr\_mx\_schema\_validation\_xsd/cbpr6553\_head112\_pain008

b. It utilizes the following nodes:

i. Source Nodes:

- mx\_input

This node identifies the input data to be validated in the flow. It uses the INPUT\_FILE variable to set the location of the data.

ii. Map Nodes:

- MX pre-check

Runs map mxut1001\_bizsvc\_cbpr which sets flow variables, checks pre-validation conditions and creates infoset.json for validation.

- EXT\_VAL

Runs map cbpr6200\_val which calls the appropriate map to perform extended validation of a CBPR+ message.

- XSD\_VAL

Runs map cbpr6100\_val which calls the appropriate map to perform schema validation of a CBPR+ message.

Note: The maps cbpr6200\_val and cbpr6100\_val internally call all the other maps identified in the above Flow Description step.

- iii. Decision Nodes:
    - pre-valid chk  
This node checks the flow variable stopValidation to decide if further validation/processing is not required.
    - EXT\_RESULT\_FORMAT  
This node checks the value of the flow variable REPORT\_FORMAT to determine if the resulting report for extended validation should be generated in XML (default) or JSON.
    - XSD\_RESULT\_FORMAT  
This node checks the value of the flow variable REPORT\_FORMAT to determine if the resulting report for schema-only validation should be generated in XML (default) or JSON.
  - iv. Route Nodes:
    - VAL\_TYPE  
This node checks the variable VALIDATION\_TYPE to determine if the process will do extended validation or schema-only validation on the data.
  - v. Log Nodes:
    - FAILURE  
In case of no validation due to precondition check failures, this node creates a log file specified by the flow variable FAILURE\_LOG.
  - vi. Fail Node:
    - STOP  
It generates error message setup in the node in case of no validation performed.
  - vii. Passthrough Nodes:
    - EXT\_XML\_CONVERT  
This node receives an extended validation report in XML format and does not modify the content, thus passing it as is to the appropriate target node.
    - XSD\_XML\_CONVERT  
This node receives a schema-only validation report in XML format and does not modify the content, thus passing it as is to the appropriate target node.
  - viii. Format Converter Nodes:
    - EXT\_JSON\_CONVERT  
This node receives an extended validation report in XML format and converts it to JSON before passing it to the appropriate target node.
    - XSD\_JSON\_CONVERT  
This node receives a schema-only validation report in XML format and converts it to JSON before passing it to the appropriate target node.
  - ix. Target Nodes:
    - ext\_xml  
This node contains the resulting extended validation report in XML format and creates the output as defined by the variable OUTPUT\_RESULT\_XML.
    - ext\_json  
This node contains the resulting extended validation report in JSON format and creates the output as defined by the variable OUTPUT\_RESULT\_JSON.
    - xsd\_xml  
This node contains the resulting schema-only validation report in XML format and creates the output as defined by the variable OUTPUT\_RESULT\_XML.
    - xsd\_json  
This node contains the resulting schema-only validation report in JSON format and creates the output as defined by the variable OUTPUT\_RESULT\_JSON.
  - c. It utilizes the following variables:
- Flow variables:
- VALIDATION\_TYPE  
The default value is extended.
- For schema-only validation, it can be changed to schema. This is used in Route node VAL\_TYPE.
- REPORT\_FORMAT  
The default value is xml.
- It can be changed to json. This is used in Decision nodes EXT\_RESULT\_FORMAT and XSD\_RESULT\_FORMAT.
- INPUT\_FILE  
The default value is ..../tools/mxservice/data/cbpr\_pacs\_009\_adv.xml.
- This is the data file to be used for validation and can be customized. This is used in Source node mx\_input.
- BIC\_FILE  
The default value is ..../data/bic.xml.
- This is the location of the bic cross-reference file that is used in all the maps executed by the map in node EXT\_VAL. It can be customized.
- CCY\_FILE  
The default value is ..../data/currencycodedecimals.xml.
- This is the location of the country code cross-reference file that is used in all the maps executed by the map in node EXT\_VAL. It can be customized.
- MXCONFIG\_FILE  
The default value is ..../data/mxconfig.xml.

This is the location of the file containing the rule validation settings that is used in all the maps executed by the map in node EXT\_VAL. It can be customized.

- OUTPUT\_RESULT\_XML

The default value is validation\_result.xml.

This is the location of the validation report file in xml format. It is used in Target nodes ext\_xml and xsd\_xml. It can be customized.

- OUTPUT\_RESULT\_JSON

The default value is validation\_result.json.

This is the location of the validation report file in json format. It is used in Target nodes ext\_json and xsd\_json. It can be customized.

- FAILURE\_LOG

The default value is stopMXValidation.json.

This is the location of the failure log. It is used in Log node FAILURE. It can be customized.

- bizSvc

For internal use in the Map node MX pre-check to determine the type of data being read in the input file.

- stopValidation

For internal use in the Map node MX pre-check and is checked in Decision node pre-valid chk to cause a Failure log in case a data file was not recognized as a valid CBPR+ message.

3. Open the main flow cbpr\_validation in the Design Server. It utilizes above one source node, three map nodes, one route nodes, three decision nodes, one log node, one fail node, two passthrough nodes, two format converter nodes and four target nodes.

4. In Design Server, create a package cbpr\_mx\_validation that contains the input files and one flow cbpr\_validation. The maps will automatically be included during deployment of the package onto the runtime server.

Note: Not required for running flows on Design Server user interface directly.

- [\*\*Run the example using three different methods\*\*](#)

You can run the example using three different methods:

---

## Run the example using three different methods

You can run the example using three different methods:

1. Running the example from the Design Server user interface.

2. Deploying the example to tx-rest and running it using the Swagger interface.

3. Deploying locally or to ftp server and using the flow command server process (flowcmdserver).

- [\*\*Run the example from the Design Server user interface\*\*](#)

Use the following steps to run the example from the Design Server user interface:

• [\*\*Run the example using the Swagger interface\*\*](#)

To run the example, deploy to tx-rest and run it using the Swagger interface.

- [\*\*Run the example using the flow command server process \(flowcmdserver\)\*\*](#)

To run the example, deploy locally or to ftp server and use the flow command server process (flowcmdserver).

---

## Run the example from the Design Server user interface

Use the following steps to run the example from the Design Server user interface:

1. In the Design Server, open cbpr\_validation flow and click on Run button.

2. Set toggle switch Run On Design Server and select flow input file. Example is cbpr\_pacs\_009\_adv.xml.

Note: If a default path is assigned to variable INPUT\_FILE, not selecting an input file will use the value defined for the variable.

Flow will run and report with the green check box. The report lists the execution of all nodes.

3. To examine the data passed on each link, right click on each link and select View Link Data.

4. To view the target node, right click on the node input icon and select View Data.

---

## Run the example using the Swagger interface

To run the example, deploy to tx-rest and run it using the Swagger interface.

1. Create a Web type server definition where the runtime tx-rest is installed if you do not have a server definition to deploy in Design Server.

2. If not running, start tx-rest on the server: <DTX\_HOME>/restapi/tomcat/dtxtomcat start tx-rest.

3. Deploy the created package to the server definition in Design Server.

4. Bring up the tx-rest Swagger UI: <DTX\_HOME>/restapi/tomcat/dtxtomcat showdocs tx-rest.

5. In the Swagger Explore window, enter /v2/docs and click on the Explore button to view the deployed package.

You should see a Miscellaneous section having two actions: A **PUT** /v2/run/cbpr\_validation and a **POST** /v2/run/cbpr\_validation.

6. In the **PUT** action:

a. Expand the **PUT** action and select the Try it out button.

b. Leave the input and output sections empty.

c. Under flow\_vars section, delete the entire content and replace it with the commands to run the flow. For example:

```

 {
 "INPUT_FILE": "C:/mydir/data/cbpr_pacs_009_adv.xml",
 "OUTPUT_RESULT_JSON": "C:/mydir/data/validation_result.json",
 "REPORT_FORMAT": "json",
 "VALIDATION_TYPE": "extended"
 }

```

or

```

 {
 "INPUT_FILE": "C:/mydir/data/cbpr_pacs_009_adv.xml",
 "OUTPUT_RESULT_XML": "C:/mydir/data/validation_result.xml",
 "REPORT_FORMAT": "xml",
 "VALIDATION_TYPE": "extended"
 }

```

- d. Click Execute blue bar for running the flow. When flow completes execution, 200 response code is shown in the Server Response section. This run should generate the output same as above.
7. Refresh the browser to delete the deployed package and get the Swagger Explore back to /tx-rest/openapi.json. There will be a DELETE /v2/packages/{name} action.
  8. Expand the DELETE /v2/packages/{name} action and select Try it out.
  9. In the Name field, enter the name you gave to the created package.
  10. Select the true option from the stop query drop down, select the blue Execute bar.  
You can see a response of 204 with a timestamp, if it is successful.

## Run the example using the flow command server process (flowcmdserver)

To run the example, deploy locally or to ftp server and use the flow command server process (flowcmdserver).

1. Create a server definition where the runtime is installed, if you do not have a server local definition or ftp execution definition to deploy to, in Design Server.
2. In Design Server, deploy the created package to the server definition. For example, deploy to c:\ftpserver\deployment directory.
3. Execute the flow using the flow command server. The below command is for CBPR+ cbpr\_pacs\_009\_adv.xml message.

```
flowcmdserver.exe --dir c:\ftpserver\deployment\flows --flow cbpr_validation.json
--var "INPUT_FILE=C:\ftpserver\deployment\tools\mx_service\data\cbpr_pacs_009_adv.xml"
--audit "C:\ftpserver\deployment\tools\mx_service\data\cbpr_pacs_009_adv_adt.json" -ad
```

- The results will be in C:\ftpserver\deployment\flows as validation\_result.xml or validation\_result.json depending on the value of the REPORT\_FORMAT variable.
4. Use the following command to execute with the variable VALIDATION\_TYPE value defined in the Flow settings in Design Server (extended or schema).

```
--flow cbpr_validation.json
--var "INPUT_FILE=C:\ftpserver\deployment\tools\mx_service\data\cbpr_pacs_009_adv.xml"
--var "VALIDATION_TYPE=extended"
--audit "C:\ftpserver\deployment\tools\mx_service\data\cbpr_pacs_009_adv_adt.json" -ad
```

5. The flow will report status similar to:

```
***Starting flow command server

Flow UUID: 85eabe2c-2302-4543-b347-9e18d65c6816
The flow audit file is: "C:\\ftpserver\\\\deployment\\\\tools\\\\mx_service\\\\data\\\\cbpr_pacs_009_adv_adt.json"
Flow completed successfully
Elapsed time: 3048ms
```

6. Examine the flow output result file validation\_results.xml and the audit file cbpr\_pacs\_009\_adv\_adt.json.

## How to customize the mxconfig.xml file

Rules enforced by the MX validation framework can be customized by editing the mxconfig.xml located under the validation/mx\_extended/data folder.

All the rules enforced by the MX validation map are under element ExtendedValidation.

Any of the sub-elements under ExtendedValidation can be enabled by setting the value to T or disabled by setting to F.

### Example

To disable the BIC lookup validation feature.

### About this task

Change sub-element ExtendedValidation/BICValidation value from T to F.

### Procedure

1. Download the mxconfig.xml file from the Files tab.
2. Open the mxconfig.xml file and edit the sub-element ExtendedValidation/BICValidation value setting from T to F, then save the file.
3. Upload the edited mxconfig.xml file to the project.
4. Run the flow.  
Run using an input with BIC lookup validation issue.

Right click on the output card on canvas and select View data. No BIC lookup validation reported.

## CBPR+ enhanced MX validation (using maps) Example

This example demonstrates the CBPR+ enhanced MX validation for CBPR+ messages using maps only.

The maps can perform following level of validation:

- MX enhanced validation includes following checks:
  - BIC lookup
  - Country code lookup
  - Currency code lookup
  - IBAN format
  - Allowed maximum fractional digits per currency
  - Usage guideline rules, see mxconfig.xml for list of rules
- Schema validation:
  - Enforces validation based on XSD.

- **What the example contains**

Files and directories included in this example are as follows:

- **Run the example in Design Studio**
- **Run the example in Design Server User Interface**
- **How to customize the mxconfig.xml file**

Rules enforced by the MX validation framework can be customized by editing the mxconfig.xml located under the validation/mx\_extended/data folder.

## What the example contains

Files and directories included in this example are as follows:

- Maps:  
The maps directory contains the following map sources to use when running under Design Studio.

- cbpr\_mx\_camt\_validation\_enh.mms  
Utility map called by main map for all the camt CBPR xml message MX validation.
- cbpr\_mx\_pacs\_validation\_enh.mms  
Utility map called by main map for all the pacs CBPR xml message MX validation.
- cbpr\_mx\_pain\_validation\_enh.mms  
Utility map called by main map for all the pain CBPR xml message MX validation.
- cbpr\_mx\_validation\_frmwrk\_map\_enh.mms  
The Main map used to apply MX validation to CBPR xml messages.

Note: When running under Design Server, the main framework map is cbpr\_mx\_validation\_frmwrk\_map\_flw\_enh.mms.

- Schemas:  
The schemas directory contains the following files:

- bic.xsd  
Metadata that represents the bic.xml repository file structure.
- ccy.xsd  
Metadata that represents the currencycodedecimals.xml repository file structure.
- mxconfig.xsd  
Metadata that represents the mxconfig.xml configuration file structure.

(Based from SWIFT MyStandards Readiness CBPR+ Portal)

- camt.029.001.09.xsd
- camt.052.001.08.xsd
- camt.053.001.08.xsd
- camt.054.001.08.xsd
- camt.055.001.08.xsd
- camt.056.001.08.xsd
- camt.057.001.06.xsd
- camt.058.001.08.xsd
- camt.107.001.01.xsd
- camt.108.001.01.xsd
- camt.060.001.05.xsd
- camt.105.001.02.xsd
- camt.105.001.02\_mlp.xsd
- camt.106.001.02.xsd
- camt.106.001.02\_mlp.xsd
- camt.109.001.01.xsd
- pacs.002.001.10.xsd
- pacs.003.001.08.xsd
- pacs.004.001.09.xsd

- pacs.008.001.08.xsd
- pacs.008.001.08\_stp.xsd
- pacs.009.001.08.xsd
- pacs.009.001.08\_cov.xsd
- pacs.009.001.08\_adv.xsd
- pacs.010.001.03.xsd
- pacs.010.001.03\_mgn.xsd
- pain.001.001.09.xsd
- pain.002.001.10.xsd
- pain.008.001.08.xsd
- head.001.001.02.camt\_029.xsd
- head.001.001.02.camt\_052.xsd
- head.001.001.02.camt\_053.xsd
- head.001.001.02.camt\_054.xsd
- head.001.001.02.camt\_055.xsd
- head.001.001.02.camt\_056.xsd
- head.001.001.02.camt\_057.xsd
- head.001.001.02.camt\_058.xsd
- head.001.001.02.camt\_060.xsd
- head.001.001.02.camt\_105.xsd
- head.001.001.02.camt\_105\_mlp.xsd
- head.001.001.02.camt\_106.xsd
- head.001.001.02.camt\_106\_mlp.xsd
- head.001.001.02.camt\_107.xsd
- head.001.001.02.camt\_108.xsd
- head.001.001.02.camt\_109.xsd
- head.001.001.02.pacs\_002.xsd
- head.001.001.02.pacs\_003.xsd
- head.001.001.02.pacs\_004.xsd
- head.001.001.02.pacs\_008.xsd
- head.001.001.02\_pacs\_008\_stp.xsd
- head.001.001.02.pacs\_009.xsd
- head.001.001.02\_pacs\_009\_cov.xsd
- head.001.001.02\_pacs\_009\_adv.xsd
- head.001.001.02.pacs\_010.xsd
- head.001.001.02.pacs\_010\_mgn.xsd
- head.001.001.02.pain\_001.xsd
- head.001.001.02.pain\_002.xsd
- head.001.001.02.pain\_008.xsd

- Trees:

The trees directory contains the following files:

- swiftroute\_funds.mtt  
Metadata that is used as an internal element placeholder.
- cbprErrorReport.mtt  
Metadata that represents the xml based structure of the validation report.

- Data:

The data directory contains the following files:

- bic.xml  
Repository file listing all BICs which are used during validation.
- currencycodedecimals.xml  
Repository file listing country codes, currency codes and corresponding maximum fractionally digits, used as reference for validation.
- mxconfig.xml  
Contains the MX configuration information on how to process the message.

- Sample CBPR+ valid files for test purpose with header envelope:

- bah\_pacs\_002\_001\_10\_valid.xml
- bah\_pacs\_003\_001\_08\_valid.xml
- bah\_pacs\_004\_001\_09\_valid.xml
- bah\_pacs\_008\_001\_08\_valid.xml
- bah\_pacs\_008\_001\_08\_stp\_valid.xml
- bah\_pacs\_009\_001\_08\_valid.xml
- bah\_pacs\_009\_001\_08\_cov\_valid.xml
- bah\_pacs\_009\_001\_08\_adv\_valid.xml
- bah\_pacs\_010\_001\_03\_valid.xml
- bah\_pacs\_010\_001\_03\_mgn\_valid.xml
- bah\_camt\_029\_001\_09\_valid.xml
- bah\_camt\_052\_001\_08\_valid.xml
- bah\_camt\_053\_001\_08\_valid.xml
- bah\_camt\_054\_001\_08\_valid.xml
- bah\_camt\_055\_001\_08\_valid.xml
- bah\_camt\_056\_001\_08\_valid.xml
- bah\_camt\_057\_001\_06\_valid.xml
- bah\_camt\_058\_001\_08\_valid.xml
- bah\_camt\_060\_001\_05\_valid.xml
- bah\_camt\_105\_001\_02\_valid.xml

- bah\_camt\_105\_001\_02\_mlp\_valid.xml
- bah\_camt\_106\_001\_02\_valid.xml
- bah\_camt\_106\_001\_02\_mlp\_valid.xml
- bah\_camt\_107\_001\_01\_valid.xml
- bah\_camt\_108\_001\_01\_valid.xml
- bah\_camt\_109\_001\_01\_valid.xml
- bah\_pain\_001\_001\_09\_valid.xml
- bah\_pain\_002\_001\_10\_valid.xml
- bah\_pain\_008\_001\_08\_valid.xml

## Run the example in Design Studio

1. Build all the maps in the .mms files listed in What the example contains section.
2. Replace the input card 1 in router map, cbpr6000\_val (under cbpr\_mx\_validation\_frmwrk\_map\_enh.mms file or cbpr\_mx\_validation\_frmwrk\_map\_flw\_enh.mms file if running under Design Server) with desired input file.
3. Run the main router map, cbpr6000\_val.  
You will see the output file in the data folder called cbpr\_error\_report.xml.

## Run the example in Design Server User Interface

1. Open a browser pointing to the installation of the IBM Sterling Transformation Extender.
2. Enter the user and password credentials.
3. If the project does not exist, import cbpr\_validation\_map.zip.
4. If absent, create a package containing all the Maps, Files and Flows.
5. Build the package in the desired server (to build all maps).
6. Open the project.
7. In the Maps tab, open the map cbpr6000\_val.
8. Modify the settings for input card 1 on design canvas to point to the desired test file.
9. Save, build, and run the map cbpr6000\_val.
10. Right click on the output card on canvas and select View data.

## How to customize the mxconfig.xml file

Rules enforced by the MX validation framework can be customized by editing the mxconfig.xml located under the validation/mx\_extended/data folder.

All the rules enforced by the MX validation map are under element ExtendedValidation.

Any of the sub-elements under ExtendedValidation can be enabled by setting the value to T or disabled by setting to F.

### Example

To disable the BIC lookup validation feature.

### About this task

Change sub-element ExtendedValidation/BICValidation value from T to F.

### Procedure

1. Download the mxconfig.xml file from the Files tab.
2. Open the mxconfig.xml file and edit the sub-element ExtendedValidation/BICValidation value setting from T to F, then save the file.
3. Upload the edited mxconfig.xml file to the project.
4. Run the map.  
Run using an input with BIC lookup validation issue.

Right click on the output card on canvas and select View data. No BIC lookup validation reported.

## CBPR+ schema validation (using maps) Example

This example demonstrates the CBPR+ schema validation for CBPR+ messages using maps only.

The maps can perform following level of validation:

- Schema validation
- [What the example contains](#)  
Files and directories included in this example are as follows:
- [Run the example in Design Studio](#)

- [Run the example in Design Server User Interface](#)
- 

## What the example contains

Files and directories included in this example are as follows:

- Maps:  
The maps directory contains the following map sources to use when running under Design Studio.
  - cbpr\_mx\_schema\_validation\_xsd.mms  
Contains maps for Schema validation of any CBPR message.
  - cbpr\_mx\_schema\_validation\_frmwrk\_map\_xsd.mms  
The Main map used to apply schema validation to CBPR xml messages.
- Schemas:  
The schemas directory contains the following files:  
(Based from SWIFT MyStandards Readiness CBPR+ portal)
  - camt.029.001.09.xsd
  - camt.052.001.08.xsd
  - camt.053.001.08.xsd
  - camt.054.001.08.xsd
  - camt.055.001.08.xsd
  - camt.056.001.08.xsd
  - camt.057.001.06.xsd
  - camt.058.001.08.xsd
  - camt.107.001.01.xsd
  - camt.108.001.01.xsd
  - camt.060.001.05.xsd
  - camt.105.001.02.xsd
  - camt.105.001.02\_mlp.xsd
  - camt.106.001.02.xsd
  - camt.106.001.02\_mlp.xsd
  - camt.109.001.01.xsd
  - pacs.002.001.10.xsd
  - pacs.003.001.08.xsd
  - pacs.004.001.09.xsd
  - pacs.008.001.08.xsd
  - pacs.008.001.08\_stp.xsd
  - pacs.009.001.08.xsd
  - pacs.009.001.08\_cov.xsd
  - pacs.009.001.08\_adv.xsd
  - pacs.010.001.03.xsd
  - pacs.010.001.03\_mgn.xsd
  - pain.001.001.09.xsd
  - pain.002.001.10.xsd
  - pain.008.001.08.xsd
  - head.001.001.02.camt\_029.xsd
  - head.001.001.02.camt\_052.xsd
  - head.001.001.02.camt\_053.xsd
  - head.001.001.02.camt\_054.xsd
  - head.001.001.02.camt\_055.xsd
  - head.001.001.02.camt\_056.xsd
  - head.001.001.02.camt\_057.xsd
  - head.001.001.02.camt\_058.xsd
  - head.001.001.02.camt\_060.xsd
  - head.001.001.02.camt\_105.xsd
  - head.001.001.02.camt\_105\_mlp.xsd
  - head.001.001.02.camt\_106.xsd
  - head.001.001.02.camt\_106\_mlp.xsd
  - head.001.001.02.camt\_107.xsd
  - head.001.001.02.camt\_108.xsd
  - head.001.001.02.camt\_109.xsd
  - head.001.001.02.pacs\_002.xsd
  - head.001.001.02.pacs\_003.xsd
  - head.001.001.02.pacs\_004.xsd
  - head.001.001.02.pacs\_008.xsd
  - head.001.001.02\_pacs\_008\_stp.xsd
  - head.001.001.02.pacs\_009.xsd
  - head.001.001.02\_pacs\_009\_cov.xsd
  - head.001.001.02\_pacs\_009\_adv.xsd
  - head.001.001.02.pacs\_010.xsd
  - head.001.001.02.pacs\_010\_mgn.xsd
  - head.001.001.02.pain\_001.xsd
  - head.001.001.02.pain\_002.xsd
  - head.001.001.02.pain\_008.xsd

- Trees:

The trees directory contains the following files:

- swiftroute\_funds.mtt  
Metadata that is used as an internal element placeholder.
- cbprErrorReport.mtt  
Metadata that represents the xml based structure of the validation report.

- Data:

The data directory contains the following files:

- mxconfig.xml  
Contains the MX configuration information on how to process the message.

Note: The mxconfig.xml file is not used in schema validation and can be ignored.

- Sample CBPR+ valid files for test purpose with header envelope:

- bah\_pacs\_002\_001\_10\_valid.xml
- bah\_pacs\_003\_001\_08\_valid.xml
- bah\_pacs\_004\_001\_09\_valid.xml
- bah\_pacs\_008\_001\_08\_valid.xml
- bah\_pacs\_008\_001\_08\_stp\_valid.xml
- bah\_pacs\_009\_001\_08\_valid.xml
- bah\_pacs\_009\_001\_08\_cov\_valid.xml
- bah\_pacs\_009\_001\_08\_adv\_valid.xml
- bah\_pacs\_010\_001\_03\_valid.xml
- bah\_pacs\_010\_001\_03\_mgn\_valid.xml
- bah\_camt\_029\_001\_09\_valid.xml
- bah\_camt\_052\_001\_08\_valid.xml
- bah\_camt\_053\_001\_08\_valid.xml
- bah\_camt\_054\_001\_08\_valid.xml
- bah\_camt\_055\_001\_08\_valid.xml
- bah\_camt\_056\_001\_08\_valid.xml
- bah\_camt\_057\_001\_06\_valid.xml
- bah\_camt\_058\_001\_08\_valid.xml
- bah\_camt\_060\_001\_05\_valid.xml
- bah\_camt\_105\_001\_02\_valid.xml
- bah\_camt\_105\_001\_02\_mlp\_valid.xml
- bah\_camt\_106\_001\_02\_valid.xml
- bah\_camt\_106\_001\_02\_mlp\_valid.xml
- bah\_camt\_107\_001\_01\_valid.xml
- bah\_camt\_108\_001\_01\_valid.xml
- bah\_camt\_109\_001\_01\_valid.xml
- bah\_pain\_001\_001\_09\_valid.xml
- bah\_pain\_002\_001\_10\_valid.xml
- bah\_pain\_008\_001\_08\_valid.xml

---

## Run the example in Design Studio

1. Build all the maps in the .mms files listed in What the example contains section.

2. Replace the input card 1 in router map, cbpr6100\_val (under cbpr\_schema\_only\_frmwrk\_map\_xsd.mms file) with desired input file.

3. Run the main router map, cbpr6100\_val.

You will see the output file in the data folder called cbpr\_error\_report.xml.

---

## Run the example in Design Server User Interface

1. Open a browser pointing to the installation of the IBM Sterling Transformation Extender.

2. Enter the user and password credentials.

3. If the project does not exist, import cbpr\_validation\_map.zip.

4. If absent, create a package containing all the Maps, Files and Flows.

5. Build the package in the desired server (to build all maps).

6. Open the project.

7. In the Maps tab, open the map cbpr6100\_val.

8. Modify the settings for input card 1 on design canvas to point to the desired test file.

9. Save, build, and run the map cbpr6100\_val.

10. Right click on the output card on canvas and select View data.

---

## CBPR+ Translation Examples:

- [CBPR+ Translation Examples using flows:](#)
- [CBPR+ Translation Examples using maps:](#)

## CBPR+ Translation Examples using flows:

- [\*\*CBPR+ MX to MT \(camt.029.001.09 to MT196/MT296\) Translation Example\*\*](#)  
This example demonstrates the CBPR MX to MT (camt.029.001.09 to MT196/MT296) message translation using flow server.
- [\*\*CBPR+ MX to MT \(camt.052.001.08 to MT942\) Translation Example\*\*](#)  
This example demonstrates the CBPR MX to MT (camt.052.001.08 to MT942) message translation using flow server.
- [\*\*CBPR+ MX to MT \(camt.053.001.08 to MT940\) Translation Example\*\*](#)  
This example demonstrates the CBPR+ MX to MT (camt.053.001.08 to MT940) message translation using maps.
- [\*\*CBPR+ MX to MT \(camt.054.001.08 to MT900/MT910\) Translation Example\*\*](#)  
This example demonstrates the CBPR MX to MT (camt.054.001.08 to MT900/MT910) message translation using flow server.
- [\*\*CBPR+ MX to MT \(camt.056.001.08 to MT192/MT292\) Translation Example\*\*](#)  
This example demonstrates the CBPR MX to MT (camt.056.001.08 to MT192/MT292) message translation using flow server.
- [\*\*CBPR+ MX to MT \(camt.057.001.06 to MT210\) Translation Example\*\*](#)  
This example demonstrates the CBPR MX to MT (camt.057.001.06 to MT210) message translation using flow server.
- [\*\*CBPR+ MX to MT \(camt.058.001.08 to MT292\) Translation Example\*\*](#)  
This example demonstrates the CBPR MX to MT (camt.058.001.08 to MT292) message translation using flow server.
- [\*\*CBPR+ MX to MT \(camt.107.001.01 to MT110\) Translation Example\*\*](#)  
This example demonstrates the CBPR MX to MT (camt.107.001.01 to MT110) message translation using flow server.
- [\*\*CBPR+ MX to MT \(camt.108.001.01 to MT111\) Translation Example\*\*](#)  
This example demonstrates the CBPR MX to MT (camt.108.001.01 to MT111) message translation using flow server.
- [\*\*CBPR+ MX to MT \(camt.109.001.01 to MT112\) Translation Example\*\*](#)  
This example demonstrates the CBPR MX to MT (camt.109.001.01 to MT112) message translation using flow server.
- [\*\*CBPR+ MX to MT \(pacs.002.001.10 to MT199/MT299\) Translation Example\*\*](#)  
This example demonstrates the CBPR+ MX to MT (pacs.002.001.10 to MT199/MT299) message translation using flow server.
- [\*\*CBPR+ MX to MT \(pacs.004.001.09 to MTnnnRETN\) Translation Example\*\*](#)  
This example demonstrates the CBPR MX to MT (pacs.004.001.09 to MTnnnRETN) message translation using flow server.
- [\*\*CBPR+ MX to MT \(pacs.008.001.08 to MT103\) Translation Example\*\*](#)  
This example demonstrates the CBPR+ MX to MT (pacs.008.001.08 to MT103) message translation using flow server.
- [\*\*CBPR+ MX to MT \(pacs.009.001.08 to MT202\) Translation Example\*\*](#)  
This example demonstrates the CBPR+ MX to MT (pacs.009.001.08 to MT202) message translation using flow server.
- [\*\*CBPR+ MX to MT \(pacs.009.001.08 to MT205\) Translation Example\*\*](#)  
This example demonstrates the CBPR+ MX to MT (pacs.009.001.08 to MT205) message translation using flow server.
- [\*\*CBPR+ MT to MX \(MT900/MT910 to camt.054.001.08\) Translation Example\*\*](#)  
This example demonstrates the CBPR+ MT to MX (MT900/MT910 to camt.054.001.08) message translation using flow server.
- [\*\*CBPR+ MT to MX \(MT192/MT292 to camt.056.001.08\) Translation Example\*\*](#)  
This example demonstrates the CBPR+ MT to MX (MT192/MT292 to camt.056.001.08) message translation using flow server.
- [\*\*CBPR+ MT to MX \(MT196/MT296 to camt.029.001.09\) Translation Example\*\*](#)  
This example demonstrates the CBPR+ MT to MX (MT196/MT296 to camt.029.001.09) message translation using flow server.
- [\*\*CBPR+ MT to MX \(MT103 to pacs.008.001.08 OR MT103 RETN to pacs.004.001.09\) Translation Example\*\*](#)  
This example demonstrates the CBPR+ MT to MX (MT103 to pacs.008.001.08 OR MT103 RETN to pacs.004.001.09) message translation using flow server.
- [\*\*CBPR+ MT to MX \(MT103 SWIFTGo to pacs.008.001.08\) Translation Example\*\*](#)  
This example demonstrates the CBPR+ MT to MX (MT103 SWIFTGo to pacs.008.001.08) message translation using flow server.
- [\*\*CBPR+ MT to MX \(MT202 ADV/COR/COV to pacs.009.001.08 OR MT202 RETN to pacs.004.001.09\) Translation Example\*\*](#)  
This example demonstrates the CBPR+ MT to MX (MT202 ADV/COR/COV to pacs.009.001.08 OR MT202 RETN to pacs.004.001.09) message translation using flow server.
- [\*\*CBPR+ MT to MX \(MT205 COR/COV to pacs.009.001.08 OR MT205 RETN to pacs.004.001.09\) Translation Example\*\*](#)  
This example demonstrates the CBPR+ MT to MX (MT205 COR/COV to pacs.009.001.08 OR MT205 RETN to pacs.004.001.09) message translation using flow server.
- [\*\*CBPR+ MT message validation Example\*\*](#)  
This example shows how flow server can be used for validating any Swift MT message.

## CBPR+ MX to MT (camt.029.001.09 to MT196/MT296) Translation Example

This example demonstrates the CBPR MX to MT (camt.029.001.09 to MT196/MT296) message translation using flow server.

- [\*\*What the example contains\*\*](#)  
Files included in this example are as follows:
- [\*\*How to run the example\*\*](#)  
This mx-xt translation will use the sample files to demonstrate the generation of SWIFT MT196 or MT296 message output from a CBPR+ camt.029.001.09 XML message.

## What the example contains

Files included in this example are as follows:

- cbpr\_translation.zip
  - Files:
    - env\_camt\_029\_mt196\_valid.xml
    - env\_camt\_029\_mt296\_valid.xml
    - trx\_config.xml
  - Schemas:
    - head.001.001.02\_camt\_029.xsd

- camt.029.001.09.xsd
  - trx\_config.xsd
  - infoset.mtt
  - swift\_iso7775\_ccyy.mtt
  - swiftroute\_funds.mtt
  - Maps:
    - cbpr2520\_camt029\_framework
    - cbpr2522\_camt029\_translate
    - cbpr2523\_camt029\_mtn96
    - cbpr2501\_mxmt\_setvarlog
  - Flows:
    - cbpr\_camt029\_mtn96
  - cbprJnodesConfigIBM.tar.gz
- 

## How to run the example

This mx-xt translation will use the sample files to demonstrate the generation of SWIFT MT196 or MT296 message output from a CBPR+ camt.029.001.09 XML message.

The cbprJnodesConfigIBM.tar.gz file is available either in IBM\_financialpaymentsplus\_vn.n.n.n.zip or in UIProjectImports directory.

Extract from cbprJnodesConfigIBM.tar.gz file.

- jnodes0.jar

The following jars needs to be copied from <TX\_install\_dir>/jars:

- jackson-core-n.n.n.jar
- jackson-annotations-n.n.n.jar
- jackson-databind-n.n.n.jar

Or visit <https://repo1.maven.org/maven2/com/fasterxml/jackson/core/> for download.

Note: Recommend using the latest version.

For the non Docker environments,

- Copy jars to <TX\_install\_dir>/extjar.

For TX V11.0.1 and up, native based Design Server installation,

Copy the following into the directory defined in config.yaml server.persistence.libs, by default, this is set to /opt/txlibs:

- jvcwrap.jar
- jvalccyy.jar

Restart the running application ./ITX stop and then ./ITX start.

For the Docker environments,

- docker cp jnodes0.jar tx-server:/opt/ibm/wsdtx/libs/
- Restart the design server, i.e., **docker restart** tx-server.

Note: For RHEL, Docker is not supported, Podman can be used as a substitute since it provides a command line interface similar to Docker. Below is an example of using podman:

- podman cp jnodes0.jar <brand>-server:/opt/<company>/<brand>/libs/
- Restart the design server, i.e., **podman restart**<brand>-server.

This example generates MT196 or MT296 based on the <OrgnlMsgNmId>.

Also, may generate audit.log.json log file to report the translation failure due to the pre-conversion checks.

1. Import the cbpr\_translation.zip project into the Design Server.
2. Open the cbpr\_translation project in Design Server and view the flow cbpr\_camt029\_mtn96.
  - a. The Flow Description points to all the maps to be executed during the flow process, which will be called from within some of the map nodes in the flow. The following is a list of maps invoked using RUN built-in function during the flow process:
    - @packagemap=cbpr\_plus/translation/mx-xt/maps/cbpr\_t9n\_mx\_camt029\_translate/cbpr2522\_camt029\_translate
    - @packagemap=cbpr\_plus/translation/mx-xt/maps/cbpr\_t9n\_mx\_camt029\_mtn96/cbpr2523\_camt029\_mtn96
    - @packagemap=cbpr\_plus/translation/mx-xt/maps/cbpr\_t9n\_mx\_setvarlog/cbpr2501\_mxmt\_setvarlog
  - b. It utilizes the following nodes:
    - i. Source Node
      - camt\_029
 

This node identifies the input data to be translated in the flow. It uses the INPUT\_FILE variable to set the location of the data.
    - ii. Map Node
      - mx\_translate
 

Runs framework map cbpr2520\_camt029\_framework, checks pre-translation conditions and translates the input file (mx or xml) into required output (mt).
    - iii. Target Node
      - mtn96
 

This node contains the resulting translation in MT (MT196/MT296) format and creates the output as defined by the variable OUTPUT\_RESULT\_MT.

iv. Log Node

- audit\_log

This node creates a log file specified by the flow variable AUDIT\_LOG.

c. It utilizes the following flow variables:

- INPUT\_FILE

Default value for the flow variable INPUT\_FILE is ..//cbpr\_plus/translation/mx-  
mt/data/env\_camt\_029\_mt196\_valid.xml. This is the data file to be used for translation and can be customized. It is used in the Source node camt\_029

- OUTPUT\_RESULT\_MT

Default value for the flow variable OUTPUT\_RESULT\_MT is mtn96.out. This is the location of the output file in mt format. It is used in the Target node mtn96. It can be customized.

- AUDIT\_LOG

Default value for the flow variable AUDIT\_LOG is audit.log.json. This is the location of the audit log. It is used in the Log node audit\_log. It can be customized.

- CONFIG\_FILE

Default value for the flow variable CONFIG\_FILE is ..//cbpr\_plus/translation/mx-  
mt/data/trx\_config.xml. This is the location of the file containing the configuration settings. It can be customized.

3. Open the flow cbpr\_camt029\_mtn96 in the Design Server. It utilizes one source node, one map node, one log node, and one target node.

4. In Design Server, create a package that contains the input files and one flow. The maps will automatically be included during deployment of the package onto the runtime server.

- [Run the example using three different methods](#)

You can run the example using three different methods:

---

## Run the example using three different methods

You can run the example using three different methods:

1. Running the example from the Design Server user interface.

2. Deploying the example to tx-rest and running it using the Swagger interface.

3. Deploying locally or to ftp server and using the flow command server process (flowcmdserver).

- [Run the example from the Design Server user interface](#)

Use the following steps to run the example from the Design Server user interface.

- [Run the example using the Swagger interface](#)

To run the example, deploy to tx-rest and run it using the Swagger interface.

- [Run the example using the flow command server process \(flowcmdserver\)](#)

To run the example, deploy locally or to ftp server and use the flow command server process (flowcmdserver).

---

## Run the example from the Design Server user interface

Use the following steps to run the example from the Design Server user interface.

1. In the Design Server, open cbpr\_camt029\_mtn96 flow and click on Run button.

2. Set toggle switch Run On Design Server and select flow input file env\_camt\_029\_mt196\_valid.xml.

Note: If a default path is assigned to the variable INPUT\_FILE, not selecting an input file will use the value defined for the variable.

Flow will run and report with the green check box. The report lists execution of all nodes.

3. Right click and select View Data on each link to examine the data.

4. Right click on the node and select View Log to see the logs.

---

## Run the example using the Swagger interface

To run the example, deploy to tx-rest and run it using the Swagger interface.

1. Create a Web type server definition where the runtime tx-rest is installed, if you do not have a server definition to deploy in Design Server.

2. If not running, start tx-rest on the server: <DTX\_HOME>/restapi/tomcat/dtxtomcat start tx-rest.

3. Deploy the created package to the server definition in Design Server.

4. Bring up the tx-rest Swagger UI: <DTX\_HOME>/restapi/tomcat/dtxtomcat showdocs tx-rest.

5. In the Swagger Explore window, enter /tx-rest/v2/docs and click on the Explore button to view the deployed package.

You can see the cbpr\_camt029\_mtn96 section having two actions: A PUT /v2/run/cbpr\_camt029\_mtn96 and a POST /v2/run/cbpr\_camt029\_mtn96.

6. In the PUT action:

a. Expand the PUT action and select the Try it out button.

b. Leave the input and output sections empty.

c. Under flow\_vars section, delete the entire content and replace with the commands to run the flow, for example:

```
{
 "INPUT_FILE": "C:/tests_dir/data/env_camt_029_mt196_valid.xml",
 "OUTPUT_RESULT_MT": "C:/tests_dir/data/mtn96.txt",
 "AUDIT_LOG": "C:/tests_dir/data/audit_log.json"
}
```

- d. Click Execute blue bar for running the flow. When flow completes execution, 200 response code is shown in the Server Response section. You can see information about the process flow in the Response body.
7. Refresh the browser to delete the deployed package and get the Swagger Explore back to /tx-rest/openapi.json.  
There will be a DELETE /v2/packages/{name} action.
8. Expand the DELETE /v2/packages/{name} action and select the Try it out.
9. In the Name field, enter the name you gave it to the created package.
10. Select the true option from the stop query drop down, select the blue Execute bar.  
You can see a response of 204 with a timestamp, if it is successful.

## Run the example using the flow command server process (flowcmdserver)

To run the example, deploy locally or to ftp server and use the flow command server process (flowcmdserver).

1. Create a server definition where the runtime is installed, if you do not have a server local definition or ftp execution definition to deploy to, in Design Server.
2. In Design Server, deploy the created package to the server definition. For example, deploy to c:\ftpserver\deployment directory.
3. Execute the flow using the flow command server:

```
flowcmdserver.exe --dir c:\ftpserver\deployment\flows --flow cbpr_camt029_mt96.json
--var "INPUT_FILE=C:\ftpserver\deployment\cbpr_plus\translation\mx-mt\data\env_camt_029_mt196_valid.xml"
--var "OUTPUT_RESULT_MT=C:\ftpserver\deployment\cbpr_plus\translation\mx-mt\data\mt96.out"
--var "AUDIT_LOG=C:\ftpserver\deployment\cbpr_plus\translation\mx-mt\data\env_camt_029_mt196_valid_log.json"
--audit "C:\ftpserver\deployment\cbpr_plus\translation\mx-mt\data\env_camt_029_mt196_valid_adt.json" -ad
```

4. The flow will report status similar to:

```
***Starting flow command server

Flow UUID: 85eabe2c-2302-4543-b347-9e18d65c6816
The flow audit file is: "C:\ftpserver\deployment\cbpr_plus\translation\mx-
mt\data\env_camt_029_mt196_valid_adt.json"
Flow completed successfully
Elapsed time: 3048ms
```

5. Examine the flow output result file mt96.out and the audit file env\_camt\_029\_mt196\_valid\_adt.json.

## CBPR+ MX to MT (camt.052.001.08 to MT942) Translation Example

This example demonstrates the CBPR MX to MT (camt.052.001.08 to MT942) message translation using flow server.

- What the example contains**  
Files included in this example are as follows:
- How to run the example**  
This mx-xt translation will use the sample files to demonstrate the generation of SWIFT MT942 message output from a CBPR+ camt.052.001.08 XML message.

## What the example contains

Files included in this example are as follows:

- cbpr\_translation.zip
  - Files:
    - env\_camt\_052\_mt942\_valid.xml
    - env\_camt\_052\_mt942\_bad.xml
    - trx\_config.xml
  - Schemas:
    - head.001.001.02\_camt\_052.xsd
    - camt.052.001.08.xsd
    - trx\_config.xsd
    - infoset.mtt
    - swift\_iso7775\_ccyy.mtt
    - swiftroute\_funds.mtt
  - Maps:
    - cbpr2546\_camt052\_framework
    - cbpr2547\_camt052\_translate
    - cbpr2548\_camt052\_mt942
    - cbpr2501\_mxmt\_setvarlog
  - Flows:
    - cbpr\_camt052\_mt942
- cbprJnodesConfigIBM.tar.gz

## How to run the example

This mx-xt translation will use the sample files to demonstrate the generation of SWIFT MT942 message output from a CBPR+ camt.052.001.08 XML message.

The cbprJnodesConfigIBM.tar.gz file is available either in IBM\_financialpaymentsplus\_vn.n.n.n.zip or in UIProjectImports directory.

Extract from cbprJnodesConfigIBM.tar.gz file.

- jnodes0.jar

The following jars needs to be copied from <TX\_install\_dir>/jars:

- jackson-core-n.n.n.jar
- jackson-annotations-n.n.n.jar
- jackson-databind-n.n.n.jar

Or visit <https://repo1.maven.org/maven2/com/fasterxml/jackson/core/> for download.

Note: Recommend using the latest version.

For the non Docker environments,

- Copy jars to <TX\_install\_dir>/extjar.

For TX V11.0.1 and up, native based Design Server installation,

Copy the following into the directory defined in config.yaml server.persistence.libs, by default, this is set to /opt/txls:

- jvcwrap.jar
- jvalccyy.jar

Restart the running application ./ITX stop and then ./ITX start.

For the Docker environments,

- docker cp jnodes0.jar <brand>-server:/opt/<company>/<brand>/libs/.
- Restart the design server, i.e., **docker restart** <brand>-server.

Note: For RHEL, Docker is not supported, Podman can be used as a substitute since it provides a command line interface similar to Docker. Below is an example of using podman.

- podman cp jnodes0.jar <brand>-server:/opt/<company>/<brand>/libs/.
- Restart the design server, i.e., **podman restart** <brand>-server.

This example may generate failure in the audit.log.json log file to report the translation failure due to the pre-conversion checks:

<ElctrncSeqNb> must be present in <BkToCstmrAcctRpt><Rpt><ElctrncSeqNb> and its length must be less than or equal to 5.

OR

<LgISeqNb> must be present in <BkToCstmrAcctRpt><Rpt><LgISeqNb> and its length must be less than or equal to 5.

There must be a maximum 190 occurrence of <Ntry> in <BkToCstmrAcctRpt><Rpt><Ntry>.

There must be present any ONE of the following:

- <BkToCstmrAcctRpt><Rpt><Ntry>
- <BkToCstmrAcctRpt><Rpt><TxSummry><TtlCdtNtries><NbOfNtries> AND <BkToCstmrAcctRpt><Rpt><TxSummry><TtlCdtNtries><Sum>
- <BkToCstmrAcctRpt><Rpt><TxSummry><TtlDbtNtries><NbOfNtries> AND <BkToCstmrAcctRpt><Rpt><TxSummry><TtlDbtNtries><Sum>

For all occurrence of <BkToCstmrAcctRpt><Rpt><Ntry>, value of <Ccy> in <BkToCstmrAcctRpt><Rpt><Ntry><Amt><Ccy> must be equal to value of <Ccy> in <BkToCstmrAcctRpt><Rpt><Acct><Ccy>, TotalNumberOfDigits <BkToCstmrAcctRpt><Rpt><Ntry><Amt> must be less than or equal to 14.

1. Import the cbpr\_translation.zip project into the Design Server.

2. Open the cbpr\_translation project in Design Server and view the flow cbpr\_camt052\_mt942.

a. The Flow Description points to all the maps to be executed during the flow process, which will be called from within some of the map nodes in the flow. The following is a list of maps invoked using RUN built-in function during the flow process:

- @packagemap=cbpr\_plus/translation/mx-ml/maps/cbpr\_t9n\_mx\_camt052\_translate/cbpr2547\_camt052\_translate
- @packagemap=cbpr\_plus/translation/mx-ml/maps/cbpr\_t9n\_mx\_camt052\_mt942/cbpr2548\_camt052\_mt942
- @packagemap=cbpr\_plus/translation/mx-ml/maps/cbpr\_t9n\_mx\_setvarlog/cbpr2501\_mxmt\_setvarlog

b. It utilizes the following nodes:

i. Source Node

- camt\_052

This node identifies the input data to be translated in the flow. It uses the INPUT\_FILE variable to set the location of the data.

ii. Map Node

- mx\_translate

Runs framework map cbpr2546\_camt052\_framework, checks pre-translation conditions and translates the input file (mx or xml) into required output (mt).

iii. Target Node

- mt942

This node contains the resulting translation in MT format and creates the output as defined by the variable OUTPUT\_RESULT\_MT.

iv. Log Node

- audit\_log

This node creates a log file specified by the flow variable AUDIT\_LOG.

c. It utilizes the following flow variables:

- INPUT\_FILE

Default value for the flow variable INPUT\_FILE is ..//cbpr\_plus/translation/mx-ml/data/env\_camt\_052\_mt942\_valid.xml. This is the data file to be used for translation and can be customized. It is used in the Source node camt\_052.

- **OUTPUT\_RESULT\_MT**  
Default value for the flow variable OUTPUT\_RESULT\_MT is mt942.out. This is the location of the output file in mt format. It is used in the Target node mt942. It can be customized.
- **AUDIT\_LOG**  
Default value for the flow variable AUDIT\_LOG is audit.log.json. This is the location of the audit log. It is used in the Log node audit\_log. It can be customized.
- **CONFIG\_FILE**  
Default value for the flow variable CONFIG\_FILE is ..\cbpr\_plus\translation\mx-mt\data\trx\_config.xml. This is the location of the file containing the configuration settings. It can be customized.

3. Open the flow cbpr\_camt052\_mt942 in the Design Server, It utilizes one source node, one map node, one log node, and one target node.
4. In Design Server, create a package that contains the input files and one flow. The maps will automatically be included during deployment of the package onto the runtime server.

- **[Run the example using three different methods](#)**

You can run the example using three different methods:

## Run the example using three different methods

You can run the example using three different methods:

1. Running the example from the Design Server user interface.
2. Deploying the example to tx-rest and running it using the Swagger interface.
3. Deploying locally or to ftp server and using the flow command server process (flowcmdserver).

- **[Run the example from the Design Server user interface](#)**

Use the following steps to run the example from the Design Server user interface.

- **[Run the example using the Swagger interface](#)**

To run the example, deploy to tx-rest and run it using the Swagger interface.

- **[Run the example using the flow command server process \(flowcmdserver\)](#)**

To run the example, deploy locally or to ftp server and use the flow command server process (flowcmdserver).

## Run the example from the Design Server user interface

Use the following steps to run the example from the Design Server user interface.

1. In the Design Server, open cbpr\_camt052\_mt942 flow and click on Run button.
2. Set toggle switch Run On Design Server and select flow input file env\_camt\_052\_mt942\_valid.xml.  
Note: If a default path is assigned to the variable INPUT\_FILE, not selecting an input file will use the value defined for the variable.  
Flow will run and report with the green check box. The report lists execution of all nodes.
3. Right click and select View Data on each link to examine the data.
4. Right click on the node and select View Log to see the logs.

## Run the example using the Swagger interface

To run the example, deploy to tx-rest and run it using the Swagger interface.

1. Create a Web type server definition where the runtime tx-rest is installed, if you do not have a server definition to deploy in Design Server.
2. If not running, start tx-rest on the server: <DTX\_HOME>/restapi/tomcat/dtxtomcat start tx-rest.
3. Deploy the created package to the server definition in Design Server.
4. Bring up the tx-rest Swagger UI: <DTX\_HOME>/restapi/tomcat/dtxtomcat showdocs tx-rest.
5. In the Swagger Explore window, enter /tx-rest/v2/docs and click on the Explore button to view the deployed package.  
You can see the cbpr\_camt052\_mt942 section having two actions: A PUT /v2/run/cbpr\_camt052\_mt942 and a POST /v2/run/cbpr\_camt052\_mt942.
6. In the PUT action:
  - a. Expand the PUT action and select the Try it out button.
  - b. Leave the input and output sections empty.
  - c. Under flow\_vars section, delete the entire content and replace with the commands to run the flow, for example:

```
{
 "INPUT_FILE": "C:/tests_dir/data/env_camt_052_mt942_valid.xml",
 "OUTPUT_RESULT_MT": "C:/tests_dir/data/mt942.txt",
 "AUDIT_LOG": "C:/tests_dir/data/audit_log.json"
}
```

- d. Click Execute blue bar for running the flow. When flow completes execution, 200 response code is shown in the Server Response section. You can see information about the process flow in the Response body.
7. Refresh the browser to delete the deployed package and get the Swagger Explore back to /tx-rest/openapi.json.  
There will be a DELETE /v2/packages/{name} action.
8. Expand the DELETE /v2/packages/{name} action and select the Try it out.
9. In the Name field, enter the name you gave it to the created package.
10. Select the true option from the stop query drop down, select the blue Execute bar.  
You can see a response of 204 with a timestamp, if it is successful.

## Run the example using the flow command server process (flowcmdserver)

To run the example, deploy locally or to ftp server and use the flow command server process (flowcmdserver).

1. Create a server definition where the runtime is installed, if you do not have a server local definition or ftp execution definition to deploy to, in Design Server.
2. In Design Server, deploy the created package to the server definition. For example, deploy to c:\ftpserver\deployment directory.
3. Execute the flow using the flow command server:

```
flowcmdserver.exe --dir c:\ftpserver\deployment\flows --flow cbpr_camt052_mt942.json
--var "INPUT_FILE=C:\ftpserver\deployment\cbpr_plus\translation\mx-mt\data\env_camt_052_mt942_valid.xml"
--var "OUTPUT_RESULT_MT=C:\ftpserver\deployment\cbpr_plus\translation\mx-mt\data\mt942.out"
--var "AUDIT_LOG=C:\ftpserver\deployment\cbpr_plus\translation\mx-mt\data\env_camt_052_mt942_valid_log.json"
--audit "C:\ftpserver\deployment\cbpr_plus\translation\mx-mt\data\env_camt_052_mt942_valid_adt.json" -ad
```

4. The flow will report status similar to:

```
***Starting flow command server

Flow UUID: 85eabe2c-2302-4543-b347-9e18d65c6816
The flow audit file is: "C:\\\\ftpserver\\\\deployment\\\\cbpr_plus\\\\translation\\\\mx-
mt\\\\data\\\\env_camt_052_mt942_valid_adt.json"
Flow completed successfully
Elapsed time: 3048ms
```

5. Examine the flow output result file mt942.out and the audit file env\_camt\_052\_mt942\_valid\_adt.json.

## CBPR+ MX to MT (camt.053.001.08 to MT940) Translation Example

This example demonstrates the CBPR+ MX to MT (camt.053.001.08 to MT940) message translation using maps.

- **What the example contains**

Files included in this example are as follows:

- **How to run the example**

This mx-mt translation will use the sample files to demonstrate the generation of SWIFT MT940 message output from a CBPR+ camt.053.001.08 XML message.

- **What the example contains**

Files included in this example are as follows:

- **How to run the example**

This mx-mt translation will use the sample files to demonstrate the generation of SWIFT MT940 message output from a CBPR+ camt.053.001.08 XML message.

## What the example contains

Files included in this example are as follows:

- cbpr\_translation.zip
  - Files:
    - env\_camt\_053\_mt940\_valid.xml
    - env\_camt\_053\_mt940\_bad.xml
    - trx\_config.xml
  - Schemas:
    - head.001.001.02\_camt\_053.xsd
    - camt.053.001.08.xsd
    - trx\_config.xsd
    - infoSet.mtt
    - swift\_iso7775\_ccyy.mtt
    - swiftRoute\_funds.mtt
  - Maps:
    - cbpr2530\_camt053\_framework
    - cbpr2531\_camt053\_translate
    - cbpr2534\_camt053\_mt940
    - cbpr2501\_mxmt\_setvarlog
  - Flows:
    - cbpr\_camt053\_mt940
- cbprJnodesConfigIBM.tar.gz

## How to run the example

This mx-mt translation will use the sample files to demonstrate the generation of SWIFT MT940 message output from a CBPR+ camt.053.001.08 XML message.

The cbprJnodesConfigIBM.tar.gz file is available either in IBM\_financialpaymentsplus\_vn.n.n.n.zip or in UIProjectImports directory.

Extract from cbprJnodesConfigIBM.tar.gz file.

- jnodes0.jar

The following jars needs to be copied from <TX\_install\_dir>/jars:

- jackson-core-n.n.n.jar
- jackson-annotations-n.n.n.jar
- jackson-databind-n.n.n.jar

Or visit <https://repo1.maven.org/maven2/com/fasterxml/jackson/core/> for download.

Note: Recommend using the latest version.

For the non Docker environments,

- Copy jars to <TX\_install\_dir>/extjar.

For TX V11.0.1 and up, native based Design Server installation,

Copy the following into the directory defined in config.yaml server.persistence.libs, by default, this is set to /opt/txls:

- jvcwrap.jar
- jvalccyy.jar

Restart the running application ./ITX stop and then ./ITX start.

For the Docker environments,

- docker cp jnodes0.jar <brand>-server:/opt/<company>/<brand>/libs/.
- Restart the design server, i.e., **docker restart <brand>-server**.

Note: For RHEL, Docker is not supported, Podman can be used as a substitute since it provides a command line interface similar to Docker. Below is an example of using podman.

- podman cp jnodes0.jar <brand>-server:/opt/<company>/<brand>/libs/.
- Restart the design server, i.e., **podman restart <brand>-server**.

This example may generate audit.log.json log file to report the translation failure due to the pre-conversion checks:

<ElctrncSeqNb> must be present in <BkToCstmrStmt><Stmt><ElctrncSeqNb> and its length must be less than or equal to 5.

OR

<LglSeqNb> must be present in <BkToCstmrStmt><Stmt><LglSeqNb> and its length must be less than or equal to 5.

If <PgNb> in <BkToCstmrStmt><Stmt><StmtPgntr><PgNb> is equal to 1 then there must be only 1 occurrence of <Cd> with value OPBD in

<BkToCstmrStmt><Stmt><Bal><Tp><CdOrPrtry><Cd> and if <Cd> is present in <BkToCstmrStmt><Stmt><Bal><Tp><SubTp><Cd>, then the value must be different from INTM.

If <PgNb> in <BkToCstmrStmt><Stmt><StmtPgntr><PgNb> is greater than 1 then there must be only 1 occurrence of <Cd> with value OPBD in

<BkToCstmrStmt><Stmt><Bal><Tp><CdOrPrtry><Cd> and <Cd> must be present in <BkToCstmrStmt><Stmt><Bal><Tp><SubTp><Cd>, value must be INTM.

If <LastPgInd> in <BkToCstmrStmt><Stmt><StmtPgntr><LastPgInd> is equal to true then there must be only 1 occurrence of <Cd> with value CLBD in

<BkToCstmrStmt><Stmt><Bal><Tp><CdOrPrtry><Cd> and if <Cd> is present in <BkToCstmrStmt><Stmt><Bal><Tp><SubTp><Cd>, then the value must be different from INTM.

If <LastPgInd> in <BkToCstmrStmt><Stmt><StmtPgntr><LastPgInd> is equal to false then there must be only 1 occurrence of <Cd> with value CLBD in

<BkToCstmrStmt><Stmt><Bal><Tp><CdOrPrtry><Cd> and <Cd> must be present in <BkToCstmrStmt><Stmt><Bal><Tp><SubTp><Cd>, value must be INTM.

There must be only 1 occurrence of <Cd> with value CLAV in <BkToCstmrStmt><Stmt><Bal><Tp><CdOrPrtry><Cd>.

There must be a maximum 190 occurrence of <Ntry> in <BkToCstmrStmt><Stmt><Ntry>.

For all occurrence of <BkToCstmrStmt><Stmt><Ntry>, value of <Ccy> in @<BkToCstmrStmt><Stmt><Ntry><Amt><Ccy> must be equal to value of <Ccy> in <BkToCstmrStmt><Stmt><Acct><Ccy>, TotalNumberOfDigits <BkToCstmrStmt><Stmt><Ntry><Amt> must be less than or equal to 14.

For the occurrence of <BkToCstmrStmt><Stmt><Bal>, if <Cd> is present with value OPBD in

<BkToCstmrStmt><Stmt><Bal><Tp><CdOrPrtry><Cd> then first two char of currency code must be the same for other occurrence of <BkToCstmrStmt><Stmt><Bal> in <BkToCstmrStmt><Stmt><Bal><Tp><CdOrPrtry><Cd> with value CLBD OR CLAV.

1. Import the cbpr\_translation.zip project into the Design Server.

2. Open the cbpr\_translation project in Design Server and view the flow cbpr\_camt053\_mt940.

a. The Flow Description points to all the maps to be executed during the flow process, which will be called from within some of the map nodes in the flow. The following is a list of maps invoked using RUN built-in function during the flow process:

- @packagemap=cbpr\_plus/translation/mx-ml/maps/cbpr\_t9n\_mx\_camt053\_translate/cbpr2531\_camt053\_translate
- @packagemap=cbpr\_plus/translation/mx-ml/maps/cbpr\_t9n\_mx\_camt053\_mt940/cbpr2534\_camt053\_mt940
- @packagemap=cbpr\_plus/translation/mx-ml/maps/cbpr\_t9n\_mx\_setvarlog/cbpr2501\_mxmt\_setvarlog

b. It utilizes the following nodes:

i. Source Node

- camt\_053

This node identifies the input data to be translated in the flow. It uses the INPUT\_FILE variable to set the location of the data.

ii. Map Node

- mx\_translate

Runs framework map cbpr2530\_camt053\_framework, checks pre-translation conditions and translates the input file (mx or xml) into required output (mt).

iii. Target Node

- mt940  
This node contains the resulting translation in MT format and creates the output as defined by the variable OUTPUT\_RESULT\_MT.
- iv. Log Node
- audit\_log  
This node creates a log file specified by the flow variable AUDIT\_LOG.
- c. It utilizes the following flow variables:
- INPUT\_FILE  
Default value for the flow variable INPUT\_FILE is ..\cbpr\_plus\translation\mx-mt\data\env\_camt\_053\_mt940\_valid.xml. This is the data file to be used for translation and can be customized. It is used in the Source node camt\_053.
  - OUTPUT\_RESULT\_MT  
Default value for the flow variable OUTPUT\_RESULT\_MT is mt940.out. This is the location of the output file in mt format. It is used in the Target node mt940. It can be customized.
  - AUDIT\_LOG  
Default value for the flow variable AUDIT\_LOG is audit.log.json. This is the location of the audit log. It is used in the Log node audit\_log. It can be customized.
  - CONFIG\_FILE  
Default value for the flow variable CONFIG\_FILE is ..\cbpr\_plus\translation\mx-mt\data\trx\_config.xml. This is the location of the file containing the configuration settings. It can be customized.
3. Open the flow cbpr\_camt053\_mt940 in the Design Server. It utilizes one source node, one map node, one log node, and one target node.
4. In Design Server, create a package that contains the input files and one flow. The maps will automatically be included during deployment of the package onto the runtime server.
- **[Run the example using three different methods](#)**  
You can run the example using three different methods:

## Run the example using three different methods

You can run the example using three different methods:

1. Running the example from the Design Server user interface.
  2. Deploying the example to tx-rest and running it using the Swagger interface.
  3. Deploying locally or to ftp server and using the flow command server process (flowcmdserver).
- **[Run the example from the Design Server user interface](#)**  
Use the following steps to run the example from the Design Server user interface.
  - **[Run the example using the Swagger interface](#)**  
To run the example, deploy to tx-rest and run it using the Swagger interface.
  - **[Run the example using the flow command server process \(flowcmdserver\)](#)**  
To run the example, deploy locally or to ftp server and use the flow command server process (flowcmdserver).

## Run the example from the Design Server user interface

Use the following steps to run the example from the Design Server user interface.

1. In the Design Server, open cbpr\_camt053\_mt940 flow and click on Run button.
2. Set toggle switch Run On Design Server and select flow input file env\_camt\_053\_mt940\_valid.xml.  
Note: If a default path is assigned to the variable INPUT\_FILE, not selecting an input file will use the value defined for the variable. Flow will run and report with the green check box. The report lists execution of all nodes.
3. Right click and select View Data on each link to examine the data.
4. Right click on the node and select View Log to see the logs.

## Run the example using the Swagger interface

To run the example, deploy to tx-rest and run it using the Swagger interface.

1. Create a Web type server definition where the runtime tx-rest is installed, if you do not have a server definition to deploy in Design Server.
2. If not running, start tx-rest on the server: <DTX\_HOME>/restapi/tomcat/dtxtomcat start tx-rest.
3. Deploy the created package to the server definition in Design Server.
4. Bring up the tx-rest Swagger UI: <DTX\_HOME>/restapi/tomcat/dtxtomcat showdocs tx-rest.
5. In the Swagger Explore window, enter /tx-rest/v2/docs and click on the Explore button to view the deployed package.  
You can see the cbpr\_camt053\_mt940 section having two actions: A PUT /v2/run/cbpr\_camt053\_mt940 and a POST /v2/run/cbpr\_camt053\_mt940.
6. In the PUT action:
  - a. Expand the PUT action and select the Try it out button.
  - b. Leave the input and output sections empty.
  - c. Under flow\_vars section, delete the entire content and replace with the commands to run the flow, for example:

```
{
 "INPUT_FILE": "C:/tests_dir/data/env_camt_053_mt940_valid.xml",
 "OUTPUT_RESULT_MT": "C:/tests_dir/data/mt940.txt",
```

```
"AUDIT_LOG": "C:/tests_dir/data/audit_log.json"
}
```

- d. Click Execute blue bar for running the flow. When flow completes execution, 200 response code is shown in the Server Response section. You can see information about the process flow in the Response body.
7. Refresh the browser to delete the deployed package and get the Swagger Explore back to /tx-rest/openapi.json.  
There will be a DELETE /v2/packages/{name} action.
8. Expand the DELETE /v2/packages/{name} action and select the Try it out.
9. In the Name field, enter the name you gave it to the created package.
10. Select the true option from the stop query drop down, select the blue Execute bar.  
You can see a response of 204 with a timestamp, if it is successful.

---

## Run the example using the flow command server process (flowcmdserver)

To run the example, deploy locally or to ftp server and use the flow command server process (flowcmdserver).

1. Create a server definition where the runtime is installed, if you do not have a server local definition or ftp execution definition to deploy to, in Design Server.
2. In Design Server, deploy the created package to the server definition. For example, deploy to c:\ftpserver\deployment directory.
3. Execute the flow using the flow command server:

```
flowcmdserver.exe --dir c:\ftpserver\deployment\flows --flow cbpr_camt053_mt940.json
--var "INPUT_FILE=C:\ftpserver\deployment\cbpr_plus\translation\mx-mt\data\env_camt_053_mt940_valid.xml"
--var "OUTPUT_RESULT_MT=C:\ftpserver\deployment\cbpr_plus\translation\mx-mt\data\mt940.out"
--var "AUDIT_LOG=C:\ftpserver\deployment\cbpr_plus\translation\mx-mt\data\env_camt_053_mt940_valid_log.json"
--audit "C:\ftpserver\deployment\cbpr_plus\translation\mx-mt\data\env_camt_053_mt940_valid_adt.json" -ad
```

4. The flow will report status similar to:

```
***Starting flow command server

Flow UUID: 85eabe2c-2302-4543-b347-9e18d65c6816
The flow audit file is: "C:\\ftpserver\\deployment\\cbpr_plus\\translation\\mx-
mt\\data\\env_camt_053_mt940_valid_adt.json"
Flow completed successfully
Elapsed time: 3048ms
```

5. Examine the flow output result file mt940.out and the audit file env\_camt\_053\_mt940\_valid\_adt.json.

---

## CBPR+ MX to MT (camt.054.001.08 to MT900/MT910) Translation Example

This example demonstrates the CBPR MX to MT (camt.054.001.08 to MT900/MT910) message translation using flow server.

- [What the example contains](#)  
Files included in this example are as follows:
- [How to run the example](#)  
This mx-mt translation will use the sample files to demonstrate the generation of SWIFT MT900 or MT910 message output from a CBPR+ camt.054.001.08 XML message.

---

## What the example contains

Files included in this example are as follows:

- cbpr\_translation.zip
  - Files:
    - env\_camt\_054\_bad.xml
    - env\_camt\_054\_crdt.xml
    - env\_camt\_054\_dbit.xml
    - trx\_config.xml
  - Schemas:
    - head.001.001.02\_camt\_054.xsd
    - camt.054.001.08.xsd
    - trx\_config.xsd
    - infoset.mtt
    - swift\_iso7775\_ccyy.mtt
    - swiftroute\_funds.mtt
  - Maps:
    - cbpr2510\_camt054\_framework
    - cbpr2512\_camt054\_translate
    - cbpr2513\_camt054\_mt900
    - cbpr2514\_camt054\_mt910
    - cbpr2501\_mxmt\_setvarlog
  - Flows:
    - cbpr\_camt054\_mt9n0
- cbprJnodesConfigIBM.tar.gz

## How to run the example

This mx-*mt* translation will use the sample files to demonstrate the generation of SWIFT MT900 or MT910 message output from a CBPR+ camt.054.001.08 XML message.

The cbprJnodesConfigIBM.tar.gz file is available either in IBM\_financialpaymentsplus\_vn.n.n.n.zip or in UIProjectImports directory.

Extract from cbprJnodesConfigIBM.tar.gz file.

- jnodes0.jar

The following jars needs to be copied from <TX\_install\_dir>/jars:

- jackson-core-n.n.n.jar
- jackson-annotations-n.n.n.jar
- jackson-databind-n.n.n.jar

Or visit <https://repo1.maven.org/maven2/com/fasterxml/jackson/core/> for download.

For the non Docker environments,

- Copy jars to <TX\_install\_dir>/extjar.

For TX V11.0.1 and up, native based Design Server installation,

Copy the following into the directory defined in config.yaml server.persistence.libs, by default, this is set to /opt/tlibs:

- jvcwrap.jar
- jvalccyy.jar

Restart the running application ./ITX stop and then ./ITX start.

For the Docker environments,

- docker cp jnodes0.jar <brand>-server:/opt/<company>/<brand>/libs/.
- Restart the design server, i.e., **docker restart** <brand>-server.

Note: For RHEL, Docker is not supported, Podman can be used as a substitute since it provides a command line interface similar to Docker. Below is an example of using podman:

- podman cp jnodes0.jar <brand>-server:/opt/<company>/<brand>/libs/.
- Restart the design server, i.e., **podman restart** <brand>-server.

This example generates MT900 or MT910 based on the <CdtDbtInd> which could be CRDT (will generate MT910) or DBIT (will generate MT900).

Also, may generate failure in the audit.log.json log file to report the translation failure due to the pre-conversion checks:

```
Either <CdtDbtInd> is not one of these (CRDT, DBIT) or <Sts><Cd> is not BOOK.
TotalNumberOfDigits <Ntry><Amt> less than or equal to 14.
Only one occurrence is allowed for <Ntfctn>, <Ntry>, <NtryDtls> and <TxDtls>.
Either one must exist <Ntfctn><Ntry><ValDt> or <Ntfctn><Ntry><NtryDtls><TxDtls><RltdDts><IntrBkSttlmDt>.
```

1. Import the cbpr\_translation.zip project into the Design Server.
2. Open the cbpr\_translation project in Design Server and view the flow cbpr\_camt054\_mt9n0.
  - a. The Flow Description points to all the maps to be executed during the flow process, which will be called from within some of the map nodes in the flow. The following is a list of maps invoked using RUN built-in function during the flow process:
    - @packagemap=cbpr\_plus/translation/mx-*mt*/maps/cbpr\_t9n\_mx\_camt054\_translate/cbpr2512\_camt054\_translate
    - @packagemap=cbpr\_plus/translation/mx-*mt*/maps/cbpr\_t9n\_mx\_camt054\_mt900/cbpr2513\_camt054\_mt900
    - @packagemap=cbpr\_plus/translation/mx-*mt*/maps/cbpr\_t9n\_mx\_camt054\_mt910/cbpr2514\_camt054\_mt910
    - @packagemap=cbpr\_plus/translation/mx-*mt*/maps/cbpr\_t9n\_mx\_setvarlog/cbpr2501\_mxmt\_setvarlog
  - b. It utilizes the following nodes:
    - i. Source Node
      - camt\_054  
This node identifies the input data to be translated in the flow. It uses the INPUT\_FILE variable to set the location of the data.
    - ii. Map Node
      - mx\_translate  
Runs framework map cbpr2510\_camt054\_framework, checks pre-translation conditions and translates the input file (mx or xml) into required output (mt).
    - iii. Target Node
      - mt9n0  
This node contains the resulting translation in MT format and creates the output as defined by the variable OUTPUT\_RESULT\_MT.
    - iv. Log Node
      - audit\_log  
This node creates a log file specified by the flow variable AUDIT\_LOG.
  - c. It utilizes the following flow variables:
    - INPUT\_FILE  
Default value for the flow variable INPUT\_FILE is ..//cbpr\_plus/translation/mx-*mt*/data/env\_camt\_054\_crdt.xml. This is the data file to be used for translation and can be customized. It is used in the Source node camt\_054.
    - OUTPUT\_RESULT\_MT

Default value for the flow variable OUTPUT\_RESULT\_MT is mt9n0.out. This is the location of the output file in mt format. It is used in Target node mt900 or mt910 . It can be customized.

- AUDIT\_LOG  
Default value for the flow variable AUDIT\_LOG is audit.log.json. This is the location of the audit log. It is used in the Log node audit\_log. It can be customized.
- CONFIG\_FILE  
Default value for the flow variable CONFIG\_FILE is ../cbpr\_plus/translation/mx-mt/data/trx\_config.xml. This is the location of the file containing the configuration settings. It can be customized.

3. Open the flow cbpr\_camt054\_mt9n0 in the Design Server. It utilizes one source node, one map node, one log node, and one target node.
4. In Design Server, create a package that contains the input files and one flow. The maps will automatically be included during deployment of the package onto the runtime server.

- **[Run the example using three different methods](#)**

You can run the example using three different methods:

---

## Run the example using three different methods

You can run the example using three different methods:

1. Running the example from the Design Server user interface.
  2. Deploying the example to tx-rest and running it using the Swagger interface.
  3. Deploying locally or to ftp server and using the flow command server process (flowcmdserver).
- **[Run the example from the Design Server user interface](#)**  
Use the following steps to run the example from the Design Server user interface.
  - **[Run the example using the Swagger interface](#)**  
To run the example, deploy to tx-rest and run it using the Swagger interface.
  - **[Run the example using the flow command server process \(flowcmdserver\)](#)**  
To run the example, deploy locally or to ftp server and use the flow command server process (flowcmdserver).

---

## Run the example from the Design Server user interface

Use the following steps to run the example from the Design Server user interface.

1. In the Design Server, open cbpr\_camt054\_mt9n0 flow and click on Run button.
2. Set toggle switch Run On Design Server and select flow input file env\_camt\_054\_dbit.xml.  
Note: If a default path is assigned to the variable INPUT\_FILE, not selecting an input file will use the value defined for the variable.  
Flow will run and report with the green check box. The report lists execution of all nodes.
3. Right click and select View Data on each link to examine the data.
4. Right click on the node and select View Log to see the logs.

---

## Run the example using the Swagger interface

To run the example, deploy to tx-rest and run it using the Swagger interface.

1. Create a Web type server definition where the runtime tx-rest is installed, if you do not have a server definition to deploy in Design Server.
2. If not running, start tx-rest on the server: <DTX\_HOME>/restapi/tomcat/dtxtomcat start tx-rest.
3. Deploy the created package to the server definition in Design Server.
4. Bring up the tx-rest Swagger UI: <DTX\_HOME>/restapi/tomcat/dtxtomcat showdocs tx-rest.
5. In the Swagger Explore window, enter /tx-rest/v2/docs and click on the Explore button to view the deployed package.  
You can see the cbpr\_camt054\_mt9n0 section having two actions: A PUT /v2/run/cbpr\_camt054\_mt9n0 and a POST /v2/run/cbpr\_camt054\_mt9n0.
6. In the PUT action:
  - a. Expand the PUT action and select the Try it out button.
  - b. Leave the input and output sections empty.
  - c. Under flow\_vars section, delete the entire content and replace with the commands to run the flow, for example:

```
{
 "INPUT_FILE": "C:/tests_dir/data/env_camt_054_dbit.xml",
 "OUTPUT_RESULT_MT": "C:/tests_dir/data/mt900.txt",
 "AUDIT_LOG": "C:/tests_dir/data/audit_log.json"
}
```

- d. Click Execute blue bar for running the flow. When flow completes execution, 200 response code is shown in the Server Response section. You can see information about the process flow in the Response body.
7. Refresh the browser to delete the deployed package and get the Swagger Explore back to /tx-rest/openapi.json.  
There will be a DELETE /v2/packages/{name} action.
8. Expand the DELETE /v2/packages/{name} action and select the Try it out.
9. In the Name field, enter the name you gave it to the created package.
10. Select the true option from the stop query drop down, select the blue Execute bar.  
You can see a response of 204 with a timestamp, if it is successful.

## Run the example using the flow command server process (flowcmdserver)

To run the example, deploy locally or to ftp server and use the flow command server process (flowcmdserver).

1. Create a server definition where the runtime is installed, if you do not have a server local definition or ftp execution definition to deploy to, in Design Server.
2. In Design Server, deploy the created package to the server definition. For example, deploy to c:\ftpserver\deployment directory.
3. Execute the flow using the flow command server:

```
flowcmdserver.exe --dir c:\ftpserver\deployment\flows --flow cbpr_camt054_mt9n0.json
--var "INPUT_FILE=C:\ftpserver\deployment\cbpr_plus\translation\mx-mt\data\env_camt_054_dbit.xml"
--var "OUTPUT_RESULT_MT=C:\ftpserver\deployment\cbpr_plus\translation\mx-mt\data\mt9n0.out"
--var "AUDIT_LOG=C:\ftpserver\deployment\cbpr_plus\translation\mx-mt\data\env_camt_054_dbit_log.json"
--audit "C:\ftpserver\deployment\cbpr_plus\translation\mx-mt\data\env_camt_054_dbit_adt.json" -ad
```

4. The flow will report status similar to:

```
***Starting flow command server

Flow UUID: 85eabe2c-2302-4543-b347-9e18d65c6816
The flow audit file is: "C:\\\\ftpserver\\\\deployment\\\\cbpr_plus\\\\translation\\\\mx-
mt\\\\data\\\\env_camt_054_dbit_adt.json"
Flow completed successfully
Elapsed time: 3048ms
```

5. Examine the flow output result file mt900.out and the audit file env\_camt\_054\_dbit\_adt.json.

## CBPR+ MX to MT (camt.056.001.08 to MT192/MT292) Translation Example

This example demonstrates the CBPR MX to MT (camt.056.001.08 to MT192/MT292) message translation using flow server.

- [What the example contains](#)

Files included in this example are as follows:

- [How to run the example](#)

This mx-mt translation will use the sample files to demonstrate the generation of SWIFT MT192 or MT292 message output from a CBPR+ camt.056.001.08 XML message.

## What the example contains

Files included in this example are as follows:

- cbpr\_translation.zip
  - Files:
    - env\_camt\_056\_mt192\_valid.xml
    - env\_camt\_056\_mt292\_valid.xml
    - trx\_config.xml
  - Schemas:
    - head.001.001.02\_camt\_056.xsd
    - camt.056.001.08.xsd
    - trx\_config.xsd
    - infoset.mtt
    - swift\_iso7775\_ccyy.mtt
    - swiftroute\_funds.mtt
  - Maps:
    - cbpr2540\_camt056\_framework
    - cbpr2542\_camt056\_translate
    - cbpr2543\_camt056\_mtn92
    - cbpr2501\_mxmt\_setvarlog
  - Flows:
    - cbpr\_camt056\_mtn92
- cbprJnodesConfigIBM.tar.gz

## How to run the example

This mx-mt translation will use the sample files to demonstrate the generation of SWIFT MT192 or MT292 message output from a CBPR+ camt.056.001.08 XML message.

The cbprJnodesConfigIBM.tar.gz file is available either in IBM\_financialpaymentsplus\_vn.n.n.zip or in UIProjectImports directory.

Extract from cbprJnodesConfigIBM.tar.gz file.

- jnodes0.jar

The following jars needs to be copied from <TX\_install\_dir>/jars:

- jackson-core-n.n.n.jar

- jackson-annotations-n.n.n.jar
- jackson-databind-n.n.n.jar

Or visit <https://repo1.maven.org/maven2/com/fasterxml/jackson/core/> for download.

Note: Recommend using the latest version.

For the non Docker environments,

- Copy jars to <TX\_install\_dir>/extjar.

For TX V11.0.1 and up, native based Design Server installation,

Copy the following into the directory defined in config.yaml server.persistence.libs, by default, this is set to /opt/txlibs:

- jvcwrap.jar
- jvalccyy.jar

Restart the running application ./ITX stop and then ./ITX start.

For the Docker environments,

- docker cp jnodes0.jar <brand>-server:/opt/<company>/<brand>/libs/.
- Restart the design server, i.e., **docker restart** <brand>-server.

Note: For RHEL, Docker is not supported, Podman can be used as a substitute since it provides a command line interface similar to Docker. Below is an example of using podman:

- podman cp jnodes0.jar <brand>-server:/opt/<company>/<brand>/libs/.
- Restart the design server, i.e., **podman restart** <brand>-server.

This example generates MT192 or MT292 based on the <Orgn1MsgNmId>.

Also, may generate failure in the audit.log.json log file to report the translation failure due to the pre-conversion checks.

1. Import the cbpr\_translation.zip project into the Design Server.
2. Open the cbpr\_translation project in Design Server and view the flow cbpr\_camt056\_mtn92.
  - a. The Flow Description points to all the maps to be executed during the flow process, which will be called from within some of the map nodes in the flow. The following is a list of maps invoked using RUN built-in function during the flow process:
    - @packagemap=cbpr\_plus/translation/mx-mt/maps/cbpr\_t9n\_mx\_camt056\_translate/cbpr2542\_camt056\_translate
    - @packagemap=cbpr\_plus/translation/mx-mt/maps/cbpr\_t9n\_mx\_camt056\_mtn92/cbpr2543\_camt056\_mtn92
    - @packagemap=cbpr\_plus/translation/mx-mt/maps/cbpr\_t9n\_mx\_setvarlog/cbpr2501\_mxmt\_setvarlog
  - b. It utilizes the following nodes:
    - i. Source Node
      - camt\_056
 

This node identifies the input data to be translated in the flow. It uses the INPUT\_FILE variable to set the location of the data.
    - ii. Map Node
      - mx\_translate
 

Runs framework map cbpr2540\_camt056\_framework, checks pre-translation conditions and translates the input file (mx or xml) into required output (mt).
    - iii. Target Node
      - mtn92
 

This node contains the resulting translation in MT format and creates the output as defined by the variable OUTPUT\_RESULT\_MT.
    - iv. Log Node
      - audit\_log
 

This node creates a log file specified by the flow variable AUDIT\_LOG.
- c. It utilizes the following flow variables:
  - INPUT\_FILE
 

Default value for the flow variable INPUT\_FILE is ..//cbpr\_plus/translation/mx-mt/data/env\_camt\_056\_mt192\_valid.xml. This is the data file to be used for translation and can be customized. It is used in the Source node camt\_056.
  - OUTPUT\_RESULT\_MT
 

Default value for the flow variable OUTPUT\_RESULT\_MT is mtn92.out. This is the location of the output file in mt format. It is used in the Target node mt192 or mt292. It can be customized.
  - AUDIT\_LOG
 

Default value for the flow variable AUDIT\_LOG is audit.log.json. This is the location of the audit log. It is used in the Log node audit\_log. It can be customized.
  - CONFIG\_FILE
 

Default value for the flow variable CONFIG\_FILE is ..//cbpr\_plus/translation/mx-mt/data/trx\_config.xml. This is the location of the file containing the configuration settings. It can be customized.
3. Open the flow cbpr\_camt056\_mtn92 in the Design Server. It utilizes one source node, one map node, one log node, and one target node.
4. In Design Server, create a package that contains the input files and one flow. The maps will automatically be included during deployment of the package onto the runtime server.
- **Run the example using three different methods**  
 You can run the example using three different methods:

## Run the example using three different methods

You can run the example using three different methods:

1. Running the example from the Design Server user interface.
2. Deploying the example to tx-rest and running it using the Swagger interface.
3. Deploying locally or to ftp server and using the flow command server process (flowcmdserver).

- [\*\*Run the example from the Design Server user interface\*\*](#)

Use the following steps to run the example from the Design Server user interface.

- [\*\*Run the example using the Swagger interface\*\*](#)  
To run the example, deploy to tx-rest and run it using the Swagger interface.
- [\*\*Run the example using the flow command server process \(flowcmdserver\)\*\*](#)  
To run the example, deploy locally or to ftp server and use the flow command server process (flowcmdserver).

---

## Run the example from the Design Server user interface

Use the following steps to run the example from the Design Server user interface.

1. In the Design Server, open cbpr\_camt056\_mtn92 flow and click on Run button.
2. Set toggle switch Run On Design Server and select flow input file env\_camt\_056\_mt192\_valid.xml.  
Note: If a default path is assigned to the variable INPUT\_FILE, not selecting an input file will use the value defined for the variable.  
Flow will run and report with the green check box. The report lists execution of all nodes.
3. Right click and select View Data on each link to examine the data.
4. Right click on the node and select View Log to see the logs.

---

## Run the example using the Swagger interface

To run the example, deploy to tx-rest and run it using the Swagger interface.

1. Create a Web type server definition where the runtime tx-rest is installed, if you do not have a server definition to deploy in Design Server.
2. If not running, start tx-rest on the server: <DTX\_HOME>/restapi/tomcat/dtxtomcat start tx-rest.
3. Deploy the created package to the server definition in Design Server.
4. Bring up the tx-rest Swagger UI: <DTX\_HOME>/restapi/tomcat/dtxtomcat showdocs tx-rest.
5. In the Swagger Explore window, enter /tx-rest/v2/docs and click on the Explore button to view the deployed package.

You can see the cbpr\_camt056\_mtn92 section having two actions: A PUT /v2/run/cbpr\_camt056\_mtn92 and a POST /v2/run/cbpr\_camt056\_mtn92.

6. In the PUT action:
  - a. Expand the PUT action and select the Try it out button.
  - b. Leave the input and output sections empty.
  - c. Under flow\_vars section, delete the entire content and replace with the commands to run the flow, for example:

```
{
 "INPUT_FILE": "C:/tests_dir/data/env_camt_056_mt192_valid.xml",
 "OUTPUT_RESULT_MT": "C:/tests_dir/data/mtn92.txt",
 "AUDIT_LOG": "C:/tests_dir/data/audit_log.json"
}
```

7. Click Execute blue bar for running the flow. When flow completes execution, 200 response code is shown in the Server Response section. You can see information about the process flow in the Response body.
8. Refresh the browser to delete the deployed package and get the Swagger Explore back to /tx-rest/openapi.json. There will be a DELETE /v2/packages/{name} action.
9. Expand the DELETE /v2/packages/{name} action and select the Try it out.
10. In the Name field, enter the name you gave it to the created package.

You can see a response of 204 with a timestamp, if it is successful.

---

## Run the example using the flow command server process (flowcmdserver)

To run the example, deploy locally or to ftp server and use the flow command server process (flowcmdserver).

1. Create a server definition where the runtime is installed, if you do not have a server local definition or ftp execution definition to deploy to, in Design Server.
2. In Design Server, deploy the created package to the server definition. For example, deploy to c:\ftpserver\deployment directory.
3. Execute the flow using the flow command server:

```
flowcmdserver.exe --dir c:\ftpserver\deployment\flows --flow cbpr_camt056_mtn92.json
--var "INPUT_FILE=C:\ftpserver\deployment\cbpr_plus\translation\mx-mt\data\env_camt_056_mt192_valid.xml"
--var "OUTPUT_RESULT_MT=C:\ftpserver\deployment\cbpr_plus\translation\mx-mt\data\mtn92.out"
--var "AUDIT_LOG=C:\ftpserver\deployment\cbpr_plus\translation\mx-mt\data\env_camt_056_mt192_valid_log.json"
--audit "C:\ftpserver\deployment\cbpr_plus\translation\mx-mt\data\env_camt_056_mt192_valid_adt.json" -ad
```

4. The flow will report status similar to:

```
***Starting flow command server

Flow UUID: 85eabe2c-2302-4543-b347-9e18d65c6816
The flow audit file is: "C:\\ftpserver\\deployment\\cbpr_plus\\translation\\mx-
mt\\data\\env_camt_056_mt192_valid_adt.json"
Flow completed successfully
Elapsed time: 3048ms
```

5. Examine the flow output result file `mtn92.out` and the audit file `env_camt_056_mt192_valid_adt.json`.

---

## CBPR+ MX to MT (camt.057.001.06 to MT210) Translation Example

This example demonstrates the CBPR MX to MT (camt.057.001.06 to MT210) message translation using flow server.

- **What the example contains**

Files included in this example are as follows:

- `cbpr_translation.zip`
  - Files:
    - `env_camt_057_bad.xml`
    - `env_camt_057_valid.xml`
    - `trx_config.xml`
  - Schemas:
    - `head.001.001.02_camt_057.xsd`
    - `camt.057.001.06.xsd`
    - `trx_config.xsd`
    - `infoset.mtt`
    - `swift_iso7775_ccyy.mtt`
    - `swiftroute_funds.mtt`
  - Maps:
    - `cbpr2600_camt057_framework`
    - `cbpr2601_camt057_translate`
    - `cbpr2605_camt057_mt210`
    - `cbpr2501_mxmt_setvarlog`
  - Flows:
    - `cbpr_camt057_mt210`
- `cbprJnodesConfigIBM.tar.gz`

---

## How to run the example

This mx-mt translation will use the sample files to demonstrate the generation of SWIFT MT210 message output from a CBPR+ camt.057.001.06 XML message.

The `cbprJnodesConfigIBM.tar.gz` file is available either in `IBM_financialpaymentsplus_vn.n.n.n.zip` or in `UIProjectImports` directory.

Extract from `cbprJnodesConfigIBM.tar.gz` file.

- `jnodes0.jar`

The following jars needs to be copied from `<TX_install_dir>/jars`:

- `jackson-core-n.n.n.jar`
- `jackson-annotations-n.n.n.jar`
- `jackson-databind-n.n.n.jar`

Or visit <https://repo1.maven.org/maven2/com/fasterxml/jackson/core/> for download.

Note: Recommend using the latest version.

For the non Docker environments,

- Copy jars to `<TX_install_dir>/extjar`.

For TX V11.0.1 and up, native based Design Server installation,

Copy the following into the directory defined in config.yaml `server.persistence.libs`, by default, this is set to `/opt/txlabs`:

- `jvcwrap.jar`
- `jvalccyy.jar`

Restart the running application `./ITX stop` and then `./ITX start`.

For the Docker environments,

- `docker cp jnodes0.jar <brand>-server:/opt/<company>/<brand>/libs/`.
- Restart the design server, i.e., `docker restart <brand>-server`.

Note: For RHEL, Docker is not supported, Podman can be used as a substitute since it provides a command line interface similar to Docker. Below is an example of using podman:

- `podman cp jnodes0.jar <brand>-server:/opt/<company>/<brand>/libs/`.

- Restart the design server, i.e., **podman restart <brand>-server**.

This example may generate failure in the audit.log.json log file to report the translation failure due to the pre-conversion checks:

```
NumberOfOccurrences <Ntfctn><Itm> should not be greater than 1.
Currency should not be any of these: {"XAU", "XAG", "XPD", "XPT"} for <Ntfctn><Itm><Amt><Ccy>
TotalNumberOfDigits <Ntry><Amt> less than or equal to 14.
```

1. Import the cbpr\_translation.zip project into the Design Server.
2. Open the cbpr\_translation project in Design Server and view the flow cbpr\_camt057\_mt210.
  - a. The Flow Description points to all the maps to be executed during the flow process, which will be called from within some of the map nodes in the flow. The following is a list of maps invoked using RUN built-in function during the flow process:
    - @packagemap=cbpr\_plus/translation/mx-mt/maps/cbpr\_t9n\_mx\_camt057\_translate/cbpr2601\_camt057\_translate
    - @packagemap=cbpr\_plus/translation/mx-mt/maps/cbpr\_t9n\_mx\_camt057\_mt210/cbpr2605\_camt057\_mt210
    - @packagemap=cbpr\_plus/translation/mx-mt/maps/cbpr\_t9n\_mx\_setvarlog/cbpr2501\_mxmt\_setvarlog
  - b. It utilizes the following nodes:
    - i. Source Node
      - camt\_057
 This node identifies the input data to be translated in the flow. It uses the INPUT\_FILE variable to set the location of the data.
    - ii. Map Node
      - mx\_translate
 Runs framework map cbpr2600\_camt057\_framework, checks pre-translation conditions and translates the input file (mx or xml) into required output (mt).
    - iii. Target Node
      - mt210
 This node contains the resulting translation in MT format and creates the output as defined by the variable OUTPUT\_RESULT\_MT.
    - iv. Log Node
      - audit\_log
 This node creates a log file specified by the flow variable AUDIT\_LOG.
  - c. It uses the following flow variables:
    - INPUT\_FILE
 Default value for the flow variable INPUT\_FILE is ..//cbpr\_plus/translation/mx-mt/data/env\_camt\_057\_valid.xml. This is the data file to be used for translation and can be customized. It is used in the Source node camt\_057
    - OUTPUT\_RESULT\_MT
 Default value for the flow variable OUTPUT\_RESULT\_MT is mt210.out. This is the location of the output file in mt format. It is used in the Target node mt210. It can be customized.
    - AUDIT\_LOG
 Default value for the flow variable AUDIT\_LOG is audit.log.json. This is the location of the audit log. It is used in the Log node audit\_log. It can be customized.
    - CONFIG\_FILE
 Default value for the flow variable CONFIG\_FILE is ..//cbpr\_plus/translation/mx-mt/data/trx\_config.xml. This is the location of the file containing the configuration settings. It can be customized.
3. Open the flow cbpr\_camt057\_mt210 in the Design Server. It utilizes one source node, one map node, one log node, and one target node.
4. In Design Server, create a package that contains the input files and one flow. The maps will automatically be included during deployment of the package onto the runtime server.

- **Run the example using three different methods**

You can run the example using three different methods:

## Run the example using three different methods

You can run the example using three different methods:

1. Running the example from the Design Server user interface.
2. Deploying the example to tx-rest and running it using the Swagger interface.
3. Deploying locally or to ftp server and using the flow command server process (flowcmdserver).

- **Run the example from the Design Server user interface**

Use the following steps to run the example from the Design Server user interface:

- **Run the example using the Swagger interface**

To run the example, deploy to tx-rest and run it using the Swagger interface.

- **Run the example using the flow command server process (flowcmdserver)**

To run the example, deploy locally or to ftp server and use the flow command server process (flowcmdserver).

---

## Run the example from the Design Server user interface

Use the following steps to run the example from the Design Server user interface:

1. In the Design Server, open cbpr\_camt057\_mt210 flow and click on Run button.
2. Set toggle switch Run On Design Server and select flow input file env\_camt\_057\_valid.xml.  
Note: If a default path is assigned to the variable INPUT\_FILE, not selecting an input file will use the value defined for the variable.

- Flow will run and report with the green check box. The report lists execution of all nodes.
3. Right click and select View Data on each link to examine the data.
  4. Right click on the node and select View Log to see the logs.

## Run the example using the Swagger interface

To run the example, deploy to tx-rest and run it using the Swagger interface.

1. Create a Web type server definition where the runtime tx-rest is installed, if you do not have a server definition to deploy in Design Server.
2. If not running, start tx-rest on the server: <DTX\_HOME>/restapi/tomcat/dtxtomcat start tx-rest.
3. Deploy the created package to the server definition in Design Server.
4. Bring up the tx-rest Swagger UI: <DTX\_HOME>/restapi/tomcat/dtxtomcat showdocs tx-rest.
5. In the Swagger Explore window, enter /tx-rest/v2/docs and click on the Explore button to view the deployed package.  
You can see the cbpr\_camt057\_mt210 section having two actions: A PUT /v2/run/cbpr\_camt057\_mt210 and a POST /v2/run/cbpr\_camt057\_mt210.
6. In the PUT action:
  - a. Expand the PUT action and select the Try it out button.
  - b. Leave the input and output sections empty.
  - c. Under flow\_vars section, delete the entire content and replace with the commands to run the flow, for example:

```
{
 "INPUT_FILE": "C:/tests_dir/data/env_camt_057_valid.xml",
 "OUTPUT_RESULT_MT": "C:/tests_dir/data/mt210.txt",
 "AUDIT_LOG": "C:/tests_dir/data/audit_log.json"
}
```

  - d. Click Execute blue bar for running the flow. When flow completes execution, 200 response code is shown in the Server Response section. You can see information about the process flow in the Response body.
7. Refresh the browser to delete the deployed package and get the Swagger Explore back to /tx-rest/openapi.json.  
There will be a DELETE /v2/packages/{name} action.
8. Expand the DELETE /v2/packages/{name} action and select the Try it out.
9. In the Name field, enter the name you gave it to the created package.
10. Select the true option from the stop query drop down, select the blue Execute bar.  
You can see a response of 204 with a timestamp, if it is successful.

## Run the example using the flow command server process (flowcmdserver)

To run the example, deploy locally or to ftp server and use the flow command server process (flowcmdserver).

1. Create a server definition where the runtime is installed, if you do not have a server local definition or ftp execution definition to deploy to, in Design Server.
2. In Design Server, deploy the created package to the server definition. For example, deploy to c:\ftpserver\deployment directory.
3. Execute the flow using the flow command server:

```
flowcmdserver.exe --dir c:\ftpserver\deployment\flows --flow cbpr_camt057_mt210.json
--var "INPUT_FILE=C:\ftpserver\deployment\cbpr_plus\translation\mx-mt\data\env_camt_057_valid.xml"
--var "OUTPUT_RESULT_MT=C:\ftpserver\deployment\cbpr_plus\translation\mx-mt\data\mt210.out"
--var "AUDIT_LOG=C:\ftpserver\deployment\cbpr_plus\translation\mx-mt\data\env_camt_057_valid_log.json"
--audit "C:\ftpserver\deployment\cbpr_plus\translation\mx-mt\data\env_camt_057_valid_adt.json" -ad
```

4. The flow will report status similar to:

```
***Starting flow command server

 Flow UUID: 85eabe2c-2302-4543-b347-9e18d65c6816
 The flow audit file is: "C:\\\\ftpserver\\\\deployment\\\\cbpr_plus\\\\translation\\\\mx-
 mt\\\\data\\\\env_camt_057_valid_adt.json"
 Flow completed successfully
 Elapsed time: 3048ms
```

5. Examine the flow output result file mt210.out and the audit file env\_camt\_057\_valid\_adt.json.

## CBPR+ MX to MT (camt.058.001.08 to MT292) Translation Example

This example demonstrates the CBPR MX to MT (camt.058.001.08 to MT292) message translation using flow server.

- [What the example contains](#)  
Files included in this example are as follows:
- [How to run the example](#)  
This mx-mt translation will use the sample files to demonstrate the generation of SWIFT MT292 message output from a CBPR+ camt.058.001.08 XML message.

## What the example contains

Files included in this example are as follows:

- cbpr\_translation.zip

- Files:
  - env\_camt\_058\_bad.xml
  - env\_camt\_058\_valid.xml
  - trx\_config.xml
- Schemas:
  - head.001.001.02\_camt\_058.xsd
  - camt.058.001.08.xsd
  - trx\_config.xsd
  - infoSet.mtt
  - swift\_iso7775\_ccyy.mtt
  - swiftRoute\_funds.mtt
- Maps:
  - cbpr2610\_camt058\_framework
  - cbpr2611\_camt058\_translate
  - cbpr2612\_camt058\_mt292
  - cbpr2501\_mxmt\_setvarlog
- Flows:
  - cbpr\_camt058\_mt292
- cbprJnodesConfigIBM.tar.gz

## How to run the example

This mx-*mt* translation will use the sample files to demonstrate the generation of SWIFT MT292 message output from a CBPR+ camt.058.001.08 XML message.

The cbprJnodesConfigIBM.tar.gz file is available either in IBM\_financialpaymentsplus\_vn.n.n.n.zip or in UIProjectImports directory.

Extract from cbprJnodesConfigIBM.tar.gz file.

- jnodes0.jar

The following jars needs to be copied from <TX\_install\_dir>/jars:

- jackson-core-n.n.n.jar
- jackson-annotations-n.n.n.jar
- jackson-databind-n.n.n.jar

Or visit <https://repo1.maven.org/maven2/com/fasterxml/jackson/core/> for download.

Note: Recommend using the latest version.

For the non Docker environments,

- Copy jars to <TX\_install\_dir>/extjar.

For TX V11.0.1 and up, native based Design Server installation,

Copy the following into the directory defined in config.yaml server.persistence.libs, by default, this is set to /opt/txlibs:

- jvcwrap.jar
- jvalccyy.jar

Restart the running application ./ITX stop and then ./ITX start.

For the Docker environments,

- docker cp jnodes0.jar <brand>-server:/opt/<company>/<brand>/libs/.
- Restart the design server, i.e., **docker restart** <brand>-server.

Note: For RHEL, Docker is not supported, Podman can be used as a substitute since it provides a command line interface similar to Docker. Below is an example of using podman:

- podman cp jnodes0.jar <brand>-server:/opt/<company>/<brand>/libs/.
- Restart the design server, i.e., **podman restart** <brand>-server.

This example may generate failure in the audit.log.json log file to report the translation failure due to the pre-conversion checks:

**NumberOfOccurrences <OrgnlItm><OrgnlNtfctnRef>** should not be greater than 1.

1. Import the cbpr\_translation.zip project into the Design Server.
2. Open the cbpr\_translation project in Design Server and view the flow cbpr\_camt058\_mt292.
  - a. The Flow Description points to all the maps to be executed during the flow process, which will be called from within some of the map nodes in the flow. The following is a list of maps invoked using RUN built-in function during the flow process:
    - @packagemap=cbpr\_plus/translation/mx-*mt*/maps/cbpr\_t9n\_mx\_camt058\_translate/cbpr2611\_camt058\_translate
    - @packagemap=cbpr\_plus/translation/mx-*mt*/maps/cbpr\_t9n\_mx\_camt058\_mt292/cbpr2612\_camt058\_mt292
    - @packagemap=cbpr\_plus/translation/mx-*mt*/maps/cbpr\_t9n\_mx\_setvarlog/cbpr2501\_mxmt\_setvarlog
  - b. It utilizes the following nodes:
    - i. Source Node
      - camt\_058
 

This node identifies the input data to be translated in the flow. It uses the INPUT\_FILE variable to set the location of the data.
    - ii. Map Node
      - mx\_translate
 

Runs framework map cbpr2610\_camt058\_framework, checks pre-translation conditions and translates the input file (mx or xml) into required output (mt).

- iii. Target Node
  - mt292
 

This node contains the resulting translation in MT format and creates the output as defined by the variable OUTPUT\_RESULT\_MT.
- iv. Log Node
  - audit\_log
 

This node creates a log file specified by the flow variable AUDIT\_LOG.
- c. It utilizes the following flow variables:
  - INPUT\_FILE
 

Default value for the flow variable INPUT\_FILE is ..//cbpr\_plus/translation/mx-mt/data/env\_camt\_058\_valid.xml. This is the data file to be used for translation and can be customized. It is used in the Source node camt\_058
  - OUTPUT\_RESULT\_MT
 

Default value for the flow variable OUTPUT\_RESULT\_MT is mt292.out. This is the location of the output file in mt format. It is used in the Target node mt292. It can be customized.
  - AUDIT\_LOG
 

Default value for the flow variable AUDIT\_LOG is audit.log.json. This is the location of the audit log. It is used in the Log node audit\_log. It can be customized.
  - CONFIG\_FILE
 

Default value for the flow variable CONFIG\_FILE is ..//cbpr\_plus/translation/mx-mt/data/trx\_config.xml. This is the location of the file containing the configuration settings. It can be customized.

3. Open the flow cbpr\_camt058\_mt292 in the Design Server. It utilizes one source node, one map node, one log node, and one target node.
  4. In Design Server, create a package that contains the input files and one flow. The maps will automatically be included during deployment of the package onto the runtime server.
- [Run the example using three different methods](#)  
You can run the example using three different methods:

## Run the example using three different methods

You can run the example using three different methods:

1. Running the example from the Design Server user interface.
  2. Deploying the example to tx-rest and running it using the Swagger interface.
  3. Deploying locally or to ftp server and using the flow command server process (flowcmdserver).
- [Run the example from the Design Server user interface](#)  
Use the following steps to run the example from the Design Server user interface.
  - [Run the example using the Swagger interface](#)  
To run the example, deploy to tx-rest and run it using the Swagger interface.
  - [Run the example using the flow command server process \(flowcmdserver\)](#)  
To run the example, deploy locally or to ftp server and use the flow command server process (flowcmdserver).

## Run the example from the Design Server user interface

Use the following steps to run the example from the Design Server user interface.

1. In the Design Server, open cbpr\_camt058\_mt292 flow and click on Run button.
2. Set toggle switch Run On Design Server and select flow input file env\_camt\_058\_valid.xml.  
Note: If a default path is assigned to the variable INPUT\_FILE, not selecting an input file will use the value defined for the variable.  
Flow will run and report with the green check box. The report lists execution of all nodes.
3. Right click and select View Data on each link to examine the data.
4. Right click on the node and select View Log to see the logs.

## Run the example using the Swagger interface

To run the example, deploy to tx-rest and run it using the Swagger interface.

1. Create a Web type server definition where the runtime tx-rest is installed, if you do not have a server definition to deploy in Design Server.
2. If not running, start tx-rest on the server: <DTX\_HOME>/restapi/tomcat/dtxtomcat start tx-rest.
3. Deploy the created package to the server definition in Design Server.
4. Bring up the tx-rest Swagger UI: <DTX\_HOME>/restapi/tomcat/dtxtomcat showdocs tx-rest.
5. In the Swagger Explore window, enter /tx-rest/v2/docs and click on the Explore button to view the deployed package.  
You can see the cbpr\_camt058\_mt292 section having two actions: A PUT /v2/run/cbpr\_camt058\_mt292 and a POST /v2/run/cbpr\_camt058\_mt292.
6. In the PUT action:
  - a. Expand the PUT action and select the Try it out button.
  - b. Leave the input and output sections empty.
  - c. Under flow\_vars section, delete the entire content and replace with the commands to run the flow, for example:

```
{
 "INPUT_FILE": "C:/tests_dir/data/env_camt_058_valid.xml",
```

```

"OUTPUT_RESULT_MT": "C:/tests_dir/data/mt292.txt",
"AUDIT_LOG": "C:/tests_dir/data/audit_log.json"
}

```

- d. Click Execute blue bar for running the flow. When flow completes execution, 200 response code is shown in the Server Response section. You can see information about the process flow in the Response body.
7. Refresh the browser to delete the deployed package and get the Swagger Explore back to /tx-rest/openapi.json.  
There will be a DELETE /v2/packages/{name} action.
8. Expand the DELETE /v2/packages/{name} action and select the Try it out.
9. In the Name field, enter the name you gave it to the created package.
10. Select the true option from the stop query drop down, select the blue Execute bar.  
You can see a response of 204 with a timestamp, if it is successful.

## Run the example using the flow command server process (flowcmdserver)

To run the example, deploy locally or to ftp server and use the flow command server process (flowcmdserver).

1. Create a server definition where the runtime is installed, if you do not have a server local definition or ftp execution definition to deploy to, in Design Server.
2. In Design Server, deploy the created package to the server definition. For example, deploy to c:\ftpserver\deployment directory.
3. Execute the flow using the flow command server:

```

flowcmdserver.exe --dir c:\ftpserver\deployment\flows --flow cbpr_camt058_mt292.json
--var "INPUT_FILE=C:\ftpserver\deployment\cbpr_plus\translation\mx-mt\data\env_camt_058_valid.xml"
--var "OUTPUT_RESULT_MT=C:\ftpserver\deployment\cbpr_plus\translation\mx-mt\data\mt292.out"
--var "AUDIT_LOG=C:\ftpserver\deployment\cbpr_plus\translation\mx-mt\data\env_camt_058_valid_log.json"
--audit "C:\ftpserver\deployment\cbpr_plus\translation\mx-mt\data\env_camt_058_valid_adt.json" -ad

```

4. The flow will report status similar to:

```

***Starting flow command server

Flow UUID: 85eabe2c-2302-4543-b347-9e18d65c6816
The flow audit file is: "C:\\ftpserver\\deployment\\cbpr_plus\\translation\\mx-
mt\\data\\env_camt_058_valid_adt.json"
Flow completed successfully
Elapsed time: 3048ms

```

5. Examine the flow output result file mt292.out and the audit file env\_camt\_058\_valid\_adt.json.

## CBPR+ MX to MT (camt.107.001.01 to MT110) Translation Example

This example demonstrates the CBPR MX to MT (camt.107.001.01 to MT110) message translation using flow server.

- [What the example contains](#)  
Files included in this example are as follows:
- [How to run the example](#)  
This mx-mt translation will use the sample files to demonstrate the generation of SWIFT MT110 message output from a CBPR+ camt.107.001.01 XML message.

## What the example contains

Files included in this example are as follows:

- cbpr\_translation.zip
  - Files:
    - env\_camt\_107\_valid.xml
    - trx\_config.xml
  - Schemas:
    - head.001.001.02\_camt\_107.xsd
    - camt.107.001.01.xsd
    - trx\_config.xsd
    - infoset.mtt
    - swift\_iso7775\_ccyy.mtt
    - swiftroute\_funds.mtt
  - Maps:
    - cbpr2620\_camt107\_framework
    - cbpr2621\_camt107\_translate
    - cbpr2622\_camt107\_mt110
    - cbpr2501\_mxmt\_setvarlog
  - Flows:
    - cbpr\_camt107\_mt110
- cbprJnodesConfigIBM.tar.gz

## How to run the example

This mx-*mt* translation will use the sample files to demonstrate the generation of SWIFT MT110 message output from a CBPR+ camt.107.001.01 XML message.

The cbprJnodesConfigIBM.tar.gz file is available either in IBM\_financialpaymentsplus\_vn.n.n.n.zip or in UIProjectImports directory.

Extract from cbprJnodesConfigIBM.tar.gz file.

- jnodes0.jar

The following jars needs to be copied from <TX\_install\_dir>/jars:

- jackson-core-n.n.n.jar
- jackson-annotations-n.n.n.jar
- jackson-databind-n.n.n.jar

Or visit <https://repo1.maven.org/maven2/com/fasterxml/jackson/core/> for download.

Note: Recommend using the latest version.

For the non Docker environments,

- Copy jars to <TX\_install\_dir>/extjar.

For TX V11.0.1 and up, native based Design Server installation,

Copy the following into the directory defined in config.yaml server.persistence.libs, by default, this is set to /opt/tlxlibs:

- jvcwrap.jar
- jvalccyy.jar

Restart the running application ./ITX stop and then ./ITX start.

For the Docker environments,

- docker cp jnodes0.jar <brand>-server:/opt/<company>/<brand>/libs/.
- Restart the design server, i.e., **docker restart <brand>-server**.

Note: For RHEL, Docker is not supported, Podman can be used as a substitute since it provides a command line interface similar to Docker. Below is an example of using podman:

- podman cp jnodes0.jar <brand>-server:/opt/<company>/<brand>/libs/.
- Restart the design server, i.e., **podman restart <brand>-server**.

1. Import the cbpr\_translation.zip project into the Design Server.

2. Open the cbpr\_translation project in Design Server and view the flow cbpr\_camt107\_mt110.

a. The Flow Description points to all the maps to be executed during the flow process, which will be called from within some of the map nodes in the flow. The following is a list of maps invoked using RUN built-in function during the flow process:

- @packagemap=cbpr\_plus/translation/mx-*mt*/maps/cbpr\_t9n\_mx\_camt107\_translate/cbpr2621\_camt107\_translate
- @packagemap=cbpr\_plus/translation/mx-*mt*/maps/cbpr\_t9n\_mx\_camt107\_mt110/cbpr2622\_camt107\_mt110
- @packagemap=cbpr\_plus/translation/mx-*mt*/maps/cbpr\_t9n\_mx\_setvarlog/cbpr2501\_mxmt\_setvarlog

b. It utilizes the following nodes:

i. Source Node

- camt\_107  
This node identifies the input data to be translated in the flow. It uses the INPUT\_FILE variable to set the location of the data.

ii. Map Node

- mx\_translate  
Runs framework map cbpr2620\_camt107\_framework, checks pre-translation conditions and translates the input file (mx or xml) into required output (mt).

iii. Target Node

- mt110  
This node contains the resulting translation in MT format and creates the output as defined by the variable OUTPUT\_RESULT\_MT.

iv. Log Node

- audit\_log  
This node creates a log file specified by the flow variable AUDIT\_LOG.

c. It utilizes the following flow variables:

- INPUT\_FILE

Default value for the flow variable INPUT\_FILE is ..//cbpr\_plus/translation/mx-*mt*/data/env\_camt\_107\_valid.xml. This is the data file to be used for translation and can be customized. It is used in the Source node camt\_107

- OUTPUT\_RESULT\_MT

Default value for the flow variable OUTPUT\_RESULT\_MT is mt110.out. This is the location of the output file in mt format. It is used in the Target node mt110. It can be customized.

- AUDIT\_LOG

Default value for the flow variable AUDIT\_LOG is audit.log.json. This is the location of the audit log. It is used in the Log node audit\_log. It can be customized.

- CONFIG\_FILE

Default value for the flow variable CONFIG\_FILE is ..//cbpr\_plus/translation/mx-*mt*/data/trx\_config.xml. This is the location of the file containing the configuration settings. It can be customized.

3. Open the flow cbpr\_camt107\_mt110 in the Design Server. It utilizes one source node, one map node, one log node, and one target node.

4. In Design Server, create a package that contains the input files and one flow. The maps will automatically be included during deployment of the package onto the runtime server.

- [Run the example using three different methods](#)

You can run the example using three different methods:

## Run the example using three different methods

You can run the example using three different methods:

1. Running the example from the Design Server user interface.
2. Deploying the example to tx-rest and running it using the Swagger interface.
3. Deploying locally or to ftp server and using the flow command server process (flowcmdserver).

- [Run the example from the Design Server user interface](#)

Use the following steps to run the example from the Design Server user interface.

- [Run the example using the Swagger interface](#)

To run the example, deploy to tx-rest and run it using the Swagger interface.

- [Run the example using the flow command server process \(flowcmdserver\)](#)

To run the example, deploy locally or to ftp server and use the flow command server process (flowcmdserver).

## Run the example from the Design Server user interface

Use the following steps to run the example from the Design Server user interface.

1. In the Design Server, open cbpr\_camt107\_mt110 flow and click on Run button.
2. Set toggle switch Run On Design Server and select flow input file env\_camt\_107\_valid.xml.  
Note: If a default path is assigned to the variable INPUT\_FILE, not selecting an input file will use the value defined for the variable.  
Flow will run and report with the green check box. The report lists execution of all nodes.
3. Right click and select View Data on each link to examine the data.
4. Right click on the node and select View Log to see the logs.

## Run the example using the Swagger interface

To run the example, deploy to tx-rest and run it using the Swagger interface.

1. Create a Web type server definition where the runtime tx-rest is installed, if you do not have a server definition to deploy in Design Server.
2. If not running, start tx-rest on the server: <DTX\_HOME>/restapi/tomcat/dtxtomcat start tx-rest.
3. Deploy the created package to the server definition in Design Server.
4. Bring up the tx-rest Swagger UI: <DTX\_HOME>/restapi/tomcat/dtxtomcat showdocs tx-rest.
5. In the Swagger Explore window, enter /tx-rest/v2/docs and click on the Explore button to view the deployed package.  
You can see the cbpr\_camt107\_mt110 section having two actions: A PUT /v2/run/cbpr\_camt107\_mt110 and a POST /v2/run/cbpr\_camt107\_mt110.
6. In the PUT action:
  - a. Expand the PUT action and select the Try it out button.
  - b. Leave the input and output sections empty.
  - c. Under flow\_vars section, delete the entire content and replace with the commands to run the flow, for example:

```
{
 "INPUT_FILE": "C:/tests_dir/data/env_camt_107_valid.xml",
 "OUTPUT_RESULT_MT": "C:/tests_dir/data/mt110.txt",
 "AUDIT_LOG": "C:/tests_dir/data/audit_log.json"
}
```

- d. Click Execute blue bar for running the flow. When flow completes execution, 200 response code is shown in the Server Response section. You can see information about the process flow in the Response body.
7. Refresh the browser to delete the deployed package and get the Swagger Explore back to /tx-rest/openapi.json.  
There will be a DELETE /v2/packages/{name} action.
8. Expand the DELETE /v2/packages/{name} action and select the Try it out.
9. In the Name field, enter the name you gave it to the created package.
10. Select the true option from the stop query drop down, select the blue Execute bar.  
You can see a response of 204 with a timestamp, if it is successful.

## Run the example using the flow command server process (flowcmdserver)

To run the example, deploy locally or to ftp server and use the flow command server process (flowcmdserver).

1. Create a server definition where the runtime is installed, if you do not have a server local definition or ftp execution definition to deploy to, in Design Server.
2. In Design Server, deploy the created package to the server definition. For example, deploy to c:\ftpserver\deployment directory.
3. Execute the flow using the flow command server:

```
flowcmdserver.exe --dir c:\ftpserver\deployment\flows --flow cbpr_camt107_mt110.json
--var "INPUT_FILE=C:\ftpserver\deployment\cbpr_plus\translation\mx-mt\data\env_camt_107_valid.xml"
--var "OUTPUT_RESULT_MT=C:\ftpserver\deployment\cbpr_plus\translation\mx-mt\data\mt110.out"
--var "AUDIT_LOG=C:\ftpserver\deployment\cbpr_plus\translation\mx-mt\data\env_camt_107_valid_log.json"
--audit "C:\ftpserver\deployment\cbpr_plus\translation\mx-mt\data\env_camt_107_valid_adt.json" -ad
```

4. The flow will report status similar to:

```
***Starting flow command server
Flow UUID: 85eabe2c-2302-4543-b347-9e18d65c6816
The flow audit file is: "C:\\ftpserver\\deployment\\cbpr_plus\\translation\\mx-
mt\\data\\env_camt_107_valid_adt.json"
Flow completed successfully
Elapsed time: 3048ms
```

5. Examine the flow output result file `mt110.out` and the audit file `env_camt_107_valid_adt.json`.

## CBPR+ MX to MT (camt.108.001.01 to MT111) Translation Example

This example demonstrates the CBPR MX to MT (camt.108.001.01 to MT111) message translation using flow server.

- **What the example contains**

Files included in this example are as follows:

- `cbpr_translation.zip`
  - Files:
    - `env_camt_108_valid.xml`
    - `trx_config.xml`
  - Schemas:
    - `head.001.001.02_camt_108.xsd`
    - `camt.108.001.01.xsd`
    - `trx_config.xsd`
    - `infoset.mtt`
    - `swift_iso7775_ccyy.mtt`
    - `swiftroute_funds.mtt`
  - Maps:
    - `cbpr2630_camt108_framework`
    - `cbpr2631_camt108_translate`
    - `cbpr2632_camt108_mt111`
    - `cbpr2501_mxmt_setvarlog`
  - Flows:
    - `cbpr_camt108_mt111`
- `cbprJnodesConfigIBM.tar.gz`

## What the example contains

Files included in this example are as follows:

- `cbpr_translation.zip`
  - Files:
    - `env_camt_108_valid.xml`
    - `trx_config.xml`
  - Schemas:
    - `head.001.001.02_camt_108.xsd`
    - `camt.108.001.01.xsd`
    - `trx_config.xsd`
    - `infoset.mtt`
    - `swift_iso7775_ccyy.mtt`
    - `swiftroute_funds.mtt`
  - Maps:
    - `cbpr2630_camt108_framework`
    - `cbpr2631_camt108_translate`
    - `cbpr2632_camt108_mt111`
    - `cbpr2501_mxmt_setvarlog`
  - Flows:
    - `cbpr_camt108_mt111`
- `cbprJnodesConfigIBM.tar.gz`

## How to run the example

This mx-mt translation will use the sample files to demonstrate the generation of SWIFT MT111 message output from a CBPR camt.108.001.01 XML message.

The `cbprJnodesConfigIBM.tar.gz` file is available either in `IBM_financialpaymentsplus_vn.n.n.n.zip` or in `UIProjectImports` directory.

Extract from `cbprJnodesConfigIBM.tar.gz` file.

- `jnodes0.jar`

The following jars needs to be copied from `<TX_install_dir>/jars`:

- `jackson-core-n.n.n.jar`
- `jackson-annotations-n.n.n.jar`
- `jackson-databind-n.n.n.jar`

Or visit <https://repo1.maven.org/maven2/com/fasterxml/jackson/core/> for download.

Note: Recommend using the latest version.

For the non Docker environments,

- Copy jars to `<TX_install_dir>/extjar`.

For TX V11.0.1 and up, native based Design Server installation,

Copy the following into the directory defined in `config.yaml` `server.persistence.libs`, by default, this is set to `/opt/txlabs`:

- `jvcwrap.jar`
- `jvalccyy.jar`

Restart the running application `./ITX stop` and then `./ITX start`.

For the Docker environments,

- docker cp jnodes0.jar <brand>-server:/opt/<company>/<brand>/libs/.
- Restart the design server, i.e., **docker restart** <brand>-server.

Note: For RHEL, Docker is not supported, Podman can be used as a substitute since it provides a command line interface similar to Docker. Below is an example of using podman:

- podman cp jnodes0.jar <brand>-server:/opt/<company>/<brand>/libs/.
- Restart the design server, i.e., **podman restart** <brand>-server.

This example may generate failure in the audit.log.json log file to report the translation failure, if the input message is not camt.108.001.01.

1. Import the cbpr\_translation.zip project into the Design Server.
  2. Open the cbpr\_translation project in Design Server and view the flow cbpr\_camt108\_mt111.
    - a. The Flow Description points to all the maps to be executed during the flow process, which will be called from within some of the map nodes in the flow. The following is a list of maps invoked using RUN built-in function during the flow process:
      - @packagemap=cbpr\_plus/translation/mx-ml/maps/cbpr\_t9n\_mx\_camt108\_translate/cbpr2631\_camt108\_translate
      - @packagemap=cbpr\_plus/translation/mx-ml/maps/cbpr\_t9n\_mx\_camt108\_mt111/cbpr2632\_camt108\_mt111
      - @packagemap=cbpr\_plus/translation/mx-ml/maps/cbpr\_t9n\_mx\_setvarlog/cbpr2501\_mxmt\_setvarlog
    - b. It utilizes the following nodes:
      - i. Source Node
        - camt\_108
 

This node identifies the input data to be translated in the flow. It uses the INPUT\_FILE variable to set the location of the data.
      - ii. Map Node
        - mx\_translate
 

Runs framework map cbpr2630\_camt108\_framework, checks pre-translation conditions and translates the input file (mx or xml) into required output (mt).
      - iii. Target Node
        - mt111
 

This node contains the resulting translation in MT format and creates the output as defined by the variable OUTPUT\_RESULT\_MT.
      - iv. Log Node
        - audit\_log
 

This node creates a log file specified by the flow variable AUDIT\_LOG.
    - c. It utilizes the following flow variables:
      - INPUT\_FILE
 

Default value for the flow variable INPUT\_FILE is ..//cbpr\_plus/translation/mx-ml/data/env\_camt\_108\_valid.xml. This is the data file to be used for translation and can be customized. It is used in the Source node camt\_108
      - OUTPUT\_RESULT\_MT
 

Default value for the flow variable OUTPUT\_RESULT\_MT is mt111.out. This is the location of the output file in mt format. It is used in the Target node mt111. It can be customized.
      - AUDIT\_LOG
 

Default value for the flow variable AUDIT\_LOG is audit.log.json. This is the location of the audit log. It is used in the Log node audit\_log. It can be customized.
      - CONFIG\_FILE
 

Default value for the flow variable CONFIG\_FILE is ..//cbpr\_plus/translation/mx-ml/data/trx\_config.xml. This is the location of the file containing the configuration settings. It can be customized.
  3. Open the flow cbpr\_camt108\_mt111 in the Design Server. It utilizes one source node, one map node, one log node, and one target node.
  4. In Design Server, create a package that contains the input files and one flow. The maps will automatically be included during deployment of the package onto the runtime server.
- [\*\*Run the example using three different methods\*\*](#)  
You can run the example using three different methods:

## Run the example using three different methods

You can run the example using three different methods:

1. Running the example from the Design Server user interface.
  2. Deploying the example to tx-rest and running it using the Swagger interface.
  3. Deploying locally or to ftp server and using the flow command server process (flowcmdserver).
- [\*\*Run the example from the Design Server user interface\*\*](#)  
Use the following steps to run the example from the Design Server user interface.
  - [\*\*Run the example using the Swagger interface\*\*](#)  
To run the example, deploy to tx-rest and run it using the Swagger interface.
  - [\*\*Run the example using the flow command server process \(flowcmdserver\)\*\*](#)  
To run the example, deploy locally or to ftp server and use the flow command server process (flowcmdserver).

## Run the example from the Design Server user interface

Use the following steps to run the example from the Design Server user interface.

1. In the Design Server, open cbpr\_camt108\_mt111 flow and click on Run button.
2. Set toggle switch Run On Design Server and select flow input file env\_camt\_108\_valid.xml.  
Note: If a default path is assigned to the variable INPUT\_FILE, not selecting an input file will use the value defined for the variable.  
Flow will run and report with the green check box. The report lists execution of all nodes.
3. Right click and select View Data on each link to examine the data.
4. Right click on the node and select View Log to see the logs.

---

## Run the example using the Swagger interface

To run the example, deploy to tx-rest and run it using the Swagger interface.

1. Create a Web type server definition where the runtime tx-rest is installed, if you do not have a server definition to deploy in Design Server.
2. If not running, start tx-rest on the server: <DTX\_HOME>/restapi/tomcat/dtxtomcat start tx-rest.
3. Deploy the created package to the server definition in Design Server.
4. Bring up the tx-rest Swagger UI: <DTX\_HOME>/restapi/tomcat/dtxtomcat showdocs tx-rest.
5. In the Swagger Explore window, enter /tx-rest/v2/docs and click on the Explore button to view the deployed package.

You can see the cbpr\_camt108\_mt111 section having two actions: A PUT /v2/run/cbpr\_camt108\_mt111 and a POST /v2/run/cbpr\_camt108\_mt111.

6. In the PUT action:
  - a. Expand the PUT action and select the Try it out button.
  - b. Leave the input and output sections empty.
  - c. Under flow\_vars section, delete the entire content and replace with the commands to run the flow, for example:

```
{
 "INPUT_FILE": "C:/tests_dir/data/env_camt_108_valid.xml",
 "OUTPUT_RESULT_MT": "C:/tests_dir/data/mt111.txt",
 "AUDIT_LOG": "C:/tests_dir/data/audit_log.json"
}
```

- d. Click Execute blue bar for running the flow. When flow completes execution, 200 response code is shown in the Server Response section. You can see information about the process flow in the Response body.
7. Refresh the browser to delete the deployed package and get the Swagger Explore back to /tx-rest/openapi.json.  
There will be a DELETE /v2/packages/{name} action.
8. Expand the DELETE /v2/packages/{name} action and select the Try it out.
9. In the Name field, enter the name you gave it to the created package.
10. Select the true option from the stop query drop down, select the blue Execute bar.

You can see a response of 204 with a timestamp, if it is successful.

---

## Run the example using the flow command server process (flowcmdserver)

To run the example, deploy locally or to ftp server and use the flow command server process (flowcmdserver).

1. Create a server definition where the runtime is installed, if you do not have a server local definition or ftp execution definition to deploy to, in Design Server.
2. In Design Server, deploy the created package to the server definition. For example, deploy to c:\ftpserver\deployment directory.
3. Execute the flow using the flow command server:

```
flowcmdserver.exe --dir c:\ftpserver\deployment\flows --flow cbpr_camt108_mt111.json
--var "INPUT_FILE=C:\ftpserver\deployment\cbpr_plus\translation\mx-mt\data\env_camt_108_valid.xml"
--var "OUTPUT_RESULT_MT=C:\ftpserver\deployment\cbpr_plus\translation\mx-mt\data\env_camt_108_111.out"
--var "AUDIT_LOG=C:\ftpserver\deployment\cbpr_plus\translation\mx-mt\data\env_camt_108_111_log.json"
--audit "C:\ftpserver\deployment\cbpr_plus\translation\mx-mt\data\env_camt_108_111_adt.json" -ad
```

4. The flow will report status similar to:

```
***Starting flow command server

Flow UUID: 85eabe2c-2302-4543-b347-9e18d65c6816
The flow audit file is: "C:\\\\ftpserver\\\\deployment\\\\cbpr_plus\\\\translation\\\\mx-
mt\\\\data\\\\env_camt_108_111_adt.json"
Flow completed successfully
Elapsed time: 3048ms
```

5. Examine the flow output result file env\_camt\_108\_111.out and the audit file env\_camt\_108\_111\_adt.json.

---

## CBPR+ MX to MT (camt.109.001.01 to MT112) Translation Example

This example demonstrates the CBPR MX to MT (camt.109.001.01 to MT112) message translation using flow server.

- **What the example contains**

Files included in this example are as follows:

- **How to run the example**

This mx-mt translation will use the sample files to demonstrate the generation of SWIFT MT112 message output from a CBPR+ camt.109.001.01 XML message.

---

## What the example contains

Files included in this example are as follows:

- cbpr\_translation.zip
  - Files:
    - env\_camt\_109\_valid.xml
    - trx\_config.xml
  - Schemas:
    - head.001.001.02\_camt\_109.xsd
    - camt.109.001.01.xsd
    - trx\_config.xsd
    - infoset.mtt
    - swift\_iso7775\_ccyy.mtt
    - swiftroute\_funds.mtt
  - Maps:
    - cbpr2640\_camt109\_framework
    - cbpr2641\_camt109\_translate
    - cbpr2642\_camt109\_mt112
    - cbpr2501\_mxmt\_setvarlog
  - Flows:
    - cbpr\_camt109\_mt112
- cbprJnodesConfigIBM.tar.gz

## How to run the example

This mx-*mt* translation will use the sample files to demonstrate the generation of SWIFT MT112 message output from a CBPR+ camt.109.001.01 XML message.

The cbprJnodesConfigIBM.tar.gz file is available either in IBM\_financialpaymentsplus\_vn.n.n.n.zip or in UIProjectImports directory.

Extract from cbprJnodesConfigIBM.tar.gz file.

- jnodes0.jar

The following jars needs to be copied from <TX\_install\_dir>/jars:

- jackson-core-n.n.n.jar
- jackson-annotations-n.n.n.jar
- jackson-databind-n.n.n.jar

Or visit <https://repo1.maven.org/maven2/com/fasterxml/jackson/core/> for download.

Note: Recommend using the latest version.

For the non Docker environments,

- Copy jars to <TX\_install\_dir>/extjar.

For TX V11.0.1 and up, native based Design Server installation,

Copy the following into the directory defined in config.yaml server.persistence.libs, by default, this is set to /opt/tlxlibs:

- jvcwrap.jar
- jvalccyy.jar

Restart the running application ./ITX stop and then ./ITX start.

For the Docker environments,

- docker cp jnodes0.jar <brand>-server:/opt/<company>/<brand>/libs/.
- Restart the design server, i.e., **docker restart** <brand>-server.

Note: For RHEL, Docker is not supported, Podman can be used as a substitute since it provides a command line interface similar to Docker. Below is an example of using podman:

- podman cp jnodes0.jar <brand>-server:/opt/<company>/<brand>/libs/.
- Restart the design server, i.e., **podman restart** <brand>-server.

This example may generate failure in the audit.log.json log file to report the translation failure, if input message is not camt.109.001.01

1. Import the cbpr\_translation.zip project into the Design Server.
2. Open the cbpr\_translation project in Design Server and view the flow cbpr\_camt109\_mt112.
  - a. The Flow Description points to all the maps to be executed during the flow process, which will be called from within some of the map nodes in the flow. The following is a list of maps invoked using RUN built-in function during the flow process:
    - @packagemap=cbpr\_plus/translation/mx-*mt*/maps/cbpr\_t9n\_mx\_camt109\_translate/cbpr2641\_camt109\_translate
    - @packagemap=cbpr\_plus/translation/mx-*mt*/maps/cbpr\_t9n\_mx\_camt109\_mt112/cbpr2642\_camt109\_mt112
    - @packagemap=cbpr\_plus/translation/mx-*mt*/maps/cbpr\_t9n\_mx\_setvarlog/cbpr2501\_mxmt\_setvarlog
  - b. It utilizes the following nodes:
    - i. Source Node
      - camt\_109

This node identifies the input data to be translated in the flow. It uses the INPUT\_FILE variable to set the location of the data.
    - ii. Map Node
      - mx\_translate

Runs framework map cbpr2640\_camt109\_framework, checks pre-translation conditions and translates the input file (mx or xml) into required output (mt).

iii. Target Node

- mt112  
This node contains the resulting translation in MT format and creates the output as defined by the variable OUTPUT\_RESULT\_MT.

iv. Log Node

- audit\_log  
This node creates a log file specified by the flow variable AUDIT\_LOG.

c. It utilizes the following flow variables:

- INPUT\_FILE  
Default value for the flow variable INPUT\_FILE is ..../cbpr\_plus/translation/mx-mt/data/env\_camt\_109\_valid.xml. This is the data file to be used for translation and can be customized. It is used in the Source node camt\_109.
- OUTPUT\_RESULT\_MT  
Default value for the flow variable OUTPUT\_RESULT\_MT is mt112.out. This is the location of the output file in mt format. It is used in the Target node mt112. It can be customized.
- AUDIT\_LOG  
Default value for the flow variable AUDIT\_LOG is audit.log.json. This is the location of the audit log. It is used in the Log node audit\_log. It can be customized.
- CONFIG\_FILE  
Default value for the flow variable CONFIG\_FILE is ..../cbpr\_plus/translation/mx-mt/data/trx\_config.xml. This is the location of the file containing the configuration settings. It can be customized.

3. Open the flow cbpr\_camt109\_mt112 in the Design Server. It utilizes one source node, one map node, one log node, and one target node.

4. In Design Server, create a package that contains the input files and one flow. The maps will automatically be included during deployment of the package onto the runtime server.

- [\*\*Run the example using three different methods\*\*](#)

You can run the example using three different methods:

---

## Run the example using three different methods

You can run the example using three different methods:

1. Running the example from the Design Server user interface.
2. Deploying the example to tx-rest and running it using the Swagger interface.
3. Deploying locally or to ftp server and using the flow command server process (flowcmdserver).

- [\*\*Run the example from the Design Server user interface\*\*](#)

Use the following steps to run the example from the Design Server user interface.

- [\*\*Run the example using the Swagger interface\*\*](#)

To run the example, deploy to tx-rest and run it using the Swagger interface.

- [\*\*Run the example using the flow command server process \(flowcmdserver\)\*\*](#)

To run the example, deploy locally or to ftp server and use the flow command server process (flowcmdserver).

---

## Run the example from the Design Server user interface

Use the following steps to run the example from the Design Server user interface.

1. In the Design Server, open cbpr\_camt109\_mt112 flow and click on Run button.
2. Set toggle switch Run On Design Server and select flow input file env\_camt\_109\_valid.xml.  
Note: If a default path is assigned to the variable INPUT\_FILE, not selecting an input file will use the value defined for the variable.  
Flow will run and report with the green check box. The report lists execution of all nodes.
3. Right click and select View Data on each link to examine the data.
4. Right click on the node and select View Log to see the logs.

---

## Run the example using the Swagger interface

To run the example, deploy to tx-rest and run it using the Swagger interface.

1. Create a Web type server definition where the runtime tx-rest is installed, if you do not have a server definition to deploy in Design Server.
2. If not running, start tx-rest on the server: <DTX\_HOME>/restapi/tomcat/dtxtomcat start tx-rest.
3. Deploy the created package to the server definition in Design Server.
4. Bring up the tx-rest Swagger UI: <DTX\_HOME>/restapi/tomcat/dtxtomcat showdocs tx-rest.
5. In the Swagger Explore window, enter /tx-rest/v2/docs and click on the Explore button to view the deployed package.  
You can see the cbpr\_camt109\_mt112 section having two actions: A PUT /v2/run/cbpr\_camt109\_mt112 and a POST /v2/run/cbpr\_camt109\_mt112.
6. In the PUT action:
  - a. Expand the PUT action and select the Try it out button.
  - b. Leave the input and output sections empty.
  - c. Under flow\_vars section, delete the entire content and replace with the commands to run the flow, for example:

```
{
 "INPUT_FILE": "C:/tests_dir/data/env_camt_109_valid.xml",
 "OUTPUT_RESULT_MT": "C:/tests_dir/data/mt112.txt",
 "AUDIT_LOG": "C:/tests_dir/data/audit_log.json"
}
```

- d. Click Execute blue bar for running the flow. When flow completes execution, 200 response code is shown in the Server Response section. You can see information about the process flow in the Response body.
7. Refresh the browser to delete the deployed package and get the Swagger Explore back to /tx-rest/openapi.json. There will be a DELETE /v2/packages/{name} action.
8. Expand the DELETE /v2/packages/{name} action and select the Try it out.
9. In the Name field, enter the name you gave it to the created package.
10. Select the true option from the stop query drop down, select the blue Execute bar.  
You can see a response of 204 with a timestamp, if it is successful.

## Run the example using the flow command server process (flowcmdserver)

To run the example, deploy locally or to ftp server and use the flow command server process (flowcmdserver).

1. Create a server definition where the runtime is installed, if you do not have a server local definition or ftp execution definition to deploy to, in Design Server.
2. In Design Server, deploy the created package to the server definition. For example, deploy to c:\ftpserver\deployment directory.
3. Execute the flow using the flow command server:

```
flowcmdserver.exe --dir c:\ftpserver\deployment\flows --flow cbpr_camt109_mt112.json
--var "INPUT_FILE=C:\ftpserver\deployment\cbpr_plus\translation\mx-mt\data\env_camt_109_valid.xml"
--var "OUTPUT_RESULT_MT=C:\ftpserver\deployment\cbpr_plus\translation\mx-mt\data\env_camt_109_112.out"
--var "AUDIT_LOG=C:\ftpserver\deployment\cbpr_plus\translation\mx-mt\data\env_camt_109_112_log.json"
--audit "C:\ftpserver\deployment\cbpr_plus\translation\mx-mt\data\env_camt_109_112_adt.json" -ad
```

4. The flow will report status similar to:

```
***Starting flow command server

 Flow UUID: 85eabe2c-2302-4543-b347-9e18d65c6816
 The flow audit file is: "C:\\\\ftpserver\\\\deployment\\\\cbpr_plus\\\\translation\\\\mx-
mt\\\\data\\\\env_camt_109_112_adt.json"
 Flow completed successfully
 Elapsed time: 3048ms
```

5. Examine the flow output result file env\_camt\_109\_112.out and the audit file env\_camt\_109\_112\_adt.json.

## CBPR+ MX to MT (pacs.002.001.10 to MT199/MT299) Translation Example

This example demonstrates the CBPR+ MX to MT (pacs.002.001.10 to MT199/MT299) message translation using flow server.

- [What the example contains](#)  
Files included in this example are as follows:
- [How to run the example](#)  
This mx-mt translation will use the sample files to demonstrate the generation of SWIFT MT199 or MT299 message output from a CBPR pacs.002.001.10 XML message.

## What the example contains

Files included in this example are as follows:

- cbpr\_translator.zip
  - Files:
    - env\_pacs\_002\_199.xml
    - env\_pacs\_002\_299.xml
    - env\_pacs\_002\_bad.xml
    - trx\_config.xml
  - Schemas:
    - head.001.001.02\_pacs\_002.xsd
    - pacs.002.001.10.xsd
    - trx\_config.xsd
    - infoset.mtt
    - swift\_iso7775\_ccyy.mtt
    - swiftroute\_funds.mtt
  - Maps:
    - cbpr2500\_pacs002\_framework
    - cbpr2502\_pacs002\_translate
    - cbpr2503\_pacs\_002\_mtn99
    - cbpr2501\_mxmt\_setvarlog
  - Flow:
    - cbpr\_pacs002\_mtn99
- cbprJnodesConfigIBM.tar.gz

## How to run the example

This mx-*mt* translation will use the sample files to demonstrate the generation of SWIFT MT199 or MT299 message output from a CBPR pacs.002.001.10 XML message.

The cbprJnodesConfigIBM.tar.gz file is available either in IBM\_financialpaymentsplus\_vn.n.n.n.zip or in UIProjectImports directory.

Extract from cbprJnodesConfigIBM.tar.gz file.

- jnodes0.jar

The following jars needs to be copied from <TX\_install\_dir>/jars:

- jackson-core-n.n.n.jar
- jackson-annotations-n.n.n.jar
- jackson-databind-n.n.n.jar

Or visit <https://repo1.maven.org/maven2/com/fasterxml/jackson/core/> for download.

Note: Recommend using the latest version.

For the non Docker environments,

- Copy jars to <TX\_install\_dir>/extjar.

For TX V11.0.1 and up, native based Design Server installation,

Copy the following into the directory defined in config.yaml server.persistence.libs, by default, this is set to /opt/txlibs:

- jvcwrap.jar
- jvalccyy.jar

Restart the running application ./ITX stop and then ./ITX start.

For the Docker environments,

- docker cp jnodes0.jar <brand>-server:/opt/<company>/<brand>/libs/.
- Restart the design server, i.e., **docker restart** <brand>-server.

Note: For RHEL, Docker is not supported, Podman can be used as a substitute since it provides a command line interface similar to Docker. Below is an example of using podman:

- podman cp jnodes0.jar <brand>-server:/opt/<company>/<brand>/libs/.
- Restart the design server, i.e., **podman restart** <brand>-server.

This example generates MT199 or MT299 based on the <OrgnlMsgNmId> which could be pacs.008 or MT103 (will generate MT199) or pacs.009 or MT202 or MT205 (will generate MT299).

Also, may generate failure in the audit.log.json log file to report the translation failure due to the pre-conversion checks:

**Either <OrgnlMsgNmId> is not one of these ("pacs.008", "MT103", "pacs.009", "MT202", "MT205") or <TxSts> "transaction status" is not "RJCT".**

1. Import the cbpr\_translation.zip project into the Design Server.
2. Open the cbpr\_translation project in Design Server and view the flow cbpr\_pacs002\_mtn99.
  - a. The Flow Description points to all the maps to be executed during the flow process, which will be called from within some of the map nodes in the flow. The following is a list of maps invoked using RUN built-in function during the flow process:
    - @packagemap=cbpr\_plus/translation/mx-*mt*/maps/cbpr\_t9n\_mx\_pacs002\_translate/cbpr2502\_pacs002\_translate
    - @packagemap=cbpr\_plus/translation/mx-*mt*/maps/cbpr\_t9n\_mx\_pacs002\_mtn99/cbpr2503\_pacs\_002\_mtn99
    - @packagemap=cbpr\_plus/translation/mx-*mt*/maps/cbpr\_t9n\_mx\_setvarlog/cbpr2501\_mxmt\_setvarlog
  - b. It utilizes the following nodes:
    - i. Source Node
      - pacs\_002

This node identifies the input data to be translated in the flow. It uses the INPUT\_FILE variable to set the location of the data.
    - ii. Map Node
      - mx\_translate

Runs framework map cbpr2500\_pacs002\_framework, checks pre-translation conditions and translates the input file (mx or xml) into required output (mt).
    - iii. Target Node
      - mtn99

This node contains the resulting translation in MT format (mt199.out or mt299.out) and creates the output as defined by the variable OUTPUT\_RESULT\_MT.
    - iv. Log Node
      - audit\_log

This node creates a log file specified by the flow variable AUDIT\_LOG.
  - c. It utilizes the following flow variables:
    - INPUT\_FILE

Default value for the flow variable INPUT\_FILE is ..//cbpr\_plus/translation/mx-*mt*/data/env\_pacs\_002\_199.xml. This is the data file to be used for translation and can be customized. It is used in the Source node pacs\_002

    - OUTPUT\_RESULT\_MT

Default value for the flow variable OUTPUT\_RESULT\_MT is mtn99.out. This is the location of the output file in mt format. It is used in the Target nodes mt199 or mt299 . It can be customized.

- AUDIT\_LOG  
Default value for the flow variable AUDIT\_LOG is audit.log.json. This is the location of the audit log. It is used in the Log node audit\_log. It can be customized.
- CONFIG\_FILE  
Default value for the flow variable CONFIG\_FILE is ../cbpr\_plus/translation/mx-mlt/data/trx\_config.xml. This is the location of the file containing the configuration settings. It can be customized.

3. Open the flow cbpr\_pacs002\_mtn99 in the Design Server. It utilizes one source node, one map node, one log node, and one target node.
4. In Design Server, create a package that contains the input files and one flow. The maps will automatically be included during deployment of the package onto the runtime server.

- **[Run the example using three different methods](#)**

You can run the example using three different methods:

## Run the example using three different methods

You can run the example using three different methods:

1. Running the example from the Design Server user interface.
2. Deploying the example to tx-rest and running it using the Swagger interface.
3. Deploying locally or to ftp server and using the flow command server process (flowcmdserver).

- **[Run the example from the Design Server user interface](#)**

Use the following steps to run the example from the Design Server user interface:

- **[Run the example using the Swagger interface](#)**

To run the example, deploy to tx-rest and run it using the Swagger interface.

- **[Run the example using the flow command server process \(flowcmdserver\)](#)**

To run the example, deploy locally or to ftp server and use the flow command server process (flowcmdserver).

## Run the example from the Design Server user interface

Use the following steps to run the example from the Design Server user interface:

1. In the Design Server, open cbpr\_pacs002\_mtn99 flow and click on Run button.
2. Set toggle switch Run On Design Server and select flow input file env\_pacs\_002\_199.xml or env\_pacs\_002\_299.xml.  
Note: If a default path is assigned to the variable INPUT\_FILE, not selecting an input file will use the value defined for the variable.  
Flow will run and report with the green check box. The report lists execution of all nodes.
3. Right click and select View Data on each link to examine the data.
4. Right click on the node and select View Log to see the logs.

## Run the example using the Swagger interface

To run the example, deploy to tx-rest and run it using the Swagger interface.

1. Create a Web type server definition where the runtime tx-rest is installed, if you do not have a server definition to deploy in Design Server.
2. If not running, start tx-rest on the server: <DTX\_HOME>/restapi/tomcat/dtxtomcat start tx-rest.
3. Deploy the created package to the server definition in Design Server.
4. Bring up the tx-rest Swagger UI: <DTX\_HOME>/restapi/tomcat/dtxtomcat showdocs tx-rest.
5. In the Swagger Explore window, enter /tx-rest/v2/docs and click on the Explore button to view the deployed package.  
You can see the cbpr\_pacs002\_mtn99 section having two actions: A PUT /v2/run/cbpr\_pacs002\_mtn99 and a POST /v2/run/cbpr\_pacs002\_mtn99.
6. In the PUT action:
  - a. Expand the PUT action and select the Try it out button.
  - b. Leave the input and output sections empty.
  - c. Under flow\_vars section, delete the entire content and replace with the commands to run the flow, for example:

```
{
 "INPUT_FILE": "C:/tests_dir/data/env_pacs_002_199.xml",
 "OUTPUT_RESULT_MTL": "C:/tests_dir/data/mtl99.out",
 "AUDIT_LOG": "C:/tests_dir/data/audit_log.json"
}
```

d. Click Execute blue bar for running the flow. When flow completes execution, 200 response code is shown in the Server Response section. You can see information about the process flow in the Response body.

7. Refresh the browser to delete the deployed package and get the Swagger Explore back to /tx-rest/openapi.json.  
There will be a DELETE /v2/packages/{name} action.
8. Expand the DELETE /v2/packages/{name} action and select the Try it out.
9. In the Name field, enter the name you gave it to the created package.
10. Select the true option from the stop query drop down, select the blue Execute bar.  
You can see a response of 204 with a timestamp, if it is successful.

## Run the example using the flow command server process (flowcmdserver)

To run the example, deploy locally or to ftp server and use the flow command server process (flowcmdserver).

1. Create a server definition where the runtime is installed, if you do not have a server local definition or ftp execution definition to deploy to, in Design Server.
2. In Design Server, deploy the created package to the server definition. For example, deploy to c:\ftpserver\deployment directory.
3. Execute the flow using the flow command server:

```
flowcmdserver.exe --dir c:\ftpserver\deployment\flows --flow cbpr_pacs002_mt99.json
--var "INPUT_FILE=C:\ftpserver\deployment\cbpr_plus\translation\mx-mt\data\env_pacs_002_199.xml"
--var "OUTPUT_RESULT_MF=C:\ftpserver\deployment\cbpr_plus\translation\mx-mt\data\env_pacs_002_n99.out"
--var "AUDIT_LOG=C:\ftpserver\deployment\cbpr_plus\translation\mx-mt\data\env_pacs_002_n99_log.json"
--audit "C:\ftpserver\deployment\cbpr_plus\translation\mx-mt\data\env_pacs_002_n99_adt.json" -ad
```

4. The flow will report status similar to:

```
***Starting flow command server

Flow UUID: 85eabe2c-2302-4543-b347-9e18d65c6816
The flow audit file is: "C:\\ftpserver\\deployment\\cbpr_plus\\translation\\mx-
mt\\data\\env_pacs_002_n99_adt.json"
Flow completed successfully
Elapsed time: 3048ms
```

5. Examine the flow output result file env\_pacs\_002\_199.out and the audit file env\_pacs\_002\_n99\_adt.json.

## CBPR+ MX to MT (pacs.004.001.09 to MTnnnRETN) Translation Example

This example demonstrates the CBPR MX to MT (pacs.004.001.09 to MTnnnRETN) message translation using flow server.

- [What the example contains](#)  
Files included in this example are as follows:
- [How to run the example](#)  
This mx-mt translation will use the sample files to demonstrate the generation of SWIFT MTnnn RETN message output from a CBPR pacs.004.001.09 XML message.

## What the example contains

Files included in this example are as follows:

- cbpr\_translation.zip
  - Files:
    - env\_pacs\_004\_mt202rtn\_valid.xml
    - env\_pacs\_004\_mt205rtn\_valid.xml
    - env\_pacs\_004\_mt103rtn\_valid.xml
    - env\_pacs\_004\_mt202rtn\_bad.xml
    - env\_pacs\_004\_mt205rtn\_bad.xml
    - env\_pacs\_004\_mt103rtn\_bad.xml
    - trx\_config.xml
  - Schemas:
    - head.001.001.02\_pacs\_004.xsd
    - pacs.004.001.09.xsd
    - trx\_config.xsd
    - infoset.mtt
    - swift\_iso7775\_ccyy.mtt
    - swiftroute\_funds.mtt
  - Maps:
    - cbpr2850\_pacs004\_mtnnn\_RTN\_frw
    - cbpr2851\_pacs004RTN\_trx
    - cbpr2852\_pacs004\_mt202RTN
    - cbpr2853\_pacs004\_mt205RTN
    - cbpr2857\_pacs004\_mt103RTN
    - cbpr2501\_mxmt\_setvarlog
  - Flow:
    - cbpr\_pacs004\_mtnnnRTN
- cbprJnodesConfigIBM.tar.gz

## How to run the example

This mx-mt translation will use the sample files to demonstrate the generation of SWIFT MTnnn RETN message output from a CBPR pacs.004.001.09 XML message.

The cbprJnodesConfigIBM.tar.gz file is available either in IBM\_financialpaymentsplus\_vn.n.n.n.zip or in UIProjectImports directory.

Extract from cbprJnodesConfigIBM.tar.gz file.

- jnodes0.jar

The following jars needs to be copied from <TX\_install\_dir>/jars:

- jackson-core-n.n.n.jar
- jackson-annotations-n.n.n.jar
- jackson-databind-n.n.n.jar

Or visit <https://repo1.maven.org/maven2/com/fasterxml/jackson/core/> for download.

Note: Recommend using the latest version.

For the non Docker environments,

- Copy jars to <TX\_install\_dir>/extjar.

For TX V11.0.1 and up, native based Design Server installation,

Copy the following into the directory defined in config.yaml server.persistence.libs, by default, this is set to /opt/txlibs:

- jvcwrap.jar
- jvalccyy.jar

Restart the running application ./ITX stop and then ./ITX start.

For the Docker environments,

- docker cp jnodes0.jar <brand>-server:/opt/<company>/<brand>/libs/.
- Restart the design server, i.e., **docker restart <brand>-server**.

Note: For RHEL, Docker is not supported, Podman can be used as a substitute since it provides a command line interface similar to Docker. Below is an example of using podman:

- podman cp jnodes0.jar <brand>-server:/opt/<company>/<brand>/libs/.
- Restart the design server, i.e., **podman restart <brand>-server**.

This example generates failure in the audit.log.json log file to report the translation failure due to the pre-conversion checks:

```
<IntrBkSttlmAmt>Commodities currencies {XAU, XAG, XPD, XPT} Not allowed in Field 32A and 32B.
Count of <TxInf> is greater than one.
<RtrChain><Dbtr><Pty> is present
<RtrChain><Cdtr><Pty> is present
```

1. Import the cbpr\_translation.zip project into the Design Server.

2. Open the cbpr\_translation project in Design Server and view the flow cbpr\_pacs004\_mtnnnRTN flow.

a. The Flow Description points to all the maps to be executed during the flow process, which will be called from within some of the map nodes in the flow. The following is a list of maps invoked using RUN built-in function during the flow process:

- @packagemap=cbpr\_plus/translation/mx-ml/maps/cbpr\_t9n\_mx\_pacs004\_translate/cbpr2851\_pacs004RTN\_trx
- @packagemap=cbpr\_plus/translation/mx-ml/maps/cbpr\_t9n\_mx\_pacs004\_mt20nRTN/cbpr2852\_pacs004\_mt202RTN
- @packagemap=cbpr\_plus/translation/mx-ml/maps/cbpr\_t9n\_mx\_pacs004\_mt20nRTN/cbpr2853\_pacs004\_mt205RTN
- @packagemap=cbpr\_plus/translation/mx-ml/maps/cbpr\_t9n\_mx\_pacs004\_mt103RTN/cbpr2857\_pacs004\_mt103RTN
- @packagemap=cbpr\_plus/translation/mx-ml/maps/cbpr\_t9n\_mx\_setvarlog/cbpr2501\_mxmt\_setvarlog

b. It utilizes the following nodes:

i. Source Node

- pacs\_004  
This node identifies the input data to be translated in the flow. It uses the INPUT\_FILE variable to set the location of the data.

ii. Map Node

- mx\_translate  
Runs framework map ccbpr2850\_pacs004\_mtnnnRTN\_frw, checks pre-translation conditions and translates the input file (mx or xml) into required output (mt).

iii. Target Node

- mtnnnRTN  
This node contains the resulting translation in MT format and creates the output as defined by the variable OUTPUT\_RESULT\_MT.

iv. Log Node

- audit\_log  
This node creates a log file specified by the flow variable AUDIT\_LOG.

c. It utilizes the following flow variables:

- INPUT\_FILE

Default value for the flow variable INPUT\_FILE is ..//cbpr\_plus/translation/mx-ml/data/env\_pacs\_004\_mt202rtn\_valid.xml. This is the data file to be used for translation and can be customized. It is used in the Source node pacs\_004.

- OUTPUT\_RESULT\_MT

Default value for the flow variable OUTPUT\_RESULT\_MT is mt20nRTN.out. This is the location of the output file in mt format. It can be customized.

- AUDIT\_LOG

Default value for the flow variable AUDIT\_LOG is audit.log.json. This is the location of the audit log. It is used in the Log node audit\_log. It can be customized.

- CONFIG\_FILE

Default value for the flow variable CONFIG\_FILE is ..//cbpr\_plus/translation/mx-ml/data/trx\_config.xml. This is the location of the file containing the configuration settings. It can be customized.

3. Open the flow cbpr\_pacs004\_mtnnnRTN in the Design Server. It utilizes one source node, one map node, one log node, and one target node.

4. In Design Server, create a package that contains the input files and one flow. The maps will automatically be included during deployment of the package onto the runtime server.

- **[Run the example using three different methods](#)**

You can run the example using three different methods:

## Run the example using three different methods

You can run the example using three different methods:

1. Running the example from the Design Server user interface.
2. Deploying the example to tx-rest and running it using the Swagger interface.
3. Deploying locally or to ftp server and using the flow command server process (flowcmdserver).

- **[Run the example from the Design Server user interface](#)**

Use the following steps to run the example from the Design Server user interface:

- **[Run the example using the Swagger interface](#)**

To run the example, deploy to tx-rest and run it using the Swagger interface.

- **[Run the example using the flow command server process \(flowcmdserver\)](#)**

To run the example, deploy locally or to ftp server and use the flow command server process (flowcmdserver).

## Run the example from the Design Server user interface

Use the following steps to run the example from the Design Server user interface:

1. In the Design Server, open cbpr\_pacs004\_mtnnnRTN flow and click on Run button.
2. Set toggle switch Run On Design Server and select flow input file env\_pacs\_004\_mt202rtn\_valid.xml or env\_pacs\_004\_mt202rtn\_bad.xml.  
Note: If a default path is assigned to the variable INPUT\_FILE, not selecting an input file will use the value defined for the variable.  
Flow will run and report with the green check box. The report lists execution of all nodes.
3. Right click and select View Data on each link to examine the data.
4. Right click on the node and select View Log to see the logs.

## Run the example using the Swagger interface

To run the example, deploy to tx-rest and run it using the Swagger interface.

1. Create a Web type server definition where the runtime tx-rest is installed, if you do not have a server definition to deploy in Design Server.

2. If not running, start tx-rest on the server: <DTX\_HOME>/restapi/tomcat/dtxtomcat start tx-rest.

3. Deploy the created package to the server definition in Design Server.

4. Bring up the tx-rest Swagger UI: <DTX\_HOME>/restapi/tomcat/dtxtomcat showdocs tx-rest.

5. In the Swagger Explore window, enter /tx-rest/v2/docs and click on the Explore button to view the deployed package.

You can see the cbpr\_pacs004\_mtnnnRTN section having two actions: A PUT /v2/run/cbpr\_pacs004\_mtnnnRTN and a POST /v2/run/cbpr\_pacs004\_mtnnnRTN.

6. In the PUT action:

a. Expand the PUT action and select the Try it out button.

b. Leave the input and output sections empty.

c. Under flow\_vars section, delete the entire content and replace with the commands to run the flow, for example:

```
{
 "INPUT_FILE": "C:/tests_dir/data/env_pacs_004_mt202rtn_valid.xml",
 "OUTPUT_RESULT_MT": "C:/tests_dir/data/mt202RTN.out",
 "AUDIT_LOG": "C:/tests_dir/data/audit_log.json"
}
```

d. Click Execute blue bar for running the flow. When flow completes execution, 200 response code is shown in the Server Response section. You can see information about the process flow in the Response body.

7. Refresh the browser to delete the deployed package and get the Swagger Explore back to /tx-rest/openapi.json.

There will be a DELETE /v2/packages/{name} action.

8. Expand the DELETE /v2/packages/{name} action and select the Try it out.

9. In the Name field, enter the name you gave it to the created package.

10. Select the true option from the stop query drop down, select the blue Execute bar.

You can see a response of 204 with a timestamp, if it is successful.

## Run the example using the flow command server process (flowcmdserver)

To run the example, deploy locally or to ftp server and use the flow command server process (flowcmdserver).

1. Create a server definition where the runtime is installed, if you do not have a server local definition or ftp execution definition to deploy to, in Design Server.
2. In Design Server, deploy the created package to the server definition.

3. Execute the flow using the flow command server:

```
flowcmdserver.exe --dir c:\ftpserver\deployment\flows --flow cbpr_pacs004_mtnnnRTN.json
-var "INPUT_FILE=C:\ftpserver\deployment\cbpr_plus\translation\mx-mt\data\env_pacs_004_mt202rtn_valid.xml"
--var "OUTPUT_RESULT_MT=C:\ftpserver\deployment\cbpr_plus\translation\mx-mt\data\env_pacs_004_mt202rtn_valid.out"
--var "AUDIT_LOG=C:\ftpserver\deployment\cbpr_plus\translation\mx-mt\data\env_pacs_004_mt202rtn_valid_log.json"
--audit "C:\ftpserver\deployment\cbpr_plus\translation\mx-mt\data\env_pacs_004_mt202rtn_valid_adt.json" -ad
```

4. The flow will report status similar to:

```
***Starting flow command server
```

```

Flow UUID: 85eabe2c-2302-4543-b347-9e18d65c6816
The flow audit file is: "C:\\\\ftpserver\\\\deployment\\\\cbpr_plus\\\\translation\\\\mx-
mt\\\\data\\\\env_pacs_004_mt202rtn_valid_adt.json"
Flow completed successfully
Elapsed time: 3048ms

```

5. Examine the flow output result file env\_pacs\_004\_mt202rtn\_valid.out and the audit file env\_pacs\_004\_mt202rtn\_valid\_adt.json.

## CBPR+ MX to MT (pacs.008.001.08 to MT103) Translation Example

This example demonstrates the CBPR+ MX to MT (pacs.008.001.08 to MT103) message translation using flow server.

- [What the example contains](#)**

Files included in this example are as follows:

- cbpr\_translation.zip
  - Files:
    - env\_pacs\_008\_103.xml
    - env\_pacs\_008\_bad.xml
    - trx\_config.xml
  - Schemas:
    - head.001.001.02\_pacs\_008.xsd
    - pacs.008.001.08.xsd
    - trx\_config.xsd
    - infoset.mtt
    - swift\_iso7775\_ccyy.mtt
    - swiftroute\_funds.mtt
  - Maps:
    - cbpr2700\_pacs008\_framework
    - cbpr2701\_pacs008\_translate
    - cbpr2709\_pacs008\_stp\_translate
    - cbpr2703\_pacs008\_mt103
    - cbpr2704\_pacs008\_mt103\_sgo
    - cbpr2501\_mxmt\_setvarlog
  - Flow:
    - cbpr\_pacs008\_mt103
- cbprJnodesConfigIBM.tar.gz

## What the example contains

Files included in this example are as follows:

- cbpr\_translation.zip
  - Files:
    - env\_pacs\_008\_103.xml
    - env\_pacs\_008\_bad.xml
    - trx\_config.xml
  - Schemas:
    - head.001.001.02\_pacs\_008.xsd
    - pacs.008.001.08.xsd
    - trx\_config.xsd
    - infoset.mtt
    - swift\_iso7775\_ccyy.mtt
    - swiftroute\_funds.mtt
  - Maps:
    - cbpr2700\_pacs008\_framework
    - cbpr2701\_pacs008\_translate
    - cbpr2709\_pacs008\_stp\_translate
    - cbpr2703\_pacs008\_mt103
    - cbpr2704\_pacs008\_mt103\_sgo
    - cbpr2501\_mxmt\_setvarlog
  - Flow:
    - cbpr\_pacs008\_mt103
- cbprJnodesConfigIBM.tar.gz

## How to run the example

This mx-translation will use the sample files to demonstrate the generation of SWIFT MT103 message output from a CBPR pacs.008.001.08 XML message.

The cbprJnodesConfigIBM.tar.gz file is available either in IBM\_financialpaymentsplus\_vn.n.n.n.zip or in UIProjectImports directory.

Extract from cbprJnodesConfigIBM.tar.gz file.

- jnodes0.jar

The following jars needs to be copied from <TX\_install\_dir>/jars:

- jackson-core-n.n.n.jar
- jackson-annotations-n.n.n.jar
- jackson-databind-n.n.n.jar

Or visit <https://repo1.maven.org/maven2/com/fasterxml/jackson/core/> for download.

Note: Recommend using the latest version.

For the non Docker environments,

- Copy jars to <TX\_install\_dir>/extjar.

For TX V11.0.1 and up, native based Design Server installation,

Copy the following into the directory defined in config.yaml server.persistence.libs, by default, this is set to /opt/tlibs:

- jvcwrap.jar
- jvalccyy.jar

Restart the running application ./ITX stop and then ./ITX start.

For the Docker environments,

- docker cp jnodes0.jar <brand>-server:/opt/<company>/<brand>/libs/.
- Restart the design server, i.e., **docker restart** <brand>-server.

Note: For RHEL, Docker is not supported, Podman can be used as a substitute since it provides a command line interface similar to Docker. Below is an example of using podman:

- podman cp jnodes0.jar <brand>-server:/opt/<company>/<brand>/libs/.
- Restart the design server, i.e., **podman restart** <brand>-server.

This example may generate failure in the audit.log.json log file to report the translation failure due to the pre-conversion checks:

```
<IntrBkSttlmAmt> Commodities currencies {XAU, XAG, XPD, XPT} Not allowed in Field 32A and 32B.
```

1. Import the cbpr\_translation.zip project into the Design Server.
  2. Open the cbpr\_translation project in Design Server and view the flow cbpr\_pacs008\_mt103.
    - a. The Flow Description points to all the maps to be executed during the flow process, which will be called from within some of the map nodes in the flow. The following is a list of maps invoked using RUN built-in function during the flow process:
      - @packagemap=cbpr\_plus/translation/mx-mt/maps/cbpr\_t9n\_mx\_pacs008\_translate/cbpr2701\_pacs008\_translate
      - @packagemap=cbpr\_plus/translation/mx-mt/maps/cbpr\_t9n\_mx\_pacs008\_stp\_translate/cbpr2709\_pacs008\_stp\_translate
      - @packagemap=cbpr\_plus/translation/mx-mt/maps/cbpr\_t9n\_mx\_pacs008\_mt103/cbpr2703\_pacs008\_mt103
      - @packagemap=cbpr\_plus/translation/mx-mt/maps/cbpr\_t9n\_mx\_pacs008\_mt103\_sgo/cbpr2704\_pacs008\_mt103\_sgo
      - @packagemap=cbpr\_plus/translation/mx-mt/maps/cbpr\_t9n\_mx\_setvarlog/cbpr2501\_mxmt\_setvarlog
    - b. It utilizes the following nodes:
      - i. Source Node
        - pacs\_008
 

This node identifies the input data to be translated in the flow. It uses the INPUT\_FILE variable to set the location of the data.
      - ii. Map Node
        - mx\_translate
 

Runs framework map cbpr2700\_pacs008\_framework, checks pre-translation conditions and translates the input file (mx or xml) into required output (mt).
      - iii. Target Node
        - mt103
 

This node contains the resulting translation in MT format and creates the output as defined by the variable OUTPUT\_RESULT\_MT.
      - iv. Log Node
        - audit\_log
 

This node creates a log file specified by the flow variable AUDIT\_LOG.
  - c. It utilizes the following flow variables:
    - INPUT\_FILE
 

Default value for the flow variable INPUT\_FILE is ..//cbpr\_plus/translation/mx-mt/data/env\_pacs\_008\_103.xml. This is the data file to be used for translation and can be customized. It is used in the Source node pacs\_008.
    - OUTPUT\_RESULT\_MT
 

Default value for the flow variable OUTPUT\_RESULT\_MT is mt103.out. This is the location of the output file in mt format. It is used in the Target node mt103. It can be customized.
    - AUDIT\_LOG
 

Default value for the flow variable AUDIT\_LOG is audit.log.json. This is the location of the audit log. It is used in the Log node audit\_log. It can be customized.
    - CONFIG\_FILE
 

Default value for the flow variable CONFIG\_FILE is ..//cbpr\_plus/translation/mx-mt/data/trx\_config.xml. This is the location of the file containing the configuration settings. It can be customized.
  3. Open the flow cbpr\_pacs008\_mt103 in the Design Server. It utilizes one source node, one map node, one log node, and one target node.
  4. In Design Server, create a package that contains the input files and one flow. The maps will automatically be included during deployment of the package onto the runtime server.
- **[Run the example using three different methods](#)**

You can run the example using three different methods:

## Run the example using three different methods

You can run the example using three different methods:

1. Running the example from the Design Server user interface.
2. Deploying the example to tx-rest and running it using the Swagger interface.
3. Deploying locally or to ftp server and using the flow command server process (flowcmdserver).

- **[Run the example from the Design Server user interface](#)**

Use the following steps to run the example from the Design Server user interface:

- **[Run the example using the Swagger interface](#)**

To run the example, deploy to tx-rest and run it using the Swagger interface.

- **[Run the example using the flow command server process \(flowcmdserver\)](#)**

To run the example, deploy locally or to ftp server and use the flow command server process (flowcmdserver).

## Run the example from the Design Server user interface

Use the following steps to run the example from the Design Server user interface:

1. In the Design Server, open cbpr\_pacs008\_mt103 flow and click on Run button.
2. Set toggle switch Run On Design Server and select flow input file env\_pacs\_008\_103.xml or env\_pacs\_008\_bad.xml.  
Note: If a default path is assigned to the variable INPUT\_FILE, not selecting an input file will use the value defined for the variable. Flow will run and report with the green check box. The report lists execution of all nodes.
3. Right click and select View Data on each link to examine the data.
4. Right click on the node and select View Log to see the logs.

## Run the example using the Swagger interface

To run the example, deploy to tx-rest and run it using the Swagger interface.

1. Create a Web type server definition where the runtime tx-rest is installed, if you do not have a server definition to deploy in Design Server.
2. If not running, start tx-rest on the server: <DTX\_HOME>/restapi/tomcat/dtxtomcat start tx-rest.
3. Deploy the created package to the server definition in Design Server.
4. Bring up the tx-rest Swagger UI: <DTX\_HOME>/restapi/tomcat/dtxtomcat showdocs tx-rest.
5. In the Swagger Explore window, enter /tx-rest/v2/docs and click on the Explore button to view the deployed package.  
You can see the cbpr\_pacs008\_mt103 section having two actions: A PUT /v2/run/cbpr\_pacs008\_mt103 and a POST /v2/run/cbpr\_pacs008\_mt103.
6. In the PUT action:
  - a. Expand the PUT action and select the Try it out button.
  - b. Leave the input and output sections empty.
  - c. Under flow\_vars section, delete the entire content and replace with the commands to run the flow, for example:

```
{
 "INPUT_FILE": "C:/tests_dir/data/env_pacs_008_103.xml",
 "OUTPUT_RESULT_MT": "C:/tests_dir/data/mt103.out",
 "AUDIT_LOG": "C:/tests_dir/data/audit_log.json"
}
```
- d. Click Execute blue bar for running the flow. When flow completes execution, 200 response code is shown in the Server Response section. You can see information about the process flow in the Response body.
7. Refresh the browser to delete the deployed package and get the Swagger Explore back to /tx-rest/openapi.json.  
There will be a DELETE /v2/packages/{name} action.
8. Expand the DELETE /v2/packages/{name} action and select the Try it out.
9. In the Name field, enter the name you gave it to the created package.
10. Select the true option from the stop query drop down, select the blue Execute bar.  
You can see a response of 204 with a timestamp, if it is successful.

## Run the example using the flow command server process (flowcmdserver)

To run the example, deploy locally or to ftp server and use the flow command server process (flowcmdserver).

1. Create a server definition where the runtime is installed, if you do not have a server local definition or ftp execution definition to deploy to, in Design Server.
2. In Design Server, deploy the created package to the server definition. For example, deploy to c:\ftpserver\deployment directory.
3. Execute the flow using the flow command server:

```
flowcmdserver.exe --dir c:\ftpserver\deployment\flows --flow cbpr_pacs008_mt103.json
--var "INPUT_FILE=C:\ftpserver\deployment\cbpr_plus\translation\mx-mt\data\env_pacs_008_103.xml"
--var "OUTPUT_RESULT_MT=C:\ftpserver\deployment\cbpr_plus\translation\mx-mt\data\env_pacs_008_103.out"
--var "AUDIT_LOG=C:\ftpserver\deployment\cbpr_plus\translation\mx-mt\data\env_pacs_008_103_log.json"
--audit "C:\ftpserver\deployment\cbpr_plus\translation\mx-mt\data\env_pacs_008_103_adt.json" -ad
```

4. The flow will report status similar to:

```
***Starting flow command server

Flow UUID: 85eabe2c-2302-4543-b347-9e18d65c6816
The flow audit file is: "C:\\\\ftpserver\\\\deployment\\\\cbpr_plus\\\\translation\\\\mx-
mt\\\\data\\\\env_pacs_008_103_adt.json"
Flow completed successfully
Elapsed time: 3048ms
```

5. Examine the flow output result file env\_pacs\_008\_103.out and the audit file env\_pacs\_008\_103\_adt.json.

## CBPR+ MX to MT (pacs.009.001.08 to MT202) Translation Example

This example demonstrates the CBPR+ MX to MT (pacs.009.001.08 to MT202) message translation using flow server.

- [What the example contains](#)

Files included in this example are as follows:

- [How to run the example](#)

This mx-mt translation will use the sample files to demonstrate the generation of SWIFT MT202 message output from a CBPR pacs.009.001.08 XML message.

## What the example contains

Files included in this example are as follows:

- cbpr\_translation.zip
  - Files:
    - env\_pacs\_009\_mt202cor\_valid.xml
    - env\_pacs\_009\_cor\_bad.xml
    - trx\_config.xml
  - Schemas:
    - head.001.001.02\_pacs\_009.xsd
    - pacs.009.001.08.xsd
    - trx\_config.xsd
    - infoSet.mtt
    - swift\_iso7775\_ccyy.mtt
    - swiftroute\_funds.mtt
  - Maps:
    - cbpr2710\_pacs009\_mt202\_framework
    - cbpr2713\_pacs009\_mt202adv\_trx
    - cbpr2714\_pacs009\_mt202cov\_trx
    - cbpr2712\_pacs009\_mt202\_trx
    - cbpr2715\_pacs009\_mt202
    - cbpr2501\_mxmt\_setvarlog
  - Flow:
    - cbpr\_pacs009\_mt202
- cbprJnodesConfigIBM.tar.gz

## How to run the example

This mx-xt translation will use the sample files to demonstrate the generation of SWIFT MT202 message output from a CBPR pacs.009.001.08 XML message.

The cbprJnodesConfigIBM.tar.gz file is available either in IBM\_financialpaymentsplus\_vn.n.n.n.zip or in UIProjectImports directory.

Extract from cbprJnodesConfigIBM.tar.gz file.

- jnodes0.jar

The following jars needs to be copied from <TX\_install\_dir>/jars:

- jackson-core-n.n.n.jar
- jackson-annotations-n.n.n.jar
- jackson-databind-n.n.n.jar

Or visit <https://repo1.maven.org/maven2/com/fasterxml/jackson/core/> for download.

Note: Recommend using the latest version.

For the non Docker environments,

- Copy jars to <TX\_install\_dir>/extjar.

For TX V11.0.1 and up, native based Design Server installation,

Copy the following into the directory defined in config.yaml server.persistence.libs, by default, this is set to /opt/txls:

- jvcwrap.jar
- jvalccyy.jar

Restart the running application ./ITX stop and then ./ITX start.

For the Docker environments,

- docker cp jnodes0.jar <brand>-server:/opt/<company>/<brand>/libs/.
- Restart the design server, i.e., **docker restart <brand>-server**.

Note: For RHEL, Docker is not supported, Podman can be used as a substitute since it provides a command line interface similar to Docker. Below is an example of using podman:

- podman cp jnodes0.jar <brand>-server:/opt/<company>/<brand>/libs/.
- Restart the design server, i.e., **podman restart <brand>-server**.

This example may generate failure in the audit.log.json log file to report the translation failure due to the pre-conversion checks:

```
<IntrBkSttlmAmt> Commodities currencies {XAU, XAG, XPD, XPT} Not allowed in Field 32A and 32B.
Count of <CdtTrfTxInf> is greater than one
```

1. Import the cbpr\_translation.zip project into the Design Server.

2. Open the cbpr\_translation project in Design Server and view the flow cbpr\_pacs009\_mt202.

a. The Flow Description points to all the maps to be executed during the flow process, which will be called from within some of the map nodes in the flow. The following is a list of maps invoked using RUN built-in function during the flow process:

- @packagemap=cbpr\_plus/translation/mx-xt/translation/mx-xt/maps/cbpr\_t9n\_mx\_pacs009\_adv\_translate/cbpr2713\_pacs009\_mt202adv\_trx
- @packagemap=cbpr\_plus/translation/mx-xt/translation/mx-xt/maps/cbpr\_t9n\_mx\_pacs009\_cov\_translate/cbpr2714\_pacs009\_mt202cov\_trx

- @packagemap=cbpr\_plus/translation/mx-mt/maps/cbpr\_t9n\_mx\_pacs009\_translate/cbpr2712\_pacs009\_mt202\_trx
  - @packagemap=cbpr\_plus/translation/mx-mt/maps/cbpr\_t9n\_mx\_pacs009\_mt20n/cbpr2715\_pacs009\_mt202
  - @packagemap=cbpr\_plus/translation/mx-mt/maps/cbpr\_t9n\_mx\_setvarlog/cbpr2501\_mxmt\_setvarlog
- b. It utilizes the following nodes:
- i. Source Node
    - pacs\_009
 

This node identifies the input data to be translated in the flow. It uses the INPUT\_FILE variable to set the location of the data.
  - ii. Map Node
    - mx\_translate
 

Runs framework map cbpr2710\_pacs009\_mt202\_framework, checks pre-translation conditions and translates the input file (mx or xml) into required output (mt).
  - iii. Target Node
    - mt202
 

This node contains the resulting translation in MT format and creates the output as defined by the variable OUTPUT\_RESULT\_MT.
  - iv. Log Node
    - audit\_log
 

This node creates a log file specified by the flow variable AUDIT\_LOG.
- c. It uses the following flow variables:
- INPUT\_FILE
 

Default value for the flow variable INPUT\_FILE is ..../cbpr\_plus/translation/mx-mt/data/env\_pacs\_009\_mt202cor\_valid.xml. This is the data file to be used for translation and can be customized. It is used in the Source node pacs\_009.
  - OUTPUT\_RESULT\_MT
 

Default value for the flow variable OUTPUT\_RESULT\_MT is mt202.out. This is the location of the output file in mt format. It is used in the Target node mt202. It can be customized.
  - AUDIT\_LOG
 

Default value for the flow variable AUDIT\_LOG is audit.log.json. This is the location of the audit log. It is used in the Log node audit\_log. It can be customized.
  - CONFIG\_FILE
 

Default value for the flow variable CONFIG\_FILE is ..../cbpr\_plus/translation/mx-mt/data/trx\_config.xml. This is the location of the file containing the configuration settings. It can be customized.

3. Open the flow cbpr\_pacs009\_mt202 in the Design Server. It utilizes one source node, one map node, one log node, and one target node.
  4. In Design Server, create a package that contains the input files and one flow. The maps will automatically be included during deployment of the package onto the runtime server.
- [\*\*Run the example using three different methods\*\*](#)  
You can run the example using three different methods:

## Run the example using three different methods

You can run the example using three different methods:

1. Running the example from the Design Server user interface.
  2. Deploying the example to tx-rest and running it using the Swagger interface.
  3. Deploying locally or to ftp server and using the flow command server process (flowcmdserver).
- [\*\*Run the example from the Design Server user interface\*\*](#)  
Use the following steps to run the example from the Design Server user interface:
  - [\*\*Run the example using the Swagger interface\*\*](#)  
To run the example, deploy to tx-rest and run it using the Swagger interface.
  - [\*\*Run the example using the flow command server process \(flowcmdserver\)\*\*](#)  
To run the example, deploy locally or to ftp server and use the flow command server process (flowcmdserver).

## Run the example from the Design Server user interface

Use the following steps to run the example from the Design Server user interface:

1. In the Design Server, open cbpr\_pacs009\_mt202 flow and click on Run button.
2. Set toggle switch Run On Design Server and select flow input file env\_pacs\_009\_mt202cor\_valid.xml or env\_pacs\_009\_cor\_bad.xml.  
Note: If a default path is assigned to the variable INPUT\_FILE, not selecting an input file will use the value defined for the variable.  
Flow will run and report with the green check box. The report lists execution of all nodes.
3. Right click and select View Data on each link to examine the data.
4. Right click on the node and select View Log to see the logs.

## Run the example using the Swagger interface

To run the example, deploy to tx-rest and run it using the Swagger interface.

1. Create a Web type server definition where the runtime tx-rest is installed, if you do not have a server definition to deploy in Design Server.
2. If not running, start tx-rest on the server: <DTX\_HOME>/restapi/tomcat/dtxtomcat start tx-rest.
3. Deploy the created package to the server definition in Design Server.
4. Bring up the tx-rest Swagger UI: <DTX\_HOME>/restapi/tomcat/dtxtomcat showdocs tx-rest.
5. In the Swagger Explore window, enter /tx-rest/v2/docs and click on the Explore button to view the deployed package.  
You can see the cbpr\_pacs009\_mt202 section having two actions: A PUT /v2/run/cbpr\_pacs009\_mt202 and a POST /v2/run/cbpr\_pacs009\_mt202.
6. In the PUT action:
  - a. Expand the PUT action and select the Try it out button.
  - b. Leave the input and output sections empty.
  - c. Under flow\_vars section, delete the entire content and replace with the commands to run the flow, for example:

```
{
 "INPUT_FILE": "C:/tests_dir/data/env_pacs_009_mt202cor_valid.xml",
 "OUTPUT_RESULT_MT": "C:/tests_dir/data/mt202.out",
 "AUDIT_LOG": "C:/tests_dir/data/audit_log.json"
}
```
- d. Click Execute blue bar for running the flow. When flow completes execution, 200 response code is shown in the Server Response section. You can see information about the process flow in the Response body.
7. Refresh the browser to delete the deployed package and get the Swagger Explore back to /tx-rest/openapi.json.  
There will be a DELETE /v2/packages/{name} action.
8. Expand the DELETE /v2/packages/{name} action and select the Try it out.
9. In the Name field, enter the name you gave it to the created package.
10. Select the true option from the stop query drop down, select the blue Execute bar.  
You can see a response of 204 with a timestamp, if it is successful.

## Run the example using the flow command server process (flowcmdserver)

To run the example, deploy locally or to ftp server and use the flow command server process (flowcmdserver).

1. Create a server definition where the runtime is installed, if you do not have a server local definition or ftp execution definition to deploy to, in Design Server.
2. In Design Server, deploy the created package to the server definition. For example, deploy to c:\ftpserver\deployment directory.
3. Execute the flow using the flow command server:

```
flowcmdserver.exe --dir c:\ftpserver\deployment\flows --flow cbpr_pacs009_mt202.json
--var "INPUT_FILE=C:\ftpserver\deployment\cbpr_plus\translation\mx-mt\data\env_pacs_009_mt202cor_valid.xml"
--var "OUTPUT_RESULT_MT=C:\ftpserver\deployment\cbpr_plus\translation\mx-mt\data\env_pacs_009_mt202cor_valid.out"
--var "AUDIT_LOG=C:\ftpserver\deployment\cbpr_plus\translation\mx-mt\data\env_pacs_009_mt202cor_valid_log.json"
--audit "C:\ftpserver\deployment\cbpr_plus\translation\mx-mt\data\env_pacs_009_mt202cor_valid_adt.json" -ad
```

4. The flow will report status similar to:

```
***Starting flow command server

Flow UUID: 85eabe2c-2302-4543-b347-9e18d65c6816
The flow audit file is: "C:\\ftpserver\\deployment\\\\cbpr_plus\\\\translation\\\\mx-
mt\\\\data\\\\env_pacs_009_mt202cor_valid_adt.json"
Flow completed successfully
Elapsed time: 3048ms
```

5. Examine the flow output result file env\_pacs\_009\_cor\_valid.out and the audit file env\_pacs\_009\_mt202cor\_valid\_adt.json.

## CBPR+ MX to MT (pacs.009.001.08 to MT205) Translation Example

This example demonstrates the CBPR+ MX to MT (pacs.009.001.08 to MT205) message translation using flow server.

- **What the example contains**  
Files included in this example are as follows:
- **How to run the example**  
This mx-mt translation will use the sample files to demonstrate the generation of SWIFT MT205 message output from a CBPR pacs.009.001.08 XML message.

## What the example contains

Files included in this example are as follows:

- cbpr\_translation.zip
  - Files:
    - env\_pacs\_009\_mt205cor\_valid.xml
    - env\_pacs\_009\_cor\_bad.xml
    - trx\_config.xml
  - Schemas:
    - head.001.001.02\_pacs\_009.xsd
    - pacs.009.001.08.xsd
    - trx\_config.xsd
    - infoset.mtt
    - swift\_iso7775\_ccyy.mtt
    - swiftroute\_funds.mtt

- Maps:
    - cbpr2724\_pacs009\_mt205\_framework
    - cbpr2725\_pacs009\_mt205\_trx
    - cbpr2726\_pacs009\_mt205cov\_trx
    - cbpr2727\_pacs009\_mt205
    - cbpr2501\_mxmt\_setvarlog
  - Flow:
    - cbpr\_pacs009\_mt205
  - cbprJnodesConfigIBM.tar.gz
- 

## How to run the example

This mx-*mt* translation will use the sample files to demonstrate the generation of SWIFT MT205 message output from a CBPR pacs.009.001.08 XML message.

The cbprJnodesConfigIBM.tar.gz file is available either in IBM\_financialpaymentsplus\_vn.n.n.n.zip or in UIProjectImports directory.

Extract from cbprJnodesConfigIBM.tar.gz file.

- jnodes0.jar

The following jars needs to be copied from <TX\_install\_dir>/jars:

- jackson-core-n.n.n.jar
- jackson-annotations-n.n.n.jar
- jackson-databind-n.n.n.jar

Or visit <https://repo1.maven.org/maven2/com/fasterxml/jackson/core/> for download.

Note: Recommend using the latest version.

For the non Docker environments,

- Copy jars to <TX\_install\_dir>/extjar.

For TX V11.0.1 and up, native based Design Server installation,

Copy the following into the directory defined in config.yaml server.persistence.libs, by default, this is set to /opt/tlxlibs:

- jvcwrap.jar
- jvalccyy.jar

Restart the running application ./ITX stop and then ./ITX start.

For the Docker environments,

- docker cp jnodes0.jar <brand>-server:/opt/<company>/<brand>/libs/.
- Restart the design server, i.e., **docker restart <brand>-server**.

Note: For RHEL, Docker is not supported, Podman can be used as a substitute since it provides a command line interface similar to Docker. Below is an example of using podman:

- podman cp jnodes0.jar <brand>-server:/opt/<company>/<brand>/libs/.
- Restart the design server, i.e., **podman restart <brand>-server**.

This example may generate failure in the audit.log.json log file to report the translation failure due to the pre-conversion checks:

```
<IntrBkSttlMntAmt>Commodities currencies {XAU, XAG, XPD, XPT} Not allowed in Field 32A and 32B.
Count of <CdtTrfTxInf> is greater than one.
```

1. Import the cbpr\_translation.zip project into the Design Server.

2. Open the cbpr\_translation project in Design Server and view the flow cbpr\_pacs009\_mt205.

a. The Flow Description points to all the maps to be executed during the flow process, which will be called from within some of the map nodes in the flow. The following is a list of maps invoked using RUN built-in function during the flow process:

- @packagemap=cbpr\_plus/translation/mx-*mt*/maps/cbpr\_t9n\_mx\_pacs009\_translate/cbpr2725\_pacs009\_mt205\_trx
- @packagemap=cbpr\_plus/translation/mx-*mt*/maps/cbpr\_t9n\_mx\_pacs009\_cov\_translate/cbpr2726\_pacs009\_mt205cov\_trx
- @packagemap=cbpr\_plus/translation/mx-*mt*/maps/cbpr\_t9n\_mx\_pacs009\_mt20n/cbpr2727\_pacs009\_mt205
- @packagemap=cbpr\_plus/translation/mx-*mt*/maps/cbpr\_t9n\_mx\_setvarlog/cbpr2501\_mxmt\_setvarlog

b. It utilizes the following nodes:

i. Source Node

- pacs\_009

This node identifies the input data to be translated in the flow. It uses the INPUT\_FILE variable to set the location of the data.

ii. Map Node

- mx\_translate

Runs framework map cbpr2724\_pacs009\_mt205\_framework, checks pre-translation conditions and translates the input file (mx or xml) into required output (mt).

iii. Target Node

- mt205

This node contains the resulting translation in MT format and creates the output as defined by the variable OUTPUT\_RESULT\_MT.

iv. Log Node

- audit\_log

This node creates a log file specified by the flow variable AUDIT\_LOG.

c. It utilizes the following flow variables:

- INPUT\_FILE  
Default value for flow variable INPUT\_FILE is ..//cbpr\_plus/translation/mx-mlt/data/env\_pacs\_009\_mt205cor\_valid.xml. This is the data file to be used for translation and can be customized. It is used in the Source node pacs\_009.
- OUTPUT\_RESULT\_MTLT  
Default value for the flow variable OUTPUT\_RESULT\_MTLT is mt205.out. This is the location of the output file in mt format. It is used in the Target node mt205. It can be customized.
- AUDIT\_LOG  
Default value for the flow variable AUDIT\_LOG is audit.log.json. This is the location of the audit log. It is used in the Log node audit\_log. It can be customized.
- CONFIG\_FILE  
Default value for the flow variable CONFIG\_FILE is ..//cbpr\_plus/translation/mx-mlt/data/trx\_config.xml. This is the location of the file containing the configuration settings. It can be customized.

3. Open the flow cbpr\_pacs009\_mt205 in the Design Server. It utilizes one source node, one map node, one log node, and one target node.

4. In Design Server, create a package that contains the input files and one flow. The maps will automatically be included during deployment of the package onto the runtime server.

- [Run the example using three different methods](#)

You can run the example using three different methods:

---

## Run the example using three different methods

You can run the example using three different methods:

1. Running the example from the Design Server user interface.
2. Deploying the example to tx-rest and running it using the Swagger interface.
3. Deploying locally or to ftp server and using the flow command server process (flowcmdserver).

- [Run the example from the Design Server user interface](#)

Use the following steps to run the example from the Design Server user interface:

- [Run the example using the Swagger interface](#)

To run the example, deploy to tx-rest and run it using the Swagger interface.

- [Run the example using the flow command server process \(flowcmdserver\)](#)

To run the example, deploy locally or to ftp server and use the flow command server process (flowcmdserver).

---

## Run the example from the Design Server user interface

Use the following steps to run the example from the Design Server user interface:

1. In the Design Server, open cbpr\_pacs009\_205 flow and click on Run button.
2. Set toggle switch Run On Design Server and select flow input file env\_pacs\_009\_mt205cor\_valid.xml or env\_pacs\_009\_cor\_bad.xml.  
Note: If a default path is assigned to the variable INPUT\_FILE, not selecting an input file will use the value defined for the variable.  
Flow will run and report with the green check box. The report lists execution of all nodes.
3. Right click and select View Data on each link to examine the data.
4. Right click on the node and select View Log to see the logs.

---

## Run the example using the Swagger interface

To run the example, deploy to tx-rest and run it using the Swagger interface.

1. Create a Web type server definition where the runtime tx-rest is installed, if you do not have a server definition to deploy in Design Server.
2. If not running, start tx-rest on the server: <DTX\_HOME>/restapi/tomcat/dtxtomcat start tx-rest.
3. Deploy the created package to the server definition in Design Server.
4. Bring up the tx-rest Swagger UI: <DTX\_HOME>/restapi/tomcat/dtxtomcat showdocs tx-rest.
5. In the Swagger Explore window, enter /tx-rest/v2/docs and click on the Explore button to view the deployed package.  
You can see the cbpr\_pacs009\_mt205 section having two actions: A PUT /v2/run/cbpr\_pacs009\_mt205 and a POST /v2/run/cbpr\_pacs009\_mt205.
6. In the PUT action:
  - a. Expand the PUT action and select the Try it out button.
  - b. Leave the input and output sections empty.
  - c. Under flow\_vars section, delete the entire content and replace with the commands to run the flow, for example:

```
{
 "INPUT_FILE": "C:/tests_dir/data/env_pacs_009_mt205cor_valid.xml",
 "OUTPUT_RESULT_MTLT": "C:/tests_dir/data/mt205.out",
 "AUDIT_LOG": "C:/tests_dir/data/audit_log.json"
}
```

- d. Click Execute blue bar for running the flow. When flow completes execution, 200 response code is shown in the Server Response section. You can see information about the process flow in the Response body.
7. Refresh the browser to delete the deployed package and get the Swagger Explore back to /tx-rest/openapi.json.  
There will be a DELETE /v2/packages/{name} action.

8. Expand the DELETE /v2/packages/{name} action and select the Try it out.
9. In the Name field, enter the name you gave it to the created package.
10. Select the true option from the stop query drop down, select the blue Execute bar.  
You can see a response of 204 with a timestamp, if it is successful.

## Run the example using the flow command server process (flowcmdserver)

To run the example, deploy locally or to ftp server and use the flow command server process (flowcmdserver).

1. Create a server definition where the runtime is installed, if you do not have a server local definition or ftp execution definition to deploy to, in Design Server.
2. In Design Server, deploy the created package to the server definition. For example, deploy to c:\ftpserver\deployment directory.
3. Execute the flow using the flow command server:

```
flowcmdserver.exe --dir c:\ftpserver\deployment\flows --flow cbpr_pacs009_mt205.json
--var "INPUT_FILE=C:\ftpserver\deployment\cbpr_plus\translation\mx-mt\data\env_pacs_009_mt205cor_valid.xml"
--var "OUTPUT_RESULT_MT=C:\ftpserver\deployment\cbpr_plus\translation\mx-mt\data\env_pacs_009_mt205cor_valid.out"
--var "AUDIT_LOG=C:\ftpserver\deployment\cbpr_plus\translation\mx-mt\data\env_pacs_009_mt205cor_valid_log.json"
--audit "C:\ftpserver\deployment\cbpr_plus\translation\mx-mt\data\env_pacs_009_mt205cor_valid_adt.json" -ad
```

4. The flow will report status similar to:

```
***Starting flow command server
Flow UUID: 85eabe2c-2302-4543-b347-9e18d65c6816
The flow audit file is: "C:\\\\ftpserver\\\\deployment\\\\cbpr_plus\\\\translation\\\\mx-
mt\\\\data\\\\env_pacs_009_mt205cor_valid_adt.json"
Flow completed successfully
Elapsed time: 3048ms
```

5. Examine the flow output result file env\_pacs\_009\_205cor\_valid.out and the audit file env\_pacs\_009\_mt205cor\_valid\_adt.json.

## CBPR+ MT to MX (MT900/MT910 to camt.054.001.08) Translation Example

This example demonstrates the CBPR+ MT to MX (MT900/MT910 to camt.054.001.08) message translation using flow server.

- **[What the example contains](#)**

Files included in this example are as follows:

- **[How to run the example](#)**

This mt-mx translation will use the sample files to demonstrate the generation of CBPR+ camt.054.001.08 XML message output from a SWIFT MT900 or MT910 message.

## What the example contains

Files included in this example are as follows:

- cbpr\_translation.zip
  - Files:
    - MT900.inp
    - MT910.inp
  - Schemas:
    - head.001.001.02\_camt\_054.xsd
    - camt.054.001.08.xsd
    - swift\_iso7775\_ccyy.mtt
    - swiftroute\_funds.mtt
    - infoset.mtt
  - Maps:
    - cbpr1500\_mt9n0\_framework
    - cbpr1501\_mt9n0\_camt\_054
    - cbpr1502\_mtmx\_setvarlog
  - Flow:
    - cbpr\_mt9n0\_camt054
- cbprJnodesConfigIBM.tar.gz

## How to run the example

This mt-mx translation will use the sample files to demonstrate the generation of CBPR+ camt.054.001.08 XML message output from a SWIFT MT900 or MT910 message.

The cbprJnodesConfigIBM.tar.gz file is available either in IBM\_financialpaymentsplus\_vn.n.n.zip or in UIProjectImports directory.

Extract from cbprJnodesConfigIBM.tar.gz file.

- jnodes0.jar

The following jars needs to be copied from <TX\_install\_dir>/jars:

- jackson-core-n.n.n.jar
- jackson-annotations-n.n.n.jar
- jackson-databind-n.n.n.jar

Or visit <https://repo1.maven.org/maven2/com/fasterxml/jackson/core/> for download.

Note: Recommend using the latest version.

For the non Docker environments,

- Copy jars to <TX\_install\_dir>/extjar.

For TX V11.0.1 and up, native based Design Server installation,

Copy the following into the directory defined in config.yaml server.persistence.libs, by default, this is set to /opt/txlibs:

- jvcwrap.jar
- jvalccyy.jar

Restart the running application ./ITX stop and then ./ITX start.

For the Docker environments,

- docker cp jnodes0.jar <brand>-server:/opt/<company>/<brand>/libs/.
- Restart the design server, i.e., **docker restart <brand>-server**.

Note: For RHEL, Docker is not supported, Podman can be used as a substitute since it provides a command line interface similar to Docker. Below is an example of using podman:

- podman cp jnodes0.jar <brand>-server:/opt/<company>/<brand>/libs/.
- Restart the design server, i.e., **podman restart <brand>-server**.

This example may generate failure in the audit.log.json log file to report the translation failure due to the pre-conversion checks:

**If input file is SWIFT message other than MT900 or MT910 or not a SWIFT message.**

1. Import the cbpr\_translation.zip project into the Design Server.
2. Open the cbpr\_translation project in Design Server and view the flow cbpr\_mt9n0\_camt054.
  - a. The Flow Description points to all the maps to be executed during the flow process, which will be called from within some of the map nodes in the flow. The following is a list of maps invoked using RUN built-in function during the flow process:
    - @packagemap=cbpr\_plus/translation/mt-mx/maps/cbpr\_t9n\_mt9n0\_camt054/cbpr1501\_mt9n0\_camt\_054
    - @packagemap=cbpr\_plus/translation/mt-mx/maps/cbpr\_t9n\_mt\_setvarlog/cbpr1502\_mtmx\_setvarlog
  - b. It utilizes the following nodes:
    - i. Source Node
      - mt\_9n0
 

This node identifies the input data to be translated in the flow. It uses the INPUT\_FILE variable to set the location of the data.
    - ii. Map Node
      - mt\_translate
 

Runs map cbpr1500\_mt9n0\_framework which set flow variables, checks pre-translation conditions and translates the input file (mt) into required output (XML).
    - iii. Target Node
      - camt.054
 

This node contains the resulting translation in XML format and creates the output as defined by the variable OUTPUT\_DOC\_RESULT\_XML.
    - iv. Log Node
      - audit\_log
 

This node creates a log file specified by the flow variable AUDIT\_LOG.
  - c. It utilizes the following flow variables:
    - INPUT\_FILE
 

Default value for the flow variable INPUT\_FILE is ..//cbpr\_plus/translation/mt-mx/data/MT900.inp. This is the data file to be used for translation and can be customized. It is used in the Source node mt\_9n0.
    - OUTPUT\_DOC\_RESULT\_XML
 

Default value for the flow variable OUTPUT\_DOC\_RESULT\_XML is env\_camt\_054\_out.xml. This is the location of the output file in mt format. It is used in the Target node camt.054. It can be customized.
    - AUDIT\_LOG
 

Default value for the flow variable AUDIT\_LOG is audit.log.json. This is the location of the audit log. It is used in the Log node audit\_log. It can be customized.
    - CONFIG\_FILE
 

Default value for the flow variable CONFIG\_FILE is ..//cbpr\_plus/translation/mt-mx/data/trx\_config.xml. This is the location of the file containing the configuration settings. It can be customized.
3. Open the flow cbpr\_mt9n0\_camt054 in the Design Server. It utilizes one source node, one map node, one log node, and one target node.
4. In Design Server, create a package that contains the input files and one flow. The maps will automatically be included during deployment of the package onto the runtime server.
- **Run the example using three different methods**  
You can run the example using three different methods:

## Run the example using three different methods

You can run the example using three different methods:

1. Running the example from the Design Server user interface.
2. Deploying the example to tx-rest and running it using the Swagger interface.
3. Deploying locally or to ftp server and using the flow command server process (flowcmdserver).

- [\*\*Run the example from the Design Server user interface\*\*](#)

Use the following steps to run the example from the Design Server user interface:

- [\*\*Run the example using the Swagger interface\*\*](#)
- [\*\*Run the example using the flow command server process \(flowcmdserver\)\*\*](#)

To run the example, deploy to tx-rest and run it using the Swagger interface.

To run the example, deploy locally or to ftp server and use the flow command server process (flowcmdserver).

---

## Run the example from the Design Server user interface

Use the following steps to run the example from the Design Server user interface:

1. In the Design Server, open cbpr\_mt9n0\_camt054 flow and click on Run button.

2. Set toggle switch Run On Design Server and select SWIFT MT message input file MT900.inp/MT910.inp .

Note: If a default path is assigned to the variable INPUT\_FILE, not selecting an input file will use the value defined for the variable.

Flow will run and report with the green check box. The report lists execution of all nodes.

3. Right click and select View Data on each link to examine the data.

4. Right click on the node and select View Log to see the logs.

---

## Run the example using the Swagger interface

To run the example, deploy to tx-rest and run it using the Swagger interface.

1. Create a Web type server definition where the runtime tx-rest is installed, if you do not have a server definition to deploy in Design Server.

2. If not running, start tx-rest on the server: <DTX\_HOME>/restapi/tomcat/dtxtomcat start tx-rest.

3. Deploy the created package to the server definition in Design Server.

4. Bring up the tx-rest Swagger UI: <DTX\_HOME>/restapi/tomcat/dtxtomcat showdocs tx-rest.

5. In the Swagger Explore window, enter /tx-rest/v2/docs and click on the Explore button to view the deployed package.

You should see a cbpr\_mt9n0\_camt054 section having two actions: A PUT /v2/run/cbpr\_mt9n0\_camt054 and a POST /v2/run/cbpr\_mt9n0\_camt054.

6. In the PUT action:

- a. Expand the PUT action and select the Try it out button.

- b. Leave the input and output sections empty.

- c. Under flow\_vars section, delete the entire content and replace with the commands to run the flow, for example:

```
{
 "INPUT_FILE": "C:/tests_dir/data/MT900.inp",
 "OUTPUT_DOC_RESULT_XML": "C:/tests_dir/data/env_camt_054_out.xml",
 "AUDIT_LOG": "C:/tests_dir/data/audit_log.json"
}
```

d. Click Execute blue bar for running the flow. When flow completes execution, 200 response code is shown in the Server Response section. You can see information about the process flow in the Response body.

7. Refresh the browser to delete the deployed package and get the Swagger Explore back to /tx-rest/openapi.json.

There will be a DELETE /v2/packages/{name} action.

8. Expand the DELETE /v2/packages/{name} action and select the Try it out.

9. In the Name field, enter the name you gave it to the created package.

10. Select the true option from the stop query drop down, select the blue Execute bar.

You can see a response of 204 with a timestamp, if it is successful.

---

## Run the example using the flow command server process (flowcmdserver)

To run the example, deploy locally or to ftp server and use the flow command server process (flowcmdserver).

1. Create a server definition where the runtime is installed, if you do not have a server local definition or ftp execution definition to deploy to, in Design Server.

2. In Design Server, deploy the created package to the server definition. For example, deploy to c:\ftpserver\deployment directory.

3. Execute the flow using the flow command server. In below example, command is for SWIFT message MT900.inp.

```
flowcmdserver.exe --dir c:\ftpserver\deployment\flows --flow cbpr_mt9n0_camt054.json
--var "INPUT_FILE=C:\ftpserver\deployment\cbpr_plus\translation\mt-mx\data\MT900.inp"
--var "OUTPUT_DOC_RESULT_XML=C:\ftpserver\deployment\cbpr_plus\translation\mt-mx\data\camt_054_001_08.xml"
--var "AUDIT_LOG=C:\ftpserver\deployment\cbpr_plus\translation\mt-mx\data\camt_054_001_08_log.json"
--audit "C:\ftpserver\deployment\cbpr_plus\translation\mt-mx\data\cbpr_mt9n0_camt054_adt.json"
-ad
```

4. The flow will report status similar to:

```
***Starting flow command server
Flow UUID: 85eabe2c-2302-4543-b347-9e18d65c6816
The flow audit file is: "C:\\ftpserver\\\\deployment\\\\cbpr_plus\\\\translation\\\\mt-
mx\\\\data\\\\cbpr_mt9n0_camt054_adt.json"
```

```
Flow completed successfully
Elapsed time: 3048ms
```

5. Examine the flow output result file cbpr\_mt9n0\_camt054\_adt.out and the audit file cbpr\_mt9n0\_camt054\_adt.json.

## CBPR+ MT to MX (MT192/MT292 to camt.056.001.08) Translation Example

This example demonstrates the CBPR+ MT to MX (MT192/MT292 to camt.056.001.08) message translation using flow server.

- **What the example contains**

Files included in this example are as follows:

- **How to run the example**

This mt-mx translation will use the sample files to demonstrate the generation of CBPR+ camt.056.001.08 XML message output from a SWIFT MT192 or MT292 message.

## What the example contains

Files included in this example are as follows:

- cbpr\_translation.zip
  - Files:
    - MT192.inp
    - MT292.inp
  - Schemas:
    - head.001.001.02\_camt\_056.xsd
    - camt.056.001.08.xsd
    - swift\_iso7775\_ccyy.mtt
    - swiftroute\_funds.mtt
    - infoset.mtt
  - Maps:
    - cbpr2840\_mtn92\_framework
    - cbpr2841\_mtn92\_camt056
    - cbpr1502\_mtmx\_setvarlog
  - Flow:
    - cbpr\_mtn92\_camt056
- cbprJnodesConfigIBM.tar.gz

## How to run the example

This mt-mx translation will use the sample files to demonstrate the generation of CBPR+ camt.056.001.08 XML message output from a SWIFT MT192 or MT292 message.

The cbprJnodesConfigIBM.tar.gz file is available either in IBM\_financialpaymentsplus\_vn.n.n.zip or in UIProjectImports directory.

Extract from cbprJnodesConfigIBM.tar.gz file.

- jnodes0.jar

The following jars needs to be copied from <TX\_install\_dir>/jars:

- jackson-core-n.n.n.jar
- jackson-annotations-n.n.n.jar
- jackson-databind-n.n.n.jar

Or visit <https://repo1.maven.org/maven2/com/fasterxml/jackson/core/> for download.

Note: Recommend using the latest version.

For the non Docker environments,

- Copy jars to <TX\_install\_dir>/extjar.

For TX V11.0.1 and up, native based Design Server installation,

Copy the following into the directory defined in config.yaml server.persistence.libs, by default, this is set to /opt/txlbs:

- jvcwrap.jar
- jvalccyy.jar

Restart the running application ./ITX stop and then ./ITX start.

For the Docker environments,

- docker cp jnodes0.jar <brand>-server:/opt/<company>/<brand>/libs/.
- Restart the design server, i.e., **docker restart <brand>-server**.

Note: For RHEL, Docker is not supported, Podman can be used as a substitute since it provides a command line interface similar to Docker. Below is an example of using podman:

- podman cp jnodes0.jar <brand>-server:/opt/<company>/<brand>/libs/.
- Restart the design server, i.e., **podman restart** <brand>-server.

This example may generate failure in the audit.log.json log file to report the translation failure due to the pre-conversion checks:

**If input file is SWIFT message other than MT192 or MT292 or not a SWIFT message.**

1. Import the cbpr\_translation.zip project into the Design Server.
  2. Open the cbpr\_translation project in Design Server and view the flow cbpr\_mtn92\_camt056.
    - a. The Flow Description points to all the maps to be executed during the flow process, which will be called from within some of the map nodes in the flow. The following is a list of maps invoked using RUN built-in function during the flow process:
      - @packagemap=cbpr\_plus/translation/mt-mx/maps/cbpr\_t9n\_mtn92\_camt056/cbpr2841\_mtn92\_camt056
      - @packagemap=cbpr\_plus/translation/mt-mx/maps/cbpr\_t9n\_mt\_setvarlog/cbpr1502\_mtmx\_setvarlog
    - b. It utilizes the following nodes:
      - i. Source Node
        - mt\_n92

This node identifies the input data to be translated in the flow. It uses the INPUT\_FILE variable to set the location of the data.
      - ii. Map Node
        - mt\_translate

Runs map cbpr2840\_mtn92\_framework which sets flow variables, checks pre-translation conditions and translates the input file (mt) into required output (XML).
      - iii. Target Node
        - camt.056

This node contains the resulting translation in XML format and creates the output as defined by the variable OUTPUT\_DOC\_RESULT\_XML.
      - iv. Log Node
        - audit\_log

This node creates a log file specified by the flow variable AUDIT\_LOG.
  - c. It uses the following flow variables:
    - INPUT\_FILE  
Default value for the flow variable INPUT\_FILE is ..//cbpr\_plus/translation/mt-mx/data/MT192.inp. This is the data file to be used for translation and can be customized. It is used in the Source node mt\_n92.
    - OUTPUT\_DOC\_RESULT\_XML  
Default value for the flow variable OUTPUT\_DOC\_RESULT\_XML is camt\_056\_out.xml. This is the location of the output file in mt format. It is used in the Target node camt.056. It can be customized.
    - AUDIT\_LOG  
Default value for the flow variable AUDIT\_LOG is audit.log.json. This is the location of the audit log. It is used in the Log node audit\_log. It can be customized.
    - CONFIG\_FILE  
Default value for the flow variable CONFIG\_FILE is ..//cbpr\_plus/translation/mt-mx/data/trx\_config.xml. This is the location of the file containing the configuration settings. It can be customized.
  3. Open the flow cbpr\_mtn92\_camt056 in the Design Server. It utilizes one source node, one map node, one log node, and one target node.
  4. In Design Server, create a package that contains the input files and one flow. The maps will automatically be included during deployment of the package onto the runtime server.
- **[Run the example using three different methods](#)**  
You can run the example using three different methods:

## Run the example using three different methods

You can run the example using three different methods:

1. Running the example from the Design Server user interface.
  2. Deploying the example to tx-rest and running it using the Swagger interface.
  3. Deploying locally or to ftp server and using the flow command server process (flowcmdserver).
- **[Run the example from the Design Server user interface](#)**  
Use the following steps to run the example from the Design Server user interface:
  - **[Run the example using the Swagger interface](#)**  
To run the example, deploy to tx-rest and run it using the Swagger interface.
  - **[Run the example using the flow command server process \(flowcmdserver\)](#)**  
To run the example, deploy locally or to ftp server and use the flow command server process (flowcmdserver).

## Run the example from the Design Server user interface

Use the following steps to run the example from the Design Server user interface:

1. In the Design Server, open cbpr\_mtn92\_camt056 flow and click on Run button.
2. Set toggle switch Run On Design Server and select SWIFT MT message input file MT192.inp/MT292.inp .  
Note: If a default path is assigned to the variable INPUT\_FILE, not selecting an input file will use the value defined for the variable.  
Flow will run and report with the green check box. The report lists execution of all nodes.

3. Right click and select View Data on each link to examine the data.
4. Right click on the node and select View Log to see the logs.

## Run the example using the Swagger interface

To run the example, deploy to tx-rest and run it using the Swagger interface.

1. Create a Web type server definition where the runtime tx-rest is installed, if you do not have a server definition to deploy in Design Server.
2. If not running, start tx-rest on the server: <DTX\_HOME>/restapi/tomcat/dtxtomcat start tx-rest.
3. Deploy the created package to the server definition in Design Server.
4. Bring up the tx-rest Swagger UI: <DTX\_HOME>/restapi/tomcat/dtxtomcat showdocs tx-rest.
5. In the Swagger Explore window, enter /tx-rest/v2/docs and click on the Explore button to view the deployed package.  
You should see a cbpr\_mtn92\_camt056 section having two actions: A PUT /v2/run/cbpr\_mtn92\_camt056 and a POST /v2/run/cbpr\_mtn92\_camt056.
6. In the PUT action:
  - a. Expand the PUT action and select the Try it out button.
  - b. Leave the input and output sections empty.
  - c. Under flow\_vars section, delete the entire content and replace with the commands to run the flow, for example:

```
{
 "INPUT_FILE": "C:/tests_dir/data/MT192.inp",
 "OUTPUT_DOC_RESULT_XML": "C:/tests_dir/data/camt_056_001_08.xml",
 "AUDIT_LOG": "C:/tests_dir/data/audit_log.json"
}
```
- d. Click Execute blue bar for running the flow. When flow completes execution, 200 response code is shown in the Server Response section. You can see information about the process flow in the Response body.
7. Refresh the browser to delete the deployed package and get the Swagger Explore back to /tx-rest/openapi.json. There will be a DELETE /v2/packages/{name} action.
8. Expand the DELETE /v2/packages/{name} action and select the Try it out.
9. In the Name field, enter the name you gave it to the created package.
10. Select the true option from the stop query drop down, select the blue Execute bar.  
You can see a response of 204 with a timestamp, if it is successful.

## Run the example using the flow command server process (flowcmdserver)

To run the example, deploy locally or to ftp server and use the flow command server process (flowcmdserver).

1. Create a server definition where the runtime is installed, if you do not have a server local definition or ftp execution definition to deploy to, in Design Server.
2. In Design Server, deploy the created package to the server definition. For example, deploy to c:\ftpserver\deployment directory.
3. Execute the flow using the flow command server. In below example, command is for SWIFT message MT192.inp.

```
flowcmdserver.exe --dir c:\ftpserver\deployment\flows --flow cbpr_mtn92_camt056.json
--var "INPUT_FILE=C:\ftpserver\deployment\cbpr_plus\translation\mt-mx\data\MT192.inp"
--var "AUDIT_LOG=C:\ftpserver\deployment\cbpr_plus\translation\mt-mx\data\camt_056_001_08_log.json"
--var "OUTPUT_DOC_RESULT_XML=C:\ftpserver\deployment\cbpr_plus\translation\mt-mx\data\camt_056_001_08.xml"
--audit "C:\ftpserver\deployment\cbpr_plus\translation\mt-mx\data\cbpr_mtn92_camt056_adt.json"
-ad
```

4. The flow will report status similar to:

```
***Starting flow command server

Flow UUID: 85eabe2c-2302-4543-b347-9e18d65c6816
The flow audit file is: "C:\\ftpserver\\deployment\\cbpr_plus\\\\translation\\\\mt-
mx\\\\data\\\\cbpr_mtn92_camt056_adt.json"
Flow completed successfully
Elapsed time: 3048ms
```

5. Examine the flow output result file cbpr\_mt192\_camt056.out and the audit file cbpr\_mtn92\_camt056\_adt.json.

## CBPR+ MT to MX (MT196/MT296 to camt.029.001.09) Translation Example

This example demonstrates the CBPR+ MT to MX (MT196/MT296 to camt.029.001.09) message translation using flow server.

- [What the example contains](#)  
Files included in this example are as follows:
- [How to run the example](#)  
This mt-mx translation will use the sample files to demonstrate the generation of CBPR+ camt.029.001.09 XML message output from a SWIFT MT196 or MT296 message.

## What the example contains

Files included in this example are as follows:

- cbpr\_translation.zip
  - Files:
    - MT196.inp
    - MT296.inp
    - MT196\_bad.inp
  - Schemas:
    - head.001.001.02\_camt\_029.xsd
    - camt.029.001.09.xsd
    - swift\_iso7775\_ccyy.mtt
    - swiftroute\_funds.mtt
    - infoset.mtt
  - Maps:
    - cbpr2820\_mtn96\_framework
    - cbpr2821\_mtn96\_camt029
    - cbpr1502\_mttx\_setvarlog
  - Flow:
    - cbpr\_mtn96\_camt029
- cbprJnodesConfigIBM.tar.gz

## How to run the example

This mt-mx translation will use the sample files to demonstrate the generation of CBPR+ camt.029.001.09 XML message output from a SWIFT MT196 or MT296 message.

The cbprJnodesConfigIBM.tar.gz file is available either in IBM\_financialpaymentsplus\_vn.n.n.n.zip or in UIProjectImports directory.

Extract from cbprJnodesConfigIBM.tar.gz file.

- jnodes0.jar

The following jars needs to be copied from <TX\_install\_dir>/jars:

- jackson-core-n.n.n.jar
- jackson-annotations-n.n.n.jar
- jackson-databind-n.n.n.jar

Or visit <https://repo1.maven.org/maven2/com/fasterxml/jackson/core/> for download.

Note: Recommend using the latest version.

For the non Docker environments,

- Copy jars to <TX\_install\_dir>/extjar.

For TX V11.0.1 and up, native based Design Server installation,

Copy the following into the directory defined in config.yaml server.persistence.libs, by default, this is set to /opt/tlibs:

- jvcwrap.jar
- jvalccyy.jar

Restart the running application ./ITX stop and then ./ITX start.

For the Docker environments,

- docker cp jnodes0.jar <brand>-server:/opt/<company>/<brand>/libs/.
- Restart the design server, i.e., **docker restart** <brand>-server.

Note: For RHEL, Docker is not supported, Podman can be used as a substitute since it provides a command line interface similar to Docker. Below is an example of using podman:

- podman cp jnodes0.jar <brand>-server:/opt/<company>/<brand>/libs/.
- Restart the design server, i.e., **podman restart** <brand>-server.

This example may generate failure in the audit.log.json file to report the translation failure due to the pre-conversion checks:

**If input file is SWIFT message other than MT196 or MT296 or not a SWIFT message.**

1. Import the cbpr\_translation.zip project into the Design Server.
2. Open the cbpr\_translation project in Design Server and view the flow cbpr\_mtn96\_camt029.
  - a. The Flow Description points to all the maps to be executed during the flow process, which will be called from within some of the map nodes in the flow. The following is a list of maps invoked using RUN built-in function during the flow process:
    - @packagemap=cbpr\_plus/translation/mt-mx/maps/cbpr\_t9n\_mtn96\_camt029/cbpr2821\_mtn96\_camt029
    - @packagemap=cbpr\_plus/translation/mt-mx/maps/cbpr\_t9n\_mt\_setvarlog/cbpr1502\_mttx\_setvarlog
  - b. It utilizes the following nodes:
    - i. Source Node
      - mt\_n96
 

This node identifies the input data to be translated in the flow. It uses the INPUT\_FILE variable to set the location of the data.
    - ii. Map Node
      - mt\_translate
 

Runs framework map cbpr2820\_mtn96\_framework, checks pre-translation conditions and translates the input file (mt) into required output (XML).
    - iii. Target Node

- camt.029  
This node contains the resulting translation in XML format and creates the output as defined by the variable OUTPUT\_DOC\_RESULT\_XML.

iv. Log Node

- audit\_log  
This node creates a log file specified by the flow variable AUDIT\_LOG.

c. It utilizes the following flow variables:

- INPUT\_FILE  
Default value for the flow variable INPUT\_FILE is ..\cbpr\_plus\translation\mt-mx\data\MT196.inp. This is the data file to be used for translation and can be customized. It is used in the Source node mt\_n96.
- OUTPUT\_DOC\_RESULT\_XML  
Default value for the flow variable OUTPUT\_DOC\_RESULT\_XML is camt\_029\_out.xml. This is the location of the output file in mt format. It is used in the Target node camt.029. It can be customized.
- AUDIT\_LOG  
Default value for the flow variable AUDIT\_LOG is audit.log.json. This is the location of the audit log. It is used in the Log node audit\_log. It can be customized.
- CONFIG\_FILE  
Default value for the flow variable CONFIG\_FILE is ..\cbpr\_plus\translation\mx-mt\data\trx\_config.xml. This is the location of the file containing the configuration settings. It can be customized.

3. Open the flow cbpr\_mtn96\_camt029 in the Design Server. It utilizes one source node, one map node, one log node, and one target node.

4. In Design Server, create a package that contains the input files and one flow. The maps will automatically be included during deployment of the package onto the runtime server.

**• [Run the example using three different methods](#)**

You can run the example using three different methods:

## Run the example using three different methods

You can run the example using three different methods:

1. Running the example from the Design Server user interface.
2. Deploying the example to tx-rest and running it using the Swagger interface.
3. Deploying locally or to ftp server and using the flow command server process (flowcmdserver).

**• [Run the example from the Design Server user interface](#)**

Use the following steps to run the example from the Design Server user interface:

**• [Run the example using the Swagger interface](#)**

To run the example, deploy to tx-rest and run it using the Swagger interface.

**• [Run the example using the flow command server process \(flowcmdserver\)](#)**

To run the example, deploy locally or to ftp server and use the flow command server process (flowcmdserver).

## Run the example from the Design Server user interface

Use the following steps to run the example from the Design Server user interface:

1. In the Design Server, open cbpr\_mtn96\_camt029 flow and click on Run button.
2. Set toggle switch Run On Design Server and select SWIFT MT message input file MT196.inp/MT296.inp .  
Note: If a default path is assigned to the variable INPUT\_FILE, not selecting an input file will use the value defined for the variable.  
Flow will run and report with the green check box. The report lists execution of all nodes.
3. Right click and select View Data on each link to examine the data.
4. Right click on the node and select View Log to see the logs.

## Run the example using the Swagger interface

To run the example, deploy to tx-rest and run it using the Swagger interface.

1. Create a Web type server definition where the runtime tx-rest is installed, if you do not have a server definition to deploy in Design Server.
2. If not running, start tx-rest on the server: <DTX\_HOME>/restapi/tomcat/dtxtomcat start tx-rest.
3. Deploy the created package to the server definition in Design Server.
4. Bring up the tx-rest Swagger UI: <DTX\_HOME>/restapi/tomcat/dtxtomcat showdocs tx-rest.
5. In the Swagger Explore window, enter /tx-rest/v2/docs and click on the Explore button to view the deployed package.  
You should see a cbpr\_mtn96\_camt029 section having two actions: A PUT /v2/run/cbpr\_mtn96\_camt029 and a POST /v2/run/cbpr\_mtn96\_camt029.
6. In the PUT action:
  - a. Expand the PUT action and select the Try it out button.
  - b. Leave the input and output sections empty.
  - c. Under flow\_vars section, delete the entire content replace with the commands to run the flow, for example:

```
{
 "INPUT_FILE": "C:/tests_dir/data/MT196.inp",
 "OUTPUT_DOC_RESULT_XML": "C:/tests_dir/data/camt_029_001_09.xml",
```

```
"AUDIT_LOG": "C:/tests_dir/data/audit_log.json"
}
```

- d. Click Execute blue bar for running the flow. When flow completes execution, 200 response code is shown in the Server Response section. You can see information about the process flow in the Response body.
7. Refresh the browser to delete the deployed package and get the Swagger Explore back to /tx-rest/openapi.json.  
There will be a DELETE /v2/packages/{name} action.
8. Expand the DELETE /v2/packages/{name} action and select the Try it out.
9. In the Name field, enter the name you gave it to the created package.
10. Select the true option from the stop query drop down, select the blue Execute bar.  
You can see a response of 204 with a timestamp, if it is successful.

---

## Run the example using the flow command server process (flowcmdserver)

To run the example, deploy locally or to ftp server and use the flow command server process (flowcmdserver).

1. Create a server definition where the runtime is installed, if you do not have a server local definition or ftp execution definition to deploy to, in Design Server.
2. In Design Server, deploy the created package to the server definition. For example, deploy to c:\ftpserver\deployment directory.
3. Execute the flow using the flow command server. In below example, command is for SWIFT message MT196.inp.

```
flowcmdserver.exe --dir c:\ftpserver\deployment\flows --flow cbpr_mtn96_camt029.json
--var "INPUT_FILE=C:\ftpserver\deployment\cbpr_plus\translation\mt-mx\data\MT196.inp"
--var "AUDIT_LOG=C:\ftpserver\deployment\cbpr_plus\translation\mt-mx\data\camt_029_001_09_log.json"
--var "OUTPUT_DOC_RESULT_XML=C:\ftpserver\deployment\cbpr_plus\translation\mt-mx\data\camt_029_001_09.xml"
--audit "C:\ftpserver\deployment\cbpr_plus\translation\mt-mx\data\cbpr_mtn96_camt029_adt.json"
-ad
```

4. The flow will report status similar to:

```
***Starting flow command server

Flow UUID: 85eabe2c-2302-4543-b347-9e18d65c6816
The flow audit file is: "C:\\\\ftpserver\\\\deployment\\\\cbpr_plus\\\\translation\\\\mt-
mx\\\\data\\\\cbpr_mtn96_camt029_adt.json"
Flow completed successfully
Elapsed time: 3048ms
```

5. Examine the flow output result file cbpr\_mt196\_camt029.out and the audit file cbpr\_mtn96\_camt029\_adt.json.

---

## CBPR+ MT to MX (MT103 to pacs.008.001.08 or MT103 RETN to pacs.004.001.09) Translation Example

This example demonstrates the CBPR+ MT to MX (MT103 to pacs.008.001.08 or MT103 RETN to pacs.004.001.09) message translation using flow server.

- [What the example contains](#)  
Files included in this example are as follows:
- [How to run the example](#)  
This mt-mx translation will use the sample files to demonstrate the generation of CBPR+ pacs.008.001.08 / pacs.004.001.09 XML message output from a SWIFT MT103 / MT103 RETN message.

---

## What the example contains

Files included in this example are as follows:

- cbpr\_translation.zip
  - Files:
    - MT103.inp
    - MT103\_RETN.inp
  - Schemas:
    - head.001.001.02\_pacs\_008.xsd
    - pacs.008.001.08.xsd
    - head.001.001.02\_pacs\_004.xsd
    - pacs.004.001.09.xsd
    - swift\_iso7775\_ccyy.mtt
    - swiftroute\_funds.mtt
    - infoset.mtt
  - Maps:
    - cbpr2800\_mt103\_framework
    - cbpr2801\_mt103\_pacs008
    - cbpr2799\_mt103\_pacs004
    - cbpr2805\_mt103\_pacs008\_sgo
    - cbpr1502\_mt103\_setvarlog
  - Flow:
    - cbpr\_mt103\_pacs008
- cbprJnodesConfigIBM.tar.gz

## How to run the example

This mt-mx translation will use the sample files to demonstrate the generation of CBPR+ pacs.008.001.08 / pacs.004.001.09 XML message output from a SWIFT MT103 / MT103 RETN message.

The cbprJnodesConfigIBM.tar.gz file is available either in IBM\_financialpaymentsplus\_vn.n.n.n.zip or in UIProjectImports directory.

Extract from cbprJnodesConfigIBM.tar.gz file.

- jnodes0.jar

The following jars needs to be copied from <TX\_install\_dir>/jars:

- jackson-core-n.n.n.jar
- jackson-annotations-n.n.n.jar
- jackson-databind-n.n.n.jar

Or visit <https://repo1.maven.org/maven2/com/fasterxml/jackson/core/> for download.

Note: Recommend using the latest version.

For the non Docker environments,

- Copy jars to <TX\_install\_dir>/extjar.

For TX V11.0.1 and up, native based Design Server installation,

Copy the following into the directory defined in config.yaml server.persistence.libs, by default, this is set to /opt/tlxlibs:

- jvcwrap.jar
- jvalccyy.jar

Restart the running application ./ITX stop and then ./ITX start.

For the Docker environments,

- docker cp jnodes0.jar <brand>-server:/opt/<company>/<brand>/libs/.
- Restart the design server, i.e., **docker restart <brand>-server**.

Note: For RHEL, Docker is not supported, Podman can be used as a substitute since it provides a command line interface similar to Docker. Below is an example of using podman:

- podman cp jnodes0.jar <brand>-server:/opt/<company>/<brand>/libs/.
- Restart the design server, i.e., **podman restart <brand>-server**.

This example may generate failure in the audit.log.json log file to report the translation failure if the input file is a SWIFT message other than MT103 or not a SWIFT message.

1. Import the cbpr\_translation.zip project into the Design Server.

2. Open the cbpr\_translation project in Design Server and view the flow cbpr\_mt103\_pacs00n.

a. The Flow Description points to all the maps to be executed during the flow process, which will be called from within some of the map nodes in the flow. The following is a list of maps invoked using RUN built-in function during the flow process:

- @packagemap=cbpr\_plus/translation/mt-mx/maps/cbpr\_t9n\_mt103\_pacs008/cbpr2801\_mt103\_pacs008
- @packagemap=cbpr\_plus/translation/mt-mx/maps/cbpr\_t9n\_mt103\_pacs004/cbpr2799\_mt103\_pacs004
- @packagemap=cbpr\_plus/translation/mt-mx/maps/cbpr\_t9n\_mt103\_pacs008\_sgo/cbpr2805\_mt103\_pacs008\_sgo
- @packagemap=cbpr\_plus/translation/mt-mx/maps/cbpr\_t9n\_mt\_setvarlog/cbpr1502\_mttx\_setvarlog

b. It utilizes the following nodes:

i. Source Node

- mt\_103

This node identifies the input data to be translated in the flow. It uses the INPUT\_FILE variable to set the location of the data.

ii. Map Node

- mt\_translate

Runs map cbpr2800\_mt103\_framework which sets flow variables, checks pre-translation conditions and translates the input file (mt) into required output (XML).

iii. Target Node

- pacs.00n

This node contains the resulting translation in XML format and creates the pacs.008 or pacs.004 output as defined by variable OUTPUT\_DOC\_RESULT\_XML.

iv. Log Node

- audit\_log

This node creates a log file specified by the flow variable AUDIT\_LOG.

c. It utilizes the following flow variables:

- INPUT\_FILE

Default value for the flow variable INPUT\_FILE is ..//cbpr\_plus/translation/mt-mx/data/MT103\_RETN.inp. This is the data file to be used for translation and can be customized. It is used in the Source node mt\_103.

- OUTPUT\_DOC\_RESULT\_XML

Default value for the flow variable OUTPUT\_DOC\_RESULT\_XML is pacs.00n.xml. This is the location of the output file in mt format. It is used in the Target node pacs.008 or pacs.004. It can be customized.

- AUDIT\_LOG

Default value for the flow variable AUDIT\_LOG is audit.log.json. This is the location of the audit log. It is used in the Log node audit\_log. It can be customized.

- **CONFIG\_FILE**

Default value for the flow variable CONFIG\_FILE is ..\cbpr\_plus\translation\mt-mx\data\trx\_config.xml. This is the location of the file containing the configuration settings. It can be customized.

3. Open the flow cbpr\_mt103\_pacs00n in the Design Server. It utilizes one source node, one map node, one log node, and one target node.
4. In Design Server, create a package that contains the input files and one flow. The maps will automatically be included during deployment of the package onto the runtime server.

- **Run the example using three different methods**

You can run the example using three different methods:

You can run the example using three different methods:

1. Running the example from the Design Server user interface.
2. Deploying the example to tx-rest and running it using the Swagger interface.
3. Deploying locally or to ftp server and using the flow command server process (flowcmdserver).

- **Run the example from the Design Server user interface**

Use the following steps to run the example from the Design Server user interface:

- **Run the example using the Swagger interface**

To run the example, deploy to tx-rest and run it using the Swagger interface.

- **Run the example using the flow command server process (flowcmdserver)**

To run the example, deploy locally or to ftp server and use the flow command server process (flowcmdserver).

## Run the example from the Design Server user interface

Use the following steps to run the example from the Design Server user interface:

1. In the Design Server, open cbpr\_mt103\_pacs00n flow and click on Run button.
2. Set toggle switch Run On Design Server and select SWIFT MT message input file MT103.inp.  
Note: If a default path is assigned to the variable INPUT\_FILE, not selecting an input file will use the value defined for the variable.  
Flow will run and report with the green check box. The report lists execution of all nodes.
3. Right click and select View Data on each link to examine the data.
4. Right click on the node and select View Log to see the logs.

## Run the example using the Swagger interface

To run the example, deploy to tx-rest and run it using the Swagger interface.

1. Create a Web type server definition where the runtime tx-rest is installed, if you do not have a server definition to deploy in Design Server.
2. If not running, start tx-rest on the server: <DTX\_HOME>/restapi/tomcat/dtxtomcat start tx-rest.
3. Deploy the created package to the server definition in Design Server.
4. Bring up the tx-rest Swagger UI: <DTX\_HOME>/restapi/tomcat/dtxtomcat showdocs tx-rest.
5. In the Swagger Explore window, enter /tx-rest/v2/docs and click on the Explore button to view the deployed package.  
You should see cbpr\_mt103\_pacs00n section having two actions: A PUT /v2/run/cbpr\_mt103\_pacs00n and a POST /v2/run/cbpr\_mt103\_pacs00n.
6. In the PUT action:
  - a. Expand the PUT action and select the Try it out button.
  - b. Leave the input and output sections empty.
  - c. Under flow\_vars section, delete the entire content and replace with the commands to run the flow, for example:

```
{
 "INPUT_FILE": "C:/tests_dir/data/MT103.inp",
 "OUTPUT_DOC_RESULT_XML": "C:/tests_dir/data/pacs_008_001_08.xml",
 "AUDIT_LOG": "C:/tests_dir/data/audit_log.json"
}
```

- d. Click Execute blue bar for running the flow. When flow completes execution, 200 response code is shown in the Server Response section. You can see information about the process flow in the Response body.
7. Refresh the browser to delete the deployed package and get the Swagger Explore back to /tx-rest/openapi.json.  
There will be a DELETE /v2/packages/{name} action.
8. Expand the DELETE /v2/packages/{name} action and select the Try it out.
9. In the Name field, enter the name you gave it to the created package.
10. Select the true option from the stop query drop down, select the blue Execute bar.  
You can see a response of 204 with a timestamp, if it is successful.

## Run the example using the flow command server process (flowcmdserver)

To run the example, deploy locally or to ftp server and use the flow command server process (flowcmdserver).

1. Create a server definition where the runtime is installed, if you do not have a server local definition or ftp execution definition to deploy to, in Design Server.
2. In Design Server, deploy the created package to the server definition. For example, deploy to c:\ftpserver\deployment directory.
3. Execute the flow using the flow command server. In below example, command is for SWIFT message MT103.inp.

```
flowcmdserver.exe --dir c:\ftpserver\deployment\flows --flow cbpr_mt103_pacs008.json
--var "INPUT_FILE=C:\ftpserver\deployment\cbpr_plus\translation\mt-mx\data\MT103.inp"
--var "OUTPUT_DOC_RESULT_XML=C:\ftpserver\deployment\cbpr_plus\translation\mt-mx\data\pacs_008_001_08.xml"
--var "AUDIT_LOG=C:\ftpserver\deployment\cbpr_plus\translation\mt-mx\data\pacs_008_001_08_log.json"
--audit "C:\ftpserver\deployment\cbpr_plus\translation\mt-mx\data\cbpr_mt103_pacs008_adt.json"
-ad
```

4. The flow will report status similar to:

```
***Starting flow command server

Flow UUID: 85eabe2c-2302-4543-b347-9e18d65c6816
The flow audit file is: "C:\\\\ftpserver\\\\deployment\\\\cbpr_plus\\\\translation\\\\mt-
mx\\\\data\\\\cbpr_mt103_pacs008_adt.json"
Flow completed successfully
Elapsed time: 3048ms
```

5. Examine the flow output result file cbpr\_mt103\_pacs008\_adt.out and the audit file cbpr\_mt103\_pacs008\_adt.json.

## CBPR+ MT to MX (MT103 SWIFTGo to pacs.008.001.08) Translation Example

This example demonstrates the CBPR+ MT to MX (MT103 SWIFTGo to pacs.008.001.08) message translation using flow server.

- **What the example contains**

Files included in this example are as follows:

- **How to run the example**

This mt-mx translation will use the sample files to demonstrate the generation of CBPR+ pacs.008.001.08 XML message output from a MT103 SWIFTGo message.

## What the example contains

Files included in this example are as follows:

- cbpr\_translation.zip
  - Files:
    - MT103\_SGO.inp
  - Schemas:
    - head.001.001.02\_pacs\_008.xsd
    - pacs.008.001.08.xsd
    - swift\_iso7775\_ccyy.mtt
    - swiftroute\_funds.mtt
    - infoset.mtt
  - Maps:
    - cbpr2808\_mt103sgo\_framework
    - cbpr2801\_mt103\_pacs008
    - cbpr2799\_mt103\_pacs004
    - cbpr2805\_mt103\_pacs008\_sgo
    - cbpr1502\_mtnx\_setvarlog
  - Flow:
    - cbpr\_mt103sgo\_pacs008
- cbprJnodesConfigIBM.tar.gz

## How to run the example

This mt-mx translation will use the sample files to demonstrate the generation of CBPR+ pacs.008.001.08 XML message output from a MT103 SWIFTGo message.

The cbprJnodesConfigIBM.tar.gz file is available either in IBM\_financialpaymentsplus\_vn.n.n.n.zip or in UIProjectImports directory.

Extract from cbprJnodesConfigIBM.tar.gz file.

- jnodes0.jar

The following jars needs to be copied from <TX\_install\_dir>/jars:

- jackson-core-n.n.n.jar
- jackson-annotations-n.n.n.jar
- jackson-databind-n.n.n.jar

Or visit <https://repo1.maven.org/maven2/com/fasterxml/jackson/core/> for download.

Note: Recommend using the latest version.

For the non Docker environments,

- Copy jars to <TX\_install\_dir>/extjar.

For TX V11.0.1 and up, native based Design Server installation,

Copy the following into the directory defined in config.yaml server.persistence.libs, by default, this is set to /opt/txlibs:

- jvcwrap.jar
- jvalccyy.jar

Restart the running application ./ITX stop and then ./ITX start.

For the Docker environments,

- docker cp jnodes0.jar <brand>-server:/opt/<company>/<brand>/libs/.
- Restart the design server, i.e., **docker restart** <brand>-server.

Note: For RHEL, Docker is not supported, Podman can be used as a substitute since it provides a command line interface similar to Docker. Below is an example of using podman:

- podman cp jnodes0.jar <brand>-server:/opt/<company>/<brand>/libs/.
- Restart the design server, i.e., **podman restart**<brand>-server.

This example may generate failure in the audit.log.json log file to report the translation failure if the input file is a SWIFT message other than MT103 or not a SWIFT message or Field 72, Line 1 starts with "/REJT/" OR "/RETN/".

1. Import the cbpr\_translation.zip project into the Design Server.

2. Open the cbpr\_translation project in Design Server and view the flow cbpr\_mt103sgo\_pacs008.

a. The Flow Description points to all the maps to be executed during the flow process, which will be called from within some of the map nodes in the flow. The following is a list of maps invoked using RUN built-in function during the flow process:

- @packagemap=cbpr\_plus/translation/mt-mx/maps/cbpr\_t9n\_mt103\_pacs008/cbpr2801\_mt103\_pacs008
- @packagemap=cbpr\_plus/translation/mt-mx/maps/cbpr\_t9n\_mt103\_pacs004/cbpr2799\_mt103\_pacs004
- @packagemap=cbpr\_plus/translation/mt-mx/maps/cbpr\_t9n\_mt103\_pacs008\_sgo/cbpr2805\_mt103\_pacs008\_sgo
- @packagemap=cbpr\_plus/translation/mt-mx/maps/cbpr\_t9n\_mt\_setvarlog/cbpr1502\_mtmx\_setvarlog

b. It utilizes the following nodes:

i. Source Node

- mt\_103sgo

This node identifies the input data to be translated in the flow. It uses the INPUT\_FILE variable to set the location of the data.

ii. Map Node

- mt\_translate

Runs map cbpr2808\_mt103sgo\_framework which sets flow variables, checks pre-translation conditions and translates the input file (mt) into required output (XML).

iii. Target Node

- pacs.008

This node contains the resulting translation in XML format and creates the output as defined by the variable OUTPUT\_DOC\_RESULT\_XML.

iv. Log Node

- audit\_log

This node creates a log file specified by the flow variable AUDIT\_LOG.

c. It utilizes the following flow variables:

- INPUT\_FILE

Default value for the flow variable INPUT\_FILE is ..//cbpr\_plus/translation/mt-mx/data/MT103\_SGO.inp. This is the data file to be used for translation and can be customized. It is used in the Source node mt\_103sgo.

- OUTPUT\_DOC\_RESULT\_XML

Default value for the flow variable OUTPUT\_DOC\_RESULT\_XML is pacs\_008\_sgo.xml. This is the location of the output file in mt format. It is used in the Target node pacs.008. It can be customized.

- AUDIT\_LOG

Default value for the flow variable AUDIT\_LOG is audit.log.json. This is the location of the audit log. It is used in the Log node audit\_log. It can be customized.

- CONFIG\_FILE

Default value for the flow variable CONFIG\_FILE is ..//cbpr\_plus/translation/mt-mx/data/trx\_config.xml. This is the location of the file containing the configuration settings. It can be customized.

3. Open the flow cbpr\_mt103sgo\_pacs008 in the Design Server. It utilizes one source node, one map node, one log node, and one target node.

4. In Design Server, create a package that contains the input files and one flow. The maps will automatically be included during deployment of the package onto the runtime server.

- **[Run the example using three different methods](#)**

You can run the example using three different methods:

## Run the example using three different methods

You can run the example using three different methods:

1. Running the example from the Design Server user interface.
2. Deploying the example to tx-rest and running it using the Swagger interface.
3. Deploying locally or to ftp server and using the flow command server process (flowcmdserver).

- [Run the example from the Design Server user interface](#)

Use the following steps to run the example from the Design Server user interface:

- [Run the example using the Swagger interface](#)

To run the example, deploy to tx-rest and run it using the Swagger interface.

- [Run the example using the flow command server process \(flowcmdserver\)](#)

To run the example, deploy locally or to ftp server and use the flow command server process (flowcmdserver).

---

## Run the example from the Design Server user interface

Use the following steps to run the example from the Design Server user interface:

1. In the Design Server, open cbpr\_mt103sgo\_pacs008 flow and click on Run button.
2. Set toggle switch Run On Design Server and select SWIFT MT message input file MT103\_SGO.inp.

Note: If a default path is assigned to the variable INPUT\_FILE, not selecting an input file will use the value defined for the variable.

Flow will run and report with the green check box. The report lists execution of all nodes.

3. Right click and select View Data on each link to examine the data.
4. Right click on the node and select View Log to see the logs.

---

## Run the example using the Swagger interface

To run the example, deploy to tx-rest and run it using the Swagger interface.

1. Create a Web type server definition where the runtime tx-rest is installed, if you do not have a server definition to deploy in Design Server.
2. If not running, start tx-rest on the server: <DTX\_HOME>/restapi/tomcat/dtxtomcat start tx-rest.

3. Deploy the created package to the server definition in Design Server.

4. Bring up the tx-rest Swagger UI: <DTX\_HOME>/restapi/tomcat/dtxtomcat showdocs tx-rest.

5. In the Swagger Explore window, enter /tx-rest/v2/docs and click on the Explore button to view the deployed package.

You should see a cbpr\_mt103sgo\_pacs008 section having two actions: A PUT /v2/run/cbpr\_mt103sgo\_pacs008 and a POST /v2/run/cbpr\_mt103sgo\_pacs008.

6. In the PUT action:

- a. Expand the PUT action and select the Try it out button.

- b. Leave the input and output sections empty.

- c. Under flow\_vars section, delete the entire content and replace with the commands to run the flow, for example:

```
{
 "INPUT_FILE": "C:/tests_dir/data/MT103_SGO.inp",
 "OUTPUT_DOC_RESULT_XML": "C:/tests_dir/data/pacs_008_001_08.xml",
 "AUDIT_LOG": "C:/tests_dir/data/audit_log.json"
}
```

- d. Click Execute blue bar for running the flow. When flow completes execution, 200 response code is shown in the Server Response section. You can see information about the process flow in the Response body.

7. Refresh the browser to delete the deployed package and get the Swagger Explore back to /tx-rest/openapi.json.

There will be a DELETE /v2/packages/{name} action.

8. Expand the DELETE /v2/packages/{name} action and select the Try it out.

9. In the Name field, enter the name you gave it to the created package.

10. Select the true option from the stop query drop down, select the blue Execute bar.

You can see a response of 204 with a timestamp, if it is successful.

---

## Run the example using the flow command server process (flowcmdserver)

To run the example, deploy locally or to ftp server and use the flow command server process (flowcmdserver).

1. Create a server definition where the runtime is installed, if you do not have a server local definition or ftp execution definition to deploy to, in Design Server.

2. In Design Server, deploy the created package to the server definition. For example, deploy to c:\ftpserver\deployment directory.

3. Execute the flow using the flow command server. In below example, command is for SWIFT message MT103\_SGO.inp.

```
flowcmdserver.exe --dir c:\ftpserver\deployment\flows --flow cbpr_mt103sgo_pacs008.json
--var "INPUT_FILE=C:\ftpserver\deployment\cbpr_plus\translation\mt-mx\data\MT103_SGO.inp"
--var "OUTPUT_DOC_RESULT_XML=C:\ftpserver\deployment\cbpr_plus\translation\mt-mx\data\pacs_008_001_08.xml"
--var "AUDIT_LOG=C:\ftpserver\deployment\cbpr_plus\translation\mt-mx\data\pacs_008_001_08_log.json"
--audit "C:\ftpserver\deployment\cbpr_plus\translation\mt-mx\data\cbpr_mt103sgo_pacs008_adt.json"
-ad
```

4. The flow will report status similar to:

```
***Starting flow command server

Flow UUID: 85eabec2-2302-4543-b347-9e18d65c6816
The flow audit file is: "C:\ftpserver\deployment\cbpr_plus\translation\mt-mx\data\cbpr_mt103sgo_pacs008_adt.json"
Flow completed successfully
Elapsed time: 3048ms
```

5. Examine the flow output result file cbpr\_mt103sgo\_pacs008\_adt.out and the audit file cbpr\_mt103sgo\_pacs008\_adt.json.

---

## CBPR+ MT to MX (MT202 ADV/COR/COV to pacs.009.001.08 OR MT202 RETN to pacs.004.001.09) Translation Example

This example demonstrates the CBPR+ MT to MX (MT202 ADV/COR/COV to pacs.009.001.08 OR MT202 RETN to pacs.004.001.09) message translation using flow server.

- [What the example contains](#)

Files included in this example are as follows:

- [How to run the example](#)
- This mt-mx translation will use the sample files to demonstrate the generation of SWIFT CBPR+ pacs.009.001.08 output from a MT202 message or OR CBPR+ pacs.004.001.09 output from a MT202 RETN message.

---

## What the example contains

Files included in this example are as follows:

- cbpr\_translation.zip
  - Files:
    - MT202\_COR.inp
    - MT202\_COR\_bad.inp
    - MT202\_RTN.inp
    - trx\_config.xml
  - Schemas:
    - head.001.001.02\_pacs\_004.xsd
    - head.001.001.02\_pacs\_009.xsd
    - pacs.004.001.09.xsd
    - pacs.009.001.08.xsd
    - infoset.mtt
    - swift\_iso7775\_ccyy.mtt
    - swiftroute\_funds.mtt
    - trx\_config.xsd
  - Maps:
    - cbpr2810\_mt202\_framework
    - cbpr2814\_mt202\_pacs009\_adv
    - cbpr2815\_mt202\_pacs009\_cov
    - cbpr2813\_mt202\_pacs009
    - cbpr2824\_mt202RTN\_pacs004
    - cbpr1502\_mtmx\_setvarlog
  - Flow:
    - cbpr\_mt202\_pacs00n
- cbprJnodesConfigIBM.tar.gz

---

## How to run the example

This mt-mx translation will use the sample files to demonstrate the generation of SWIFT CBPR+ pacs.009.001.08 output from a MT202 message or OR CBPR+ pacs.004.001.09 output from a MT202 RETN message.

The cbprJnodesConfigIBM.tar.gz file is available either in IBM\_financialpaymentsplus\_vn.n.n.n.zip or in UIProjectImports directory.

Extract from cbprJnodesConfigIBM.tar.gz file.

- jnodes0.jar

The following jars needs to be copied from <TX\_install\_dir>/jars:

- jackson-core-n.n.n.jar
- jackson-annotations-n.n.n.jar
- jackson-databind-n.n.n.jar

Or visit <https://repo1.maven.org/maven2/com/fasterxml/jackson/core/> for download.

Note: Recommend using the latest version.

For the non Docker environments,

- Copy jars to <TX\_install\_dir>/extjar.

For TX V11.0.1 and up, native based Design Server installation,

Copy the following into the directory defined in config.yaml server.persistence.libs, by default, this is set to /opt/tlxlibs:

- jvcwrap.jar
- jvalccyy.jar

Restart the running application ./ITX stop and then ./ITX start.

For the Docker environments,

- docker cp jnodes0.jar <brand>-server:/opt/<company>/<brand>/libs/.
- Restart the design server, i.e., **docker restart** <brand>-server.

Note: For RHEL, Docker is not supported, Podman can be used as a substitute since it provides a command line interface similar to Docker. Below is an example of using podman:

- podman cp jnodes0.jar <brand>-server:/opt/<company>/<brand>/libs/.
- Restart the design server, i.e., **podman restart** <brand>-server.

This example may generate failure in the audit.log.json log file to report the translation failure due to the pre-conversion checks:

```
Field 72, Line[1] should not start with "/REJT/".
If Field 72, Line[1] starts with "/RETN/", <Tag121><EndToEndTxnRef> should not be absent.
```

1. Import the cbpr\_translation.zip project into the Design Server.
2. Open the cbpr\_translation project in Design Server and view the flow cbpr\_mt202\_pacs00n.
  - a. The Flow Description points to all the maps to be executed during the flow process, which will be called from within some of the map nodes in the flow. The following is a list of maps invoked using RUN built-in function during the flow process:
    - @packagemap=cbpr\_plus/translation/mt-mx/maps/cbpr\_t9n\_mt202\_pacs009\_adv/cbpr2814\_mt202\_pacs009\_adv
    - @packagemap=cbpr\_plus/translation/mt-mx/maps/cbpr\_t9n\_mt20n\_pacs009\_cov/cbpr2815\_mt202\_pacs009\_cov
    - @packagemap=cbpr\_plus/translation/mt-mx/maps/cbpr\_t9n\_mt20nRTN\_pacs004/cbpr2824\_mt202RTN\_pacs004
    - @packagemap=cbpr\_plus/translation/mt-mx/maps/cbpr\_t9n\_mt20n\_pacs009/cbpr2813\_mt202\_pacs009
    - @packagemap=cbpr\_plus/translation/mt-mx/maps/cbpr\_t9n\_mt\_setvarlog/cbpr1502\_mtmx\_setvarlog
  - b. It utilizes the following nodes:
    - i. Source Node
      - mt\_202
 This node identifies the input data to be translated in the flow. It uses the INPUT\_FILE variable to set the location of the data.
    - ii. Map Node
      - mt\_translate
 Runs map cbpr2810\_mt202\_framework which set flow variables, checks pre-translation conditions and translates the input file (mt) into required output (XML).
    - iii. Target Node
      - pacs.00n
 This node contains the resulting translation in XML format and creates the pacs.004 or pacs.009 output as defined by variable OUTPUT\_DOC\_RESULT\_XML.
    - iv. Log Node
      - audit\_log
 This node creates a log file specified by the flow variable AUDIT\_LOG.
- c. It uses the following flow variables:
  - INPUT\_FILE
 Default value for the flow variable INPUT\_FILE is ..//cbpr\_plus/translation/mt-mx/data/MT202\_COR.inp. This is the data file to be used for translation and can be customized. It is used in the Source node mt\_202.
  - OUTPUT\_DOC\_RESULT\_XML
 Default value for the flow variable OUTPUT\_DOC\_RESULT\_XML is pacs.00n.xml. This is the location of the output file in mt format. It is used in the Target node pacs.00n. It can be customized.
  - AUDIT\_LOG
 Default value for the flow variable AUDIT\_LOG is audit.log.json. This is the location of the audit log. It is used in the Log node audit\_log. It can be customized.
  - CONFIG\_FILE
 Default value for the flow variable CONFIG\_FILE is ..//cbpr\_plus/translation/mt-mx/data/trx\_config.xml. This is the location of the file containing the configuration settings. It can be customized.
3. Open the flow cbpr\_mt202\_pacs00n in the Design Server. It utilizes one source node, one map node, one log node, and one target node.
4. In Design Server, create a package that contains the input files and one flow. The maps will automatically be included during deployment of the package onto the runtime server.
  - [\*\*Run the example using three different methods\*\*](#)

You can run the example using three different methods:

## Run the example using three different methods

You can run the example using three different methods:

1. Running the example from the Design Server user interface.
  2. Deploying the example to tx-rest and running it using the swagger interface.
  3. Deploying locally or to ftp server and using the flow command server process (flowcmdserver).
- [\*\*Run the example from the Design Server user interface\*\*](#)

Use the following steps to run the example from the Design Server user interface:
  - [\*\*Run the example using the Swagger interface\*\*](#)

To run the example, deploy to tx-rest and run it using the Swagger interface.
  - [\*\*Run the example using the flow command server process \(flowcmdserver\)\*\*](#)

To run the example, deploy locally or to ftp server and use the flow command server process (flowcmdserver).

## Run the example from the Design Server user interface

Use the following steps to run the example from the Design Server user interface:

1. In the Design Server, open cbpr\_mt202\_pacs00n flow and click on Run button.
2. Set toggle switch Run On Design Server and select flow input file MT202\_COR.inp or MT202\_COR\_bad.inp.  
Note: If a default path is assigned to the variable INPUT\_FILE, not selecting an input file will use the value defined for the variable.  
Flow will run and report with the green check box. The report lists execution of all nodes.
3. Right click and select View Data on each link to examine the data.
4. Right click on the node and select View Log to see the logs.

## Run the example using the Swagger interface

To run the example, deploy to tx-rest and run it using the Swagger interface.

1. Create a Web type server definition where the runtime tx-rest is installed, if you do not have a server definition to deploy in Design Server.
2. If not running, start tx-rest on the server: <DTX\_HOME>/restapi/tomcat/dtxtomcat start tx-rest.
3. Deploy the created package to the server definition in Design Server.
4. Bring up the tx-rest Swagger UI: <DTX\_HOME>/restapi/tomcat/dtxtomcat showdocs tx-rest.
5. In the Swagger Explore window, enter /tx-rest/v2/docs and click on the Explore button to view the deployed package.  
You can see the cbpr\_mt202\_pacs00n section having two actions: A PUT /v2/run/cbpr\_mt202\_pacs00n and a POST /v2/run/cbpr\_mt202\_pacs00n.
6. In the PUT action:
  - a. Expand the PUT action and select the Try it out button.
  - b. Leave the input and output sections empty.
  - c. Under flow\_vars section, delete the entire content and replace with the commands to run the flow, for example:

```
{
 "INPUT_FILE": "C:/tests_dir/data/MT202_COR.inp",
 "OUTPUT_DOC_RESULT_XML": "C:/tests_dir/data/pacs.009.001.08.xml",
 "AUDIT_LOG": "C:/tests_dir/data/audit_log.json"
}
```
- d. Click Execute blue bar for running the flow. When flow completes execution, 200 response code is shown in the Server Response section. You can see information about the process flow in the Response body.
7. Refresh the browser to delete the deployed package and get the Swagger Explore back to /tx-rest/openapi.json.  
There will be a DELETE /v2/packages/{name} action.
8. Expand the DELETE /v2/packages/{name} action and select the Try it out.
9. In the Name field, enter the name you gave it to the created package.
10. Select the true option from the stop query drop down, select the blue Execute bar.  
You can see a response of 204 with a timestamp, if it is successful.

## Run the example using the flow command server process (flowcmdserver)

To run the example, deploy locally or to ftp server and use the flow command server process (flowcmdserver).

1. Create a server definition where the runtime is installed, if you do not have a server local definition or ftp execution definition to deploy to, in Design Server.
2. In Design Server, deploy the created package to the server definition. For example, deploy to c:\ftpserver\deployment directory.
3. Execute the flow using the flow command server:

```
flowcmdserver.exe --dir c:\ftpserver\deployment\flows --flow cbpr_mt202_pacs00n.json
--var "INPUT_FILE=C:\ftpserver\deployment\cbpr_plus\translation\mt-mx\data\MT202_COR_pacs_009_valid.inp"
--var "OUTPUT_DOC_RESULT_XML=C:\ftpserver\deployment\cbpr_plus\translation\mt-mx\data\pacs.009.001.08.xml"
--var "OUTPUT_DOC_RESULT_XML=C:\ftpserver\deployment\cbpr_plus\translation\mt-mx\data\pacs.009.001.08_ADV.xml"
--var "AUDIT_LOG=C:\ftpserver\deployment\cbpr_plus\translation\mt-mx\data\MT202_pacs_009_valid_log.json"
--audit "C:\ftpserver\deployment\cbpr_plus\translation\mt-mx\data\MT202_COR_pacs_009_valid_adt.json" -ad
```

4. The flow will report status similar to:

```
***Starting flow command server

Flow UUID: 85eabe2c-2302-4543-b347-9e18d65c6816
The flow audit file is: "C:\ftpserver\deployment\cbpr_plus\translation\mt-mx\data\MT202_COR_pacs_009_valid_adt.json"
Flow completed successfully
Elapsed time: 3048ms
```

5. Examine the flow output result file MT202\_COR\_pacs\_009\_valid.out and the audit file MT202\_COR\_pacs\_009\_valid\_adt.json.

## CBPR+ MT to MX (MT205 COR/COV to pacs.009.001.08 OR MT205 RETN to pacs.004.001.09) Translation Example

This example demonstrates the CBPR+ MT to MX (MT205 COR/COV to pacs.009.001.08 OR MT205 RETN to pacs.004.001.09) message translation using flow server.

- [What the example contains](#)

Files included in this example are as follows:

- [How to run the example](#)

This mt-mx translation will use the sample files to demonstrate the generation of SWIFT CBPR+ pacs.009.001.08 output from a MT205 message or CBPR+ pacs.004.001.09 output from a MT205 RETN message.

---

## What the example contains

Files included in this example are as follows:

- cbpr\_translation.zip
  - Files:
    - MT205\_COR.inp
    - MT205\_COR\_bad.inp
    - MT205\_RTN.inp
    - trx\_config.xml
  - Schemas:
    - head.001.001.02\_pacs\_004.xsd
    - head.001.001.02\_pacs\_009.xsd
    - pacs.004.001.09.xsd
    - pacs.009.001.08.xsd
    - swift\_iso7775\_ccyy.mtt
    - swiftroute\_funds.mtt
    - infoset.mtt
    - trx\_config.xsd
  - Maps:
    - cbpr2811\_mt205\_framework
    - cbpr2818\_mt205\_pacs009\_cov
    - cbpr2819\_mt205\_pacs009
    - cbpr2825\_mt205RTN\_pacs004
    - cbpr1502\_mtmx\_setvarlog
  - Flow:
    - cbpr\_mt205\_pacs00n
- cbprJnodesConfigIBM.tar.gz

---

## How to run the example

This mt-mx translation will use the sample files to demonstrate the generation of SWIFT CBPR+ pacs.009.001.08 output from a MT205 message or CBPR+ pacs.004.001.09 output from a MT205 RETN message.

The cbprJnodesConfigIBM.tar.gz file is available either in IBM\_financialpaymentsplus\_vn.n.n.n.zip or in UIProjectImports directory.

Extract from cbprJnodesConfigIBM.tar.gz file.

- jnodes0.jar

The following jars needs to be copied from <TX\_install\_dir>/jars:

- jackson-core-n.n.n.jar
- jackson-annotations-n.n.n.jar
- jackson-databind-n.n.n.jar

Or visit <https://repo1.maven.org/maven2/com/fasterxml/jackson/core/> for download.

Note: Recommend using the latest version.

For the non Docker environments,

- Copy jars to <TX\_install\_dir>/extjar.

For TX V11.0.1 and up, native based Design Server installation,

Copy the following into the directory defined in config.yaml server.persistence.libs, by default, this is set to /opt/txlbs:

- jvcwrap.jar
- jvalccyy.jar

Restart the running application ./ITX stop and then ./ITX start.

For the Docker environments,

- docker cp jnodes0.jar <brand>-server:/opt/<company>/<brand>/libs/.
- Restart the design server, i.e., **docker restart <brand>-server**.

Note: For RHEL, Docker is not supported, Podman can be used as a substitute since it provides a command line interface similar to Docker. Below is an example of using podman:

- podman cp jnodes0.jar <brand>-server:/opt/<company>/<brand>/libs/.
- Restart the design server, i.e., **podman restart <brand>-server**.

This example may generate failure in the audit.log.json log file to report the translation failure due to the pre-conversion checks:

```
Field 72, Line[1] should not start with "/REJT/".
If Field 72, Line[1] starts with "/RETN/", <Tag121><EndToEndTxnRef> should not be absent.
```

1. Import the cbpr\_translation.zip project into the Design Server.
2. Open the cbpr\_translation project in Design Server and view the flow cbpr\_mt205\_pacs00n.
  - a. The Flow Description points to all the maps to be executed during the flow process, which will be called from within some of the map nodes in the flow. The following is a list of maps invoked using RUN built-in function during the flow process:
    - @packagemap=cbpr\_plus/translation/mt-mx/maps/cbpr\_t9n\_mt20n\_pacs009\_cov/cbpr2818\_mt205\_pacs009\_cov
    - @packagemap=cbpr\_plus/translation/mt-mx/maps/cbpr\_t9n\_mt20nRTN\_pacs004/cbpr2825\_mt205RTN\_pacs004
    - @packagemap=cbpr\_plus/translation/mt-mx/maps/cbpr\_t9n\_mt20n\_pacs009/cbpr2819\_mt205\_pacs009
    - @packagemap=cbpr\_plus/translation/mt-mx/maps/cbpr\_t9n\_mt\_setvarlog/cbpr1502\_mtmx\_setvarlog
  - b. It utilizes the following nodes:
    - i. Source Node
      - mt\_205

This node identifies the input data to be translated in the flow. It uses the INPUT\_FILE variable to set the location of the data.
    - ii. Map Nodes
      - mt\_translate

Runs map cbpr2811\_mt205\_framework which sets flow variables, checks pre-translation conditions and translates the input file (mt) into required output (XML).
    - iii. Target Node
      - pacs.00n

This node contains the resulting translation in XML format and creates the pacs.004 or pacs.009 output as defined by variable OUTPUT\_DOC\_RESULT\_XML.
    - iv. Log Node
      - audit\_log

This node creates a log file specified by the flow variable AUDIT\_LOG.
  - c. It utilizes the following flow variables:
    - INPUT\_FILE
    - Default value for the flow variable INPUT\_FILE is ..//cbpr\_plus/translation/mt-mx/data/MT205\_COR.inp. This is the data file to be used for translation and can be customized. It is used in the Source node mt\_205.
    - OUTPUT\_DOC\_RESULT\_XML
    - Default value for the flow variable OUTPUT\_DOC\_RESULT\_XML is pacs.00n.xml. This is the location of the output file in mt format. It is used in the Target node pacs.00n. It can be customized.
    - AUDIT\_LOG
    - Default value for the flow variable AUDIT\_LOG is audit.log.json. This is the location of the audit log. It is used in the Log node audit\_log. It can be customized.
    - CONFIG\_FILE
    - Default value for the flow variable CONFIG\_FILE is ..//cbpr\_plus/translation/mt-mx/data/trx\_config.xml. This is the location of the file containing the configuration settings. It can be customized.

3. Open the flow cbpr\_mt205\_pacs00n in the Design Server. It utilizes one source node, one map node, one log node, and one target node.

4. In Design Server, create a package that contains the input files and one flow. The maps will automatically be included during deployment of the package onto the runtime server.

  - [\*\*Run the example using three different methods\*\*](#)  
You can run the example using three different methods:

---

## Run the example using three different methods

You can run the example using three different methods:

1. Running the example from the Design Server user interface.
  2. Deploying the example to tx-rest and running it using the Swagger interface.
  3. Deploying locally or to ftp server and using the flow command server process (flowcmdserver).
- [\*\*Run the example from the Design Server user interface\*\*](#)  
Use the following steps to run the example from the Design Server user interface:
    - [\*\*Run the example using the Swagger interface\*\*](#)  
To run the example, deploy to tx-rest and run it using the Swagger interface.
    - [\*\*Run the example using the flow command server process \(flowcmdserver\)\*\*](#)  
To run the example, deploy locally or to ftp server and use the flow command server process (flowcmdserver).

---

## Run the example from the Design Server user interface

Use the following steps to run the example from the Design Server user interface:

1. In the Design Server, open cbpr\_mt205\_pacs00n flow and click on Run button.
2. Set toggle switch Run On Design Server and select flow input file MT205\_COR.inp or MT205\_COR\_bad.inp.  
Note: If a default path is assigned to the variable INPUT\_FILE, not selecting an input file will use the value defined for the variable.  
Flow will run and report with the green check box. The reports list execution of all nodes.
3. Right click and select View Data on each link to examine the data.

4. Right click on the node and select View Log to see the logs.

## Run the example using the Swagger interface

To run the example, deploy to tx-rest and run it using the Swagger interface.

1. Create a Web type server definition where the runtime tx-rest is installed, if you do not have a server definition to deploy in Design Server.
2. If not running, start tx-rest on the server: <DTX\_HOME>/restapi/tomcat/dtxtomcat start tx-rest.
3. Deploy the created package to the server definition in Design Server.
4. Bring up the tx-rest Swagger UI: <DTX\_HOME>/restapi/tomcat/dtxtomcat showdocs tx-rest.
5. In the Swagger Explore window, enter /tx-rest/v2/docs and click on the Explore button to view the deployed package.  
You should see an cbpr\_mt205\_pacs00n section having two actions: A PUT /v2/run/cbpr\_mt205\_pacs00n and a POST /v2/run/cbpr\_mt205\_pacs00n.
6. In the PUT action:
  - a. Expand the PUT action and select the Try it out button.
  - b. Leave the input and output sections empty.
  - c. Under flow\_vars section, delete the entire content and replace with the commands to run the flow, for example:

```
{
 "INPUT_FILE": "C:/tests_dir/data/MT205_COR.inp",
 "OUTPUT_DOC_RESULT_XML": "C:/tests_dir/data/pacs.009.001.08.xml",
 "AUDIT_LOG": "C:/tests_dir/data/audit_log.json"
}
```

- d. Click Execute blue bar for running the flow. When flow completes execution, 200 response code is shown in the Server Response section. You can see information about the process flow in the Response body.
7. Refresh the browser to delete the deployed package and get the Swagger Explore back to /tx-rest/openapi.json.  
There will be a DELETE /v2/packages/{name} action.
8. Expand the DELETE /v2/packages/{name} action and select the Try it out.
9. In the Name field, enter the name you gave it to the created package.
10. Select the true option from the stop query drop down, select the blue Execute bar.  
You can see a response of 204 with a timestamp, if it is successful.

## Run the example using the flow command server process (flowcmdserver)

To run the example, deploy locally or to ftp server and use the flow command server process (flowcmdserver).

1. Create a server definition where the runtime is installed, if you do not have a server local definition or ftp execution definition to deploy to, in Design Server.
2. In Design Server, deploy the created package to the server definition. For example, deploy to c:\ftpserver\deployment directory.
3. Execute the flow using the flow command server.

```
flowcmdserver.exe --dir c:\ftpserver\deployment\flows --flow cbpr_mt205_pacs00n.json
--var "INPUT_FILE=C:\ftpserver\deployment\cbpr_plus\translation\mt-mx\data\MT205_COR_pacs_009_valid.inp"
--var "OUTPUT_DOC_RESULT_XML=C:\ftpserver\deployment\cbpr_plus\translation\mt-mx\data\pacs.009.001.08.xml"
--var "AUDIT_LOG=C:\ftpserver\deployment\cbpr_plus\translation\mt-mx\data\MT205_COR_pacs_009_valid_log.json"
--audit "C:\ftpserver\deployment\cbpr_plus\translation\mt-mx\data\MT205_COR_pacs_009_valid_adt.json" -ad
```

4. The flow will report status similar to:

```
***Starting flow command server

Flow UUID: 85eabe2c-2302-4543-b347-9e18d65c6816
The flow audit file is: "C:\\\\ftpserver\\\\deployment\\\\cbpr_plus\\\\translation\\\\mt-
mx\\\\data\\\\MT205_COR_pacs_009_valid_adt.json"
Flow completed successfully
Elapsed time: 3048ms
```

5. Examine the flow output result file MT205\_COR\_pacs\_009\_valid.out and the audit file MT205\_COR\_pacs\_009\_valid\_adt.json.

## CBPR+ MT message validation Example

This example shows how flow server can be used for validating any Swift MT message.

- [What the example contains](#)  
Files included in this example are as follows:
- [How to run the example](#)  
This MT validation will use the sample file to demonstrate the validation of SWIFT MT messages.

## What the example contains

Files included in this example are as follows:

- cbpr\_translation.zip
  - Files:
    - mtxxxx.inp

- trx\_config.xml
  - Schemas:
    - validator.mtt
  - Maps:
    - validate\_messages\_enh
  - Flow:
    - mt\_validation
  - cbprJnodesConfigIBM.tar.gz
- 

## How to run the example

This MT validation will use the sample file to demonstrate the validation of SWIFT MT messages.

For the non Docker environments:

- Install the latest version of the Pack for Financial Payments.

For the Docker environments

- Extract the swiftMTjvcConfig.tar.gz file available from the Pack for Financial Payments.

This MT validation will use the sample file to demonstrate the validation of SWIFT MT messages.

1. Import the cbpr\_translation.zip project into the Design Server.
2. Open the cbpr\_translation project in Design Server and view the flow mt\_validation. It utilizes the following nodes:
  - a. It utilizes the following nodes:
    - i. Source Node
      - mt\_input

This node identifies the input data to be translated in the flow. It uses the INPUT\_FILE variable to set the location of the data.
    - ii. Map Node
      - validate\_messages\_enh

Runs framework map validate\_messages\_enh which validates the input SWIFT MT message and produces validation results.
    - iii. Decision Nodes
      - RESULT\_FORMAT

This checks the flow variable REPORT\_FORMAT to decide 'xml' or 'json'.
    - iv. Passthrough Node
      - XML\_CONVERT

This node receives an extended validation report in XML format and does not modify the content, thus passing it as is to the appropriate target node.
    - v. Format Converter Node
      - JSON\_CONVERT

This node receives an extended validation report in XML format and converts it to JSON before passing it to the appropriate target node.
    - vi. Target Node
      - rslt\_xml

This node contains the resulting translation in XML format and creates the output as defined by the variable OUTPUT\_RESULT\_MT.

      - rslt\_json

This node contains the resulting translation in JSON format and creates the output as defined by the variable OUTPUT\_RESULT\_MT.
  - b. It uses the following flow variables:
    - JVC\_VALIDATION\_LEVEL  
Flow variable JVC\_VALIDATION\_LEVEL is set without any default value.
    - INPUT\_FILE  
Default value for the flow variable INPUT\_FILE is ..//tools/mt\_validation/data/mtxxx.inp. This is the data file to be used for translation and can be customized. It is used in the Source node mt\_input.
    - OUTPUT\_RESULT\_JSON  
Default value for the flow variable OUTPUT\_RESULT\_JSON is mt\_validation\_results.json. This is the location of the output file in json format. It can be customized.
    - OUTPUT\_RESULT\_XML  
Default value for the flow variable OUTPUT\_RESULT\_XML is mt\_validation\_results.xml. This is the location of the output file in xml format. It can be customized.
    - REPORT\_FORMAT  
Default value for the flow variable REPORT\_FORMAT is xml. It can be customized.
  3. In Design Server, create a package that contains the input files and one flow mt\_validation. The maps will automatically be included during deployment of the package onto the runtime server.  
Note: Check for the jvalccyy.prop file. It should be present with the latest FinPay installation (<finpay\_install\_dir>/swift/mt/jvc/maps) or it will be available in the swiftMTjvcConfig.tar.gz.
  - [Additional configuration for MT JVC validation](#)
  - [Configuration files for Design Server](#)
  - [Run the example using three different methods](#)  
You can run the example using three different methods:

---

## Additional configuration for MT JVC validation

### Pre-requisites

The swiftMTjvcConfig.tar.gz file is contained within the UIProjectImports directory.

The UIProjectImports directory is located in the following location:

<install\_dir>/packs/financial\_payments\_vn.n.n.n/UIProjectImports

Where, <install\_dir> is the location where the pack zip distro was extracted and n.n.n.n is the current version of the pack.

Extract the following from swiftMTjvcConfig.tar.gz file:

- jvcwrap.jar
- jvalccyy.jar

For Design Studio, will require administrative rights:

Copy jars to <TX\_install\_dir>/extjar.

For Design Server installation:

See Configuration files for Design Server for the steps to perform before you import <project>.zip file.

where <project>.zip can be any of the following:

cbpr\_translation.zip

---

## Configuration files for Design Server

The swiftMTjvcConfig.tar.gz file is contained within the UIProjectImports directory.

The UIProjectImports directory is located in the following location:

<install\_dir>/packs/financial\_payments\_vn.n.n.n/UIProjectImports

Where, <install\_dir> is the location where the pack zip distro was extracted and n.n.n.n is the current version of the pack..

Use these steps to run this file:

Extract the swiftMTjvcConfig.tar.gz file.

Follow the below steps, before you import <project>.zip file.

Perform the steps in the same directory location as the Design Server is installed:

1. Create a temp directory called packs.

Example: /home/user/packs

2. For the UNIX environment, FTP (in binary mode) the swiftMTjvcConfig.tar.gz to the pack's directory defined in Step 1.

3. Untar the swiftMTjvcConfig.tar.gz file.

Example: \$ tar -zxf swiftMTjvcConfig.tar.gz

4. For Container based Design Server installation:

- a. Run the **jvcsetup.sh** script.

Example: \$ ./jvcsetup.sh

- b. The script copies the following to the ITX server:

- jvcwrap.jar
- jvalccyy.jar

packs/swift folder and subfolders include the jvalccyy.prop and XML metadata.

**Optional:**

- a. Run the **cleanjvc.sh** script.

Example: \$ ./cleanjvc.sh

- b. The script removes the following to the ITX server:

- jvcwrap.jar
- jvalccyy.jar

packs/swift folder and subfolders include the jvalccyy.prop and XML metadata.

5. For Native based installation:

- a. Copy the jvalccyy.jar and jvcwrap.jar into the directory defined in config.yaml server.persistence.libs, by default, this is set to /opt/txlibs.

- b. Restart the application running ./ITX stop and then ./ITX start.

6. After importing cbpr\_translation.zip project:

- a. Copy the jvalccyy.prop file to the workdir directory under directory set in the config.yaml server.persistence.files, the default settings is /opt/txfiles/workdir.

- b. On the project, update the map rule on the validate\_message\_enh map, on output card #1 item JvalPropPath. Change the map rule from =NONE to the location of the jvalccyy.prop.

Example:

- On UNIX

```
="DS_DATA_DIR/workdir"
```

- On WINDOWS  
="DS\_DATA\_DIR\\workdir

where DS\_DATA\_DIR refers to the Design Server data directory, set in the config.yamlserver.persistence.files previously TX\_FILE\_DIR

7. Restart the Design Server: ./ITX stop and ./ITX start.

Note: To reflect any jvalccyy.prop updates, repeat step 7.

---

## Run the example using three different methods

You can run the example using three different methods:

1. Running the example from the Design Server user interface.
2. Deploying the example to tx-rest and running it using the Swagger interface.
3. Deploying locally or to ftp server and using the flow command server process (flowcmdserver).

Note:

Flow variable jvc.validation.level can be set to either 0 or 1 or 2 or no input:

0 - only do Syntax validation

1 - Network rule validation/Check Methods enforced

2 - Network rule validation/Check Methods/BIC enabled no input - will take default jvc settings

- [Run the example from the Design Server user interface](#)

Use the following steps to run the example from the Design Server user interface:

- [Run the example using the Swagger interface](#)

To run the example, deploy to tx-rest and run it using the Swagger interface.

- [Run the example using the flow command server process \(flowcmdserver\)](#)

To run the example, deploy locally or to ftp server and use the flow command server process (flowcmdserver).

---

## Run the example from the Design Server user interface

Use the following steps to run the example from the Design Server user interface:

1. In the Design Server, open mt\_validation flow and click on Run button.
2. Set toggle switch Run On Design Server and select flow input file mtxxx.inp.  
Note: If a default path is assigned to the variable INPUT\_FILE, not selecting an input file will use the value defined for the variable.  
Flow will run and report with the green check box. The report lists execution of all nodes.
3. Right click and select View Data on each link to examine the data.
4. Right click on the node and select View Log to see the logs.

---

## Run the example using the Swagger interface

To run the example, deploy to tx-rest and run it using the Swagger interface.

1. Create a Web type server definition where the runtime tx-rest is installed, if you do not have a server definition to deploy in Design Server.
2. If not running, start tx-rest on the server: <DTX\_HOME>/restapi/tomcat/dtxtomcat start tx-rest.
3. Deploy the created package to the server definition in Design Server.
4. Bring up the tx-rest Swagger UI: <DTX\_HOME>/restapi/tomcat/dtxtomcat showdocs tx-rest.
5. In the Swagger Explore window, enter /tx-rest/v2/docs and click on the Explore button to view the deployed package.  
You should see a mt\_validation section having two actions: A PUT /v2/run/mt\_validation and a POST /v2/run/mt\_validation.
6. In the PUT action:
  - a. Expand the PUT action and select the Try it out button.
  - b. Leave the input and output sections empty.
  - c. Under flow\_vars section, delete the entire content and replace with the commands to run the flow, for example:

```
{
 "INPUT_FILE": "C:/tests_dir/data/mtxxx.inp",
 "REPORT_FORMAT": "json",
 "OUTPUT_RESULT_JSON": "C:/tests_dir/data/mt_validation_results.json",
 "AUDIT_LOG": "C:/tests_dir/data/audit_log.json"
}
```

or

```
{
 "INPUT_FILE": "C:/tests_dir/data/mtxxx.inp",
 "OUTPUT_RESULT_XML": "C:/tests_dir/data/mt_validation_results.xml",
 "AUDIT_LOG": "C:/tests_dir/data/audit_log.json"
}
```

- d. Click Execute blue bar for running the flow. When flow completes execution, 200 response code is shown in the Server Response section. You can see information about the process flow in the Response body.

7. Refresh the browser to delete the deployed package and get the Swagger Explore back to /tx-rest/openapi.json.  
There will be a DELETE /v2/packages/{name} action.
8. Expand the DELETE /v2/packages/{name} action and select the Try it out.
9. In the Name field, enter the name you gave it to the created package.
10. Select the true option from the stop query drop down, select the blue Execute bar.  
You can see a response of 204 with a timestamp, if it is successful.

## Run the example using the flow command server process (flowcmdserver)

To run the example, deploy locally or to ftp server and use the flow command server process (flowcmdserver).

1. Create a server definition where the runtime is installed, if you do not have a server local definition or ftp execution definition to deploy to, in Design Server.
2. In Design Server, deploy the created package to the server definition. For example, deploy to c:\ftpserver\deployment directory.
3. Execute the flow using the flow command server. The below command is for SWIFT MT mtxxx.inp message:

```
flowcmdserver.exe --dir c:\ftpserver\deployment\flows --flow mt_validation.json
--var "INPUT_FILE=C:\ftpserver\deployment\tools\mt_validation\data\mtxxx.inp"
--var "OUTPUT_RESULT_XML=C:\ftpserver\deployment\tools\mt_validation\data\mt_validation_results.xml"
--audit "C:\ftpserver\deployment\tools\mt_validation\data\mtxxx.json" -ad
```

4. The flow will report status similar to:

```
***Starting flow command server
Flow UUID: 85eabe2c-2302-4543-b347-9e18d65c6816
The flow audit file is: "C:\\ftpserver\\deployment\\tools\\mt_validation\\data\\mtxxx.json"
Flow completed successfully
Elapsed time: 3048ms
```

5. Examine the flow output result file mt\_validation\_results.xml and the audit file mtxxx.json.

## CBPR+ Translation Examples using maps:

- [CBPR+ MX to MT \(camt.029.001.09 to MT196/MT296\) Translation Example](#)

This example demonstrates the CBPR+ MX to MT (camt.029.001.09 to MT196/MT296) message translation using maps.

- [CBPR+ MX to MT \(camt.052.001.08 to MT942\) Translation Example](#)

This example demonstrates the CBPR+ MX to MT (camt.052.001.08 to MT942) message translation using maps.

- [CBPR+ MX to MT \(camt.053.001.08 to MT940\) Translation Example](#)

This example demonstrates the CBPR+ MX to MT (camt.053.001.08 to MT940) message translation using maps.

- [CBPR+ MX to MT \(camt.054.001.08 to MT900/MT910\) Translation Example](#)

This example demonstrates the CBPR+ MX to MT (camt.054.001.08 to MT900/MT910) message translation using maps.

- [CBPR+ MX to MT \(camt.056.001.08 to MT192/MT292\) Translation Example](#)

This example demonstrates the CBPR+ MX to MT (camt.056.001.08 to MT192/MT292) message translation using maps.

- [CBPR+ MX to MT \(camt.057.001.06 to MT210\) Translation Example](#)

This example demonstrates the CBPR+ MX to MT (camt.057.001.06 to MT210) message translation using maps.

- [CBPR+ MX to MT \(camt.058.001.08 to MT292\) Translation Example](#)

This example demonstrates the CBPR+ MX to MT (camt.058.001.08 to MT292) message translation using maps.

- [CBPR+ MX to MT \(camt.107.001.01 to MT110\) Translation Example](#)

This example demonstrates the CBPR+ MX to MT (camt.107.001.01 to MT110) message translation using maps.

- [CBPR+ MX to MT \(camt.108.001.01 to MT111\) Translation Example](#)

This example demonstrates the CBPR+ MX to MT (camt.108.001.01 to MT111) message translation using maps.

- [CBPR+ MX to MT \(camt.109.001.01 to MT112\) Translation Example](#)

This example demonstrates the CBPR+ MX to MT (camt.109.001.01 to MT112) message translation using maps.

- [CBPR+ MX to MT \(pacs.002.001.10 to MT199/MT299\) Translation Example](#)

This example demonstrates the CBPR+ MX to MT (pacs.002.001.10 to MT199/MT299) message translation using maps.

- [CBPR+ MX to MT \(pacs.008.001.08 SWIFT Go to MT103\) Translation Example](#)

This example demonstrates the CBPR+ MX to MT (pacs.008.001.08 SWIFT Go to MT103) message translation using maps.

- [CBPR+ MX to MT \(pacs.009.001.08 to MT202 \(CORE/ADV/COVE\)\) Translation Example](#)

This example demonstrates the CBPR+ MX to MT (pacs.009.001.08 to MT202 (CORE/ADV/COVE)) message translation using maps.

- [CBPR+ MX to MT \(pacs.009.001.08 to MT205 \(CORE/COVE\)\) Translation Example](#)

This example demonstrates the CBPR+ MX to MT (pacs.009.001.08 to MT205 (CORE/COVE)) message translation using maps.

- [CBPR+ MX to MT \(pacs.004.001.09 to MT103 RETN/MT202 RETN/MT205 RETN\) Translation Example](#)

This example demonstrates the CBPR+ MX to MT (pacs.004.001.09 to MT103 RETN/MT202 RETN/MT205 RETN) message translation using maps.

- [CBPR+ MX to MT \(pacs.008.001.08 to MT103\) Translation Example](#)

This example demonstrates the CBPR+ MX to MT (pacs.008.001.08 to MT103) message translation using maps.

- [CBPR+ MT to MX \(MT900/MT910 to camt.054.001.08\) Translation Example](#)

This example demonstrates the CBPR+ MT to MX (MT900/MT910 to camt.054.001.08) message translation using maps.

- [CBPR+ MT to MX \(MT192/MT292 to camt.056.001.08\) Translation Example](#)

This example demonstrates the CBPR+ MT to MX (MT192/MT292 to camt.056.001.08) message translation using maps.

- [CBPR+ MT to MX \(MT196/MT296 to camt.029.001.09\) Translation Example](#)

This example demonstrates the CBPR+ MT to MX (MT196/MT296 to camt.029.001.09) message translation using maps.

- [CBPR+ MT to MX \(MT103 COR/MT103 STP to pacs.008.001.08 or MT103 COR RETN to pacs.004.001.09\) Translation Example](#)

This example demonstrates the CBPR+ MT to MX (MT103 COR/MT103 STP to pacs.008.001.08 or MT103 COR RETN to pacs.004.001.09) message translation using maps.

- [CBPR+ MT to MX \(MT103 SWIFT Go to pacs.008.001.08\) Translation Example](#)

This example demonstrates the CBPR+ MT to MX (MT103 SWIFT Go to pacs.008.001.08) message translation using maps.

- **[CBPR+ MT to MX \(MT202 \(core/adv/cove\) to pacs.009.001.08 \(core/adv/cove\) or MT202 \(RETN\) to pacs.004.001.09\) Translation Example](#)**  
This example demonstrates the CBPR+ MT to MX (MT202 (core/adv/cove) to pacs.009.001.08 (core/adv/cove) or MT202 (RETN) to pacs.004.001.09) message translation using maps.
- **[CBPR+ MT to MX \(MT205 \(core/cove\) to pacs.009.001.08 \(core/cove\) or MT205 \(RETN\) to pacs.004.001.09\) Translation Example](#)**  
This example demonstrates the CBPR+ MT to MX (MT205 (core/cove) to pacs.009.001.08 (core/cove) or MT205 (RETN) to pacs.004.001.09) message translation using maps.
- **[CBPR+ MT to MX Translation \(using TX maps\) Example](#)**  
This example shows how TX maps can be used for translating CBPR+ MT to MX message formats.
- **[CBPR+ MX to MT Translation \(using TX maps\) Example](#)**  
This example shows how TX maps can be used for translating CBPR+ MX to MT message formats.

---

## CBPR+ MX to MT (camt.029.001.09 to MT196/MT296) Translation Example

This example demonstrates the CBPR+ MX to MT (camt.029.001.09 to MT196/MT296) message translation using maps.

- **[What the example contains](#)**  
Files included in this example are as follows:
- **[How to run the example](#)**  
This mx-xt translation will use the sample files to demonstrate the generation of SWIFT MT196 or MT296 message output from a CBPR+ camt.029.001.09 XML message.

---

### What the example contains

Files included in this example are as follows:

- cbpr\_translation.zip
  - Files:
    - env\_camt\_029\_mt196\_valid.xml
    - env\_camt\_029\_mt296\_valid.xml
    - trx\_config.xml
  - Schemas:
    - head.001.001.02\_camt\_029.xsd
    - camt.029.001.09.xsd
    - trx\_config.xsd
    - infoset.mtt
    - swift\_iso7775\_ccyy.mtt
    - swiftroute\_funds.mtt
  - Maps:
    - cbpr2520\_camt029\_framework
    - cbpr2501\_mxmt\_setvarlog
    - cbpr2522\_camt029\_translate
    - cbpr2523\_camt029\_mtn96
- cbprJnodesConfigIBM.tar.gz

---

### How to run the example

This mx-xt translation will use the sample files to demonstrate the generation of SWIFT MT196 or MT296 message output from a CBPR+ camt.029.001.09 XML message.

Extract from <packs\_install\_dir>/UIProjectImports/cbprJnodesConfigIBM.tar.gz file.

- jnodes0.jar

The following jars needs to be copied from <TX\_install\_dir>/jars:

- jackson-core-n.n.n.jar
- jackson-annotations-n.n.n.jar
- jackson-databind-n.n.n.jar

Or visit <https://repo1.maven.org/maven2/com/fasterxml/jackson/core/> for download.

Note: Recommend using the latest version.

For the non-Docker environments,

- Copy jars to <TX\_install\_dir>/extjar

For TX V11.0.1 and up, native based Design Server installation,

Copy the following into the directory defined in config.yaml server.persistence.libs, by default, this is set to /opt/txlabs:

- jvcwrap.jar
- jvalccyy.jar

Restart the running application ./ITX stop and then ./ITX start.

For the Docker environments,

- docker cp jnodes0.jar <brand>-server:/opt/<company>/<brand>/libs/
- Restart the design server, that is, **docker restart** <brand>-server

Note: For RHEL, Docker is not supported. Podman can be used as a substitute since it provides a command line interface similar to Docker. Following is an example of using Podman.

- podman cp jnodes0.jar <brand>-server:/opt/<company>/<brand>/libs/
- Restart the design server, that is, **podman restart** <brand>-server

1. Import the cbpr\_translation.zip project into the Design Server.
  2. Open the cbpr\_translation project in Design Server and build the following maps:
    - cbpr2520\_camt029\_framework
    - cbpr2501\_mxmt\_setvarlog
    - cbpr2522\_camt029\_translate
    - cbpr2523\_camt029\_mtn96
  3. Run the main map cbpr2520\_camt029\_framework.
- The following output files will be generated:
- mt\_out.txt  
File contains the translated MT196 or MT296 SWIFT message from CBPR+ camt\_029 message based on the <OrgnlMsgNmId>.
  - audit\_msg.json  
Reports translation logs, including failure due to the pre-conversion checks.

## CBPR+ MX to MT (camt.052.001.08 to MT942) Translation Example

This example demonstrates the CBPR+ MX to MT (camt.052.001.08 to MT942) message translation using maps.

- **What the example contains**

Files included in this example are as follows:

- cbpr\_translation.zip
  - Files:
    - env\_camt\_052\_mt942\_valid.xml
    - env\_camt\_052\_mt942\_bad.xml
    - trx\_config.xml
  - Schemas:
    - head.001.001.02\_camt\_052.xsd
    - camt.052.001.08.xsd
    - trx\_config.xsd
    - infoset.mtt
    - swift\_iso7775\_ccyy.mtt
    - swiftroute\_funds.mtt
  - Maps:
    - cbpr2546\_camt052\_framework
    - cbpr2547\_camt052\_translate
    - cbpr2501\_mxmt\_setvarlog
    - cbpr2548\_camt052\_mt942
- cbprJnodesConfigIBM.tar.gz

## What the example contains

Files included in this example are as follows:

- cbpr\_translation.zip
  - Files:
    - env\_camt\_052\_mt942\_valid.xml
    - env\_camt\_052\_mt942\_bad.xml
    - trx\_config.xml
  - Schemas:
    - head.001.001.02\_camt\_052.xsd
    - camt.052.001.08.xsd
    - trx\_config.xsd
    - infoset.mtt
    - swift\_iso7775\_ccyy.mtt
    - swiftroute\_funds.mtt
  - Maps:
    - cbpr2546\_camt052\_framework
    - cbpr2547\_camt052\_translate
    - cbpr2501\_mxmt\_setvarlog
    - cbpr2548\_camt052\_mt942
- cbprJnodesConfigIBM.tar.gz

## How to run the example

This mx-mt translation will use the sample files to demonstrate the generation of SWIFT MT942 message output from a CBPR+ camt.052.001.08 XML message.

Extract from <packs\_install\_dir>/UIProjectImports/cbprJnodesConfigIBM.tar.gz file.

- jnodes0.jar

The following jars needs to be copied from <TX\_install\_dir>/jars:

- jackson-core-n.n.n.jar
- jackson-annotations-n.n.n.jar
- jackson-databind-n.n.n.jar

Or visit <https://repo1.maven.org/maven2/com/fasterxml/jackson/core/> for download.

Note: Recommend using the latest version.

For the non-Docker environments,

- Copy jars to <TX\_install\_dir>/extjar

For TX V11.0.1 and up, native based Design Server installation,

Copy the following into the directory defined in config.yaml server.persistence.libs, by default, this is set to /opt/txlibs:

- jvcwrap.jar
- jvalccyy.jar

Restart the running application ./ITX stop and then ./ITX start.

For the Docker environments,

- docker cp jnodes0.jar <brand>-server:/opt/<company>/<brand>/libs/
- Restart the design server, that is, **docker restart** <brand>-server

Note: For RHEL, Docker is not supported. Podman can be used as a substitute since it provides a command line interface similar to Docker. Following is an example of using Podman.

- podman cp jnodes0.jar <brand>-server:/opt/<company>/<brand>/libs/
- Restart the design server, that is, **podman restart** <brand>-server

1. Import the cbpr\_translation.zip project into the Design Server.

2. Open the cbpr\_translation project in Design Server and build the following maps:

- cbpr2546\_camt052\_framework
- cbpr2547\_camt052\_translate
- cbpr2548\_camt052\_mt942
- cbpr2501\_mxmt\_setvarlog

3. Run the main map cbpr2546\_camt052\_framework.

The following output files will be generated:

- mt\_out.txt

File contains the translated MT942 from CBPR+ camt\_052 message.

- audit\_msg.json

Reports translation logs, including failure due to the pre-conversion checks:

- <ElctrncSeqNb> must be present in <BkToCstmrAcctRpt><Rpt><ElctrncSeqNb> and its length must be less than or equal to 5.

OR

<LglSeqNb> must be present in <BkToCstmrAcctRpt><Rpt><LglSeqNb> and its length must be less than or equal to 5.

- There must be a maximum 190 occurrence of <Ntry> in <BkToCstmrAcctRpt><Rpt><Ntry>.

- There must be present any ONE of the following:

- <BkToCstmrAcctRpt><Rpt><Ntry>
- <BkToCstmrAcctRpt><Rpt><TxSummry><TtlCdtNtries><NbOfNtries> AND <BkToCstmrAcctRpt><Rpt><TxSummry><TtlCdtNtries><Sum>
- <BkToCstmrAcctRpt><Rpt><TxSummry><TtlDbtNtries><NbOfNtries> AND <BkToCstmrAcctRpt><Rpt><TxSummry><TtlDbtNtries><Sum>

- For all occurrence of <BkToCstmrAcctRpt><Rpt><Ntry>, value of <Ccy> in <BkToCstmrAcctRpt><Rpt><Ntry><Amt><Ccy> must be equal to value of <Ccy> in <BkToCstmrAcctRpt><Rpt><Acct><Ccy>, TotalNumberOfDigits <BkToCstmrAcctRpt><Rpt><Ntry><Amt> must be less than or equal to 14.

## CBPR+ MX to MT (camt.053.001.08 to MT940) Translation Example

This example demonstrates the CBPR+ MX to MT (camt.053.001.08 to MT940) message translation using maps.

- **What the example contains**

Files included in this example are as follows:

- **How to run the example**

This mx-mt translation will use the sample files to demonstrate the generation of SWIFT MT940 message output from a CBPR+ camt.053.001.08 XML message.

- **What the example contains**

Files included in this example are as follows:

- **How to run the example**

This mx-mt translation will use the sample files to demonstrate the generation of SWIFT MT940 message output from a CBPR+ camt.053.001.08 XML message.

## What the example contains

Files included in this example are as follows:

- cbpr\_translation.zip
  - Files:
    - env\_camt\_053\_mt940\_valid.xml
    - env\_camt\_053\_mt940\_bad.xml
    - trx\_config.xml
  - Schemas:
    - head.001.001.02\_camt\_053.xsd
    - camt.053.001.08.xsd
    - trx\_config.xsd

- infoset.mtt
- swift\_iso7775\_ccyy.mtt
- swiftroute\_funds.mtt
- Maps:
  - cbpr2530\_camt053\_framework
  - cbpr2531\_camt053\_translate
  - cbpr2501\_mxmt\_setvarlog
  - cbpr2534\_camt053\_mt940
- cbprJnodesConfigIBM.tar.gz

## How to run the example

This mx-mt translation will use the sample files to demonstrate the generation of SWIFT MT940 message output from a CBPR+ camt.053.001.08 XML message.

Extract from `<packs_install_dir>/UIProjectImports/cbprJnodesConfigIBM.tar.gz` file.

- jnodes0.jar

The following jars needs to be copied from `<TX_install_dir>/jars`:

- jackson-core-n.n.n.jar
- jackson-annotations-n.n.n.jar
- jackson-databind-n.n.n.jar

Or visit <https://repo1.maven.org/maven2/com/fasterxml/jackson/core/> for download.

Note: Recommend using the latest version.

For the non-Docker environments,

- Copy jars to `<TX_install_dir>/extjar`

For TX V11.0.1 and up, native based Design Server installation,

Copy the following into the directory defined in config.yaml server.persistence.libs, by default, this is set to /opt/txlibs:

- jvcwrap.jar
- jvalccyy.jar

Restart the running application ./ITX stop and then ./ITX start.

For the Docker environments,

- docker cp jnodes0.jar <brand>-server:/opt/<company>/<brand>/libs/
- Restart the design server, that is, **docker restart** <brand>-server

Note: For RHEL, Docker is not supported. Podman can be used as a substitute since it provides a command line interface similar to Docker. Following is an example of using Podman.

- podman cp jnodes0.jar <brand>-server:/opt/<company>/<brand>/libs/
- Restart the design server, that is, **podman restart** <brand>-server

1. Import the cbpr\_translation.zip project into the Design Server.

2. Open the cbpr\_translation project in Design Server and build the following maps:

- cbpr2530\_camt053\_framework
- cbpr2531\_camt053\_translate
- cbpr2534\_camt053\_mt940
- cbpr2501\_mxmt\_setvarlog

3. Run the main map cbpr2530\_camt053\_framework.

The following output files will be generated:

- mt\_out.txt  
File contains the translated MT940 from CBPR+ camt\_053 message.
- audit\_msg.json  
Reports translation logs, including failure due to the pre-conversion checks as:

- **<ElctrncSeqNb>** must be present in **<BkToCstmrStmt><Stmt><ElctrncSeqNb>** and its length must be less than or equal to 5.

OR

**<LglSeqNb>** must be present in **<BkToCstmrStmt><Stmt><LglSeqNb>** and its length must be less than or equal to 5.

- If **<PgNb>** in **<BkToCstmrStmt><Stmt><StmtPgntr><PgNb>** is equal to 1, then there must be only 1 occurrence of **<Cd>** with value OPBD in **<BkToCstmrStmt><Stmt><Bal><Tp><CdOrPrtry><Cd>** and if **<Cd>** is present in **<BkToCstmrStmt><Stmt><Bal><Tp><SubTp><Cd>**, then the value must be different from INTM.
- If **<PgNb>** in **<BkToCstmrStmt><Stmt><StmtPgntr><PgNb>** is greater than 1 then there must be only 1 occurrence of **<Cd>** with value OPBD in **<BkToCstmrStmt><Stmt><Bal><Tp><CdOrPrtry><Cd>** and **<Cd>** must be present in **<BkToCstmrStmt><Stmt><Bal><Tp><SubTp><Cd>**, value must be INTM.
- If **<LastPgInd>** in **<BkToCstmrStmt><Stmt><StmtPgntr><LastPgInd>** is equal to true then there must be only 1 occurrence of **<Cd>** with value CLBD in **<BkToCstmrStmt><Stmt><Bal><Tp><CdOrPrtry><Cd>** and if **<Cd>** is present in **<BkToCstmrStmt><Stmt><Bal><Tp><SubTp><Cd>**, then the value must be different from INTM.

- If <LastPgInd> in <BkToCstmrStmt><Stmt><StmtPgntr><LastPgInd> is equal to false then there must be only 1 occurrence of <Cd> with value CLBD in <BkToCstmrStmt><Stmt><Bal><Tp><CdOrPrtry><Cd> and <Cd> must be present in <BkToCstmrStmt><Stmt><Bal><Tp><SubTp><Cd>, value must be INTM.
- There must be only 1 occurrence of <Cd> with value CLAV in <BkToCstmrStmt><Stmt><Bal><Tp><CdOrPrtry><Cd>.
- There must be a maximum 190 occurrence of <Ntry> in <BkToCstmrStmt><Stmt><Ntry>.
- For all occurrence of <BkToCstmrStmt><Stmt><Ntry>, value of <Ccy> in <BkToCstmrStmt><Stmt><Ntry><Amt><Ccy> must be equal to value of <Ccy> in <BkToCstmrStmt><Stmt><Acct><Ccy>, TotalNumberOfDigits <BkToCstmrStmt><Stmt><Ntry><Amt> must be less than or equal to 14.
- For the occurrence of <BkToCstmrStmt><Stmt><Bal>, if <Cd> is present with value OPBD in <BkToCstmrStmt><Stmt><Bal><Tp><CdOrPrtry><Cd> then first two char of currency code must be the same for other occurrence of <BkToCstmrStmt><Stmt><Bal> in <BkToCstmrStmt><Stmt><Bal><Tp><CdOrPrtry><Cd> with value CLBD OR CLAV.

## CBPR+ MX to MT (camt.054.001.08 to MT900/MT910) Translation Example

This example demonstrates the CBPR+ MX to MT (camt.054.001.08 to MT900/MT910) message translation using maps.

- [What the example contains](#)

Files included in this example are as follows:

- [How to run the example](#)

This mx-xt translation will use the sample files to demonstrate the generation of SWIFT MT900 or MT910 message output from a CBPR+ camt.054.001.08 XML message.

## What the example contains

Files included in this example are as follows:

- cbpr\_translation.zip
  - Files:
    - env\_camt\_054\_bad.xml
    - env\_camt\_054\_crdt.xml
    - env\_camt\_054\_dbit.xml
    - trx\_config.xml
  - Schemas:
    - head.001.001.02\_camt\_054.xsd
    - camt.054.001.08.xsd
    - trx\_config.xsd
    - infoSet.mtt
    - swift\_iso7775\_ccyy.mtt
    - swiftRoute\_funds.mtt
  - Maps:
    - cbpr2510\_camt054\_framework
    - cbpr2501\_mxmt\_setvarlog
    - cbpr2512\_camt054\_translate
    - cbpr2513\_camt054\_mt900
    - cbpr2514\_camt054\_mt910
- cbprJnodesConfigIBM.tar.gz

## How to run the example

This mx-xt translation will use the sample files to demonstrate the generation of SWIFT MT900 or MT910 message output from a CBPR+ camt.054.001.08 XML message.

Extract from <packs\_install\_dir>/UIProjectImports/cbprJnodesConfigIBM.tar.gz file.

- jnodes0.jar

The following jars needs to be copied from <TX\_install\_dir>/jars:

- jackson-core-n.n.n.jar
- jackson-annotations-n.n.n.jar
- jackson-databind-n.n.n.jar

Or visit <https://repo1.maven.org/maven2/com/fasterxml/jackson/core/> for download.

Note: Recommend using the latest version.

For the non-Docker environments,

- Copy jars to <TX\_install\_dir>/extjar

For TX V11.0.1 and up, native based Design Server installation,

Copy the following into the directory defined in config.yaml server.persistence.libs, by default, this is set to /opt/txlbs:

- jvcwrap.jar

- jvalccyy.jar

Restart the running application ./ITX stop and then ./ITX start.

For the Docker environments,

- docker cp jnodes0.jar <brand>-server:/opt/<company>/<brand>/libs/
- Restart the design server, that is, **docker restart** <brand>-server

Note: For RHEL, Docker is not supported. Podman can be used as a substitute since it provides a command line interface similar to Docker. Following is an example of using Podman.

- podman cp jnodes0.jar <brand>-server:/opt/<company>/<brand>/libs/
- Restart the design server, that is, **podman restart** <brand>-server

1. Import the cbpr\_translation.zip project into the Design Server.

2. Open the cbpr\_translation project in Design Server and build the following maps:

- cbpr2510\_camt054\_framework
- cbpr2501\_mxmt\_setvarlog
- cbpr2512\_camt054\_translate
- cbpr2513\_camt054\_mt900
- cbpr2514\_camt054\_mt910

3. Run the main map cbpr2510\_camt054\_framework.

The following output files will be generated:

- mt\_out.txt

File contains the translated MT900 or MT910 SWIFT message from CBPR+ camt\_054 message based on the **<CdtDbtInd>** which could be CRDT (will generate MT910) or DBIT (will generate MT900).

- audit\_msg.json

Reports translation logs, including failure due to the pre-conversion checks as:

- Either **<CdtDbtInd>** is not one of these (CRDT, DBIT) or **<Sts><Cd>** is not BOOK.
- **TotalNumberOfDigits <Ntry><Amt>** less than or equal to 14.
- Only one occurrence is allowed for **<Ntfctn>, <Ntry>, <NtryDtls>, and <TxDtls>**.
- Either one must exist **<Ntfctn><Ntry><ValDt> or <Ntfctn><Ntry><NtryDtls><TxDtls><RltdDts><IntrBkSttlmDt>**.

## CBPR+ MX to MT (camt.056.001.08 to MT192/MT292) Translation Example

This example demonstrates the CBPR+ MX to MT (camt.056.001.08 to MT192/MT292) message translation using maps.

- **[What the example contains](#)**

Files included in this example are as follows:

- **[How to run the example](#)**

This mx-xt translation will use the sample files to demonstrate the generation of SWIFT MT192 or MT292 message output from a CBPR+ camt.056.001.08 XML message.

## What the example contains

Files included in this example are as follows:

- cbpr\_translation.zip
  - Files:
    - env\_camt\_056\_mt192\_valid.xml
    - env\_camt\_056\_mt292\_valid.xml
    - trx\_config.xml
  - Schemas:
    - head.001.001.02\_camt\_056.xsd
    - camt.056.001.08.xsd
    - trx\_config.xsd
    - infoset.mtt
    - swift\_iso7775\_ccyy.mtt
    - swiftroute\_funds.mtt
  - Maps:
    - cbpr2540\_camt056\_framework
    - cbpr2501\_mxmt\_setvarlog
    - cbpr2542\_camt056\_translate
    - cbpr2543\_camt056\_mtn92
- cbprJnodesConfigIBM.tar.gz

## How to run the example

This mx-xt translation will use the sample files to demonstrate the generation of SWIFT MT192 or MT292 message output from a CBPR+ camt.056.001.08 XML message.

Extract from **<packs\_install\_dir>/UIProjectImports/cbprJnodesConfigIBM.tar.gz** file.

- jnodes0.jar

The following jars needs to be copied from <TX\_install\_dir>/jars:

- jackson-core-n.n.n.jar
- jackson-annotations-n.n.n.jar
- jackson-databind-n.n.n.jar

Or visit <https://repo1.maven.org/maven2/com/fasterxml/jackson/core/> for download.

Note: Recommend using the latest version.

For the non-Docker environments,

- Copy jars to <TX\_install\_dir>/extjar

For TX V11.0.1 and up, native based Design Server installation,

Copy the following into the directory defined in config.yaml server.persistence.libs, by default, this is set to /opt/txls:

- jvcwrap.jar
- jvalccyy.jar

Restart the running application ./ITX stop and then ./ITX start.

For the Docker environments,

- docker cp jnodes0.jar <brand>-server:/opt/<company>/<brand>/libs/
- Restart the design server, that is, **docker restart <brand>-server**

Note: For RHEL, Docker is not supported. Podman can be used as a substitute since it provides a command line interface similar to Docker. Following is an example of using Podman.

- podman cp jnodes0.jar <brand>-server:/opt/<company>/<brand>/libs/
- Restart the design server, that is, **podman restart <brand>-server**

1. Import the cbpr\_translation.zip project into the Design Server.

2. Open the cbpr\_translation project in Design Server and build the following maps:

- cbpr2540\_camt056\_framework
- cbpr2501\_mxmt\_setvarlog
- cbpr2542\_camt056\_translate
- cbpr2543\_camt056\_mtn92

3. Run the main map cbpr2540\_camt056\_framework.

The following output files will be generated:

- mt\_out.txt  
File contains the translated MT192 or MT292 SWIFT message from CBPR+ camt\_056 message based on the <OrgnlMsgNmId>.
- audit\_msg.json  
Reports translation logs, including failure due to the pre-conversion checks.

## CBPR+ MX to MT (camt.057.001.06 to MT210) Translation Example

This example demonstrates the CBPR+ MX to MT (camt.057.001.06 to MT210) message translation using maps.

- [What the example contains](#)

Files included in this example are as follows:

- [How to run the example](#)

This mx-xt translation will use the sample files to demonstrate the generation of SWIFT MT210 message output from a CBPR+ camt.057.001.06 XML message.

### What the example contains

Files included in this example are as follows:

- cbpr\_translation.zip
  - Files:
    - env\_camt\_057\_bad.xml
    - env\_camt\_057\_valid.xml
    - trx\_config.xml
  - Schemas:
    - head.001.001.02\_camt\_057.xsd
    - camt.057.001.06.xsd
    - trx\_config.xsd
    - infoset.mtt
    - swift\_iso7775\_ccyy.mtt
    - swiftroute\_funds.mtt
  - Maps:
    - cbpr2600\_camt057\_framework
    - cbpr2601\_camt057\_translate
    - cbpr2501\_mxmt\_setvarlog
    - cbpr2605\_camt057\_mt210

- cbprJnodesConfigIBM.tar.gz

---

## How to run the example

This mx-*mt* translation will use the sample files to demonstrate the generation of SWIFT MT210 message output from a CBPR+ camt.057.001.06 XML message.

Extract from <packs\_install\_dir>/UIProjectImports/cbprJnodesConfigIBM.tar.gz file.

- jnodes0.jar

The following jars needs to be copied from <TX\_install\_dir>/jars:

- jackson-core-n.n.n.jar
- jackson-annotations-n.n.n.jar
- jackson-databind-n.n.n.jar

Or visit <https://repo1.maven.org/maven2/com/fasterxml/jackson/core/> for download.

Note: Recommend using the latest version.

For the non-Docker environments,

- Copy jars to <TX\_install\_dir>/extjar

For TX V11.0.1 and up, native based Design Server installation,

Copy the following into the directory defined in config.yaml server.persistence.libs, by default, this is set to /opt/tlibs:

- jvcwrap.jar
- jvalccyy.jar

Restart the running application ./ITX stop and then ./ITX start.

For the Docker environments,

- docker cp jnodes0.jar <brand>-server:/opt/<company>/<brand>/libs/
- Restart the design server, that is, **docker restart** <brand>-server

Note: For RHEL, Docker is not supported. Podman can be used as a substitute since it provides a command line interface similar to Docker. Following is an example of using Podman.

- podman cp jnodes0.jar <brand>-server:/opt/<company>/<brand>/libs/
- Restart the design server, that is, **podman restart** <brand>-server

1. Import the cbpr\_translation.zip project into the Design Server.

2. Open the cbpr\_translation project in Design Server and build the following maps:

- cbpr2600\_camt057\_framework
- cbpr2601\_camt057\_translate
- cbpr2501\_mxmt\_setvarlog
- cbpr2605\_camt057\_mt210

3. Run the main map cbpr2600\_camt057\_framework.

The following output files will be generated:

- mt\_out.txt  
File contains the translated MT210 from CBPR+ camt\_057 message.

- audit\_msg.json

Reports translation logs, including failure due to the pre-conversion checks as:

- **NumberOfOccurrences <Ntfctn><Itm>** should not be greater than 1.
- Currency should not be any of these {XAU, XAG, XPD, XPT} for <Ntfctn><Itm><Amt><Ccy>.
- **TotalNumberOfDigits <Ntry><Amt>** less than or equal to 14.

---

## CBPR+ MX to MT (camt.058.001.08 to MT292) Translation Example

This example demonstrates the CBPR+ MX to MT (camt.058.001.08 to MT292) message translation using maps.

- **What the example contains**

Files included in this example are as follows:

- **How to run the example**

This mx-*mt* translation will use the sample files to demonstrate the generation of SWIFT MT292 message output from a CBPR+ camt.058.001.08 XML message.

---

## What the example contains

Files included in this example are as follows:

- cbpr\_translation.zip
  - Files:
    - env\_camt\_058\_valid.xml

- env\_camt\_058\_bad.xml
- trx\_config.xml
- Schemas:
  - head.001.001.02\_camt\_058.xsd
  - camt.058.001.08.xsd
  - trx\_config.xsd
  - infoset.mtt
  - swift\_iso7775\_ccyy.mtt
  - swiftroute\_funds.mtt
- Maps:
  - cbpr2610\_camt058\_framework
  - cbpr2611\_camt058\_translate
  - cbpr2612\_camt058\_mt292
  - cbpr2501\_mxmt\_setvarlog
- cbprJnodesConfigIBM.tar.gz

## How to run the example

This mx-mt translation will use the sample files to demonstrate the generation of SWIFT MT292 message output from a CBPR+ camt.058.001.08 XML message.

Extract from <packs\_install\_dir>/UIProjectImports/cbprJnodesConfigIBM.tar.gz file.

- jnodes0.jar

The following jars needs to be copied from <TX\_install\_dir>/jars:

- jackson-core-n.n.n.jar
- jackson-annotations-n.n.n.jar
- jackson-databind-n.n.n.jar

Or visit <https://repo1.maven.org/maven2/com/fasterxml/jackson/core/> for download.

Note: Recommend using the latest version.

For the non-Docker environments,

- Copy jars to <TX\_install\_dir>/extjar

For TX V11.0.1 and up, native based Design Server installation,

Copy the following into the directory defined in config.yaml server.persistence.libs, by default, this is set to /opt/txlbs:

- jvcwrap.jar
- jvalccyy.jar

Restart the running application ./ITX stop and then ./ITX start.

For the Docker environments,

- docker cp jnodes0.jar <brand>-server:/opt/<company>/<brands>/libs/
- Restart the design server, that is, **docker restart** <brand>-server

Note: For RHEL, Docker is not supported. Podman can be used as a substitute since it provides a command line interface similar to Docker. Following is an example of using Podman.

- podman cp jnodes0.jar <brand>-server:/opt/<company>/<brand>/libs/
- Restart the design server, that is, **podman restart** <brand>-server

1. Import the cbpr\_translation.zip project into the Design Server.
2. Open the cbpr\_translation project in Design Server and build the following maps:
  - cbpr2610\_camt058\_framework
  - cbpr2611\_camt058\_translate
  - cbpr2612\_camt058\_mt292
  - cbpr2501\_mxmt\_setvarlog

3. Run the main map cbpr2610\_camt058\_framework.

The following output files will be generated:

- mt\_out.txt  
File contains the translated MT292 SWIFT message from CBPR+ camt.058 message.
- audit\_msg.json  
Reports translation logs, including failure if the input message is not camt.058.001.08

## CBPR+ MX to MT (camt.107.001.01 to MT110) Translation Example

This example demonstrates the CBPR+ MX to MT (camt.107.001.01 to MT110) message translation using maps.

- **What the example contains**

Files included in this example are as follows:

- [How to run the example](#)

This mx-*mt* translation will use the sample files to demonstrate the generation of SWIFT MT110 message output from a CBPR+ camt.107.001.01 XML message.

---

## What the example contains

Files included in this example are as follows:

- cbpr\_translation.zip
  - Files:
    - env\_camt\_107\_valid.xml
    - trx\_config.xml
  - Schemas:
    - head.001.001.02\_camt\_107.xsd
    - camt.107.001.01.xsd
    - trx\_config.xsd
    - infoSet.mtt
    - swift\_iso7775\_ccyy.mtt
    - swiftRoute\_funds.mtt
  - Maps:
    - cbpr2620\_camt107\_framework
    - cbpr2621\_camt107\_translate
    - cbpr2622\_camt107\_mt110
    - cbpr2501\_mxmt\_setvarlog
- cbprJnodesConfigIBM.tar.gz

---

## How to run the example

This mx-*mt* translation will use the sample files to demonstrate the generation of SWIFT MT110 message output from a CBPR+ camt.107.001.01 XML message.

Extract from <packs\_install\_dir>/UIProjectImports/cbprJnodesConfigIBM.tar.gz file.

- jnodes0.jar

The following jars needs to be copied from <TX\_install\_dir>/jars:

- jackson-core-n.n.n.jar
- jackson-annotations-n.n.n.jar
- jackson-databind-n.n.n.jar

Or visit <https://repo1.maven.org/maven2/com/fasterxml/jackson/core/> for download.

Note: Recommend using the latest version.

For the non-Docker environments,

- Copy jars to <TX\_install\_dir>/extjar

For TX V11.0.1 and up, native based Design Server installation,

Copy the following into the directory defined in config.yaml server.persistence.libs, by default, this is set to /opt/txlbs:

- jvcwrap.jar
- jvalccyy.jar

Restart the running application ./ITX stop and then ./ITX start.

For the Docker environments,

- docker cp jnodes0.jar <brand>-server:/opt/<company>/<brand>/libs/
- Restart the design server, that is, **docker restart** <brand>-server

Note: For RHEL, Docker is not supported. Podman can be used as a substitute since it provides a command line interface similar to Docker. Following is an example of using Podman.

- podman cp jnodes0.jar <brand>-server:/opt/<company>/<brand>/libs/
- Restart the design server, that is, **podman restart** <brand>-server

1. Import the cbpr\_translation.zip project into the Design Server.

2. Open the cbpr\_translation project in Design Server and build the following maps:

- cbpr2620\_camt107\_framework
- cbpr2621\_camt107\_translate
- cbpr2622\_camt107\_mt110
- cbpr2501\_mxmt\_setvarlog

3. Run the main map cbpr2620\_camt107\_framework.

The following output files will be generated:

- mt\_out.txt  
File contains the translated MT110 from SWIFT message from CBPR+ camt.107 message.
- audit\_msg.json  
Reports translation logs, including failure if the input message is not camt.107.001.01

---

## CBPR+ MX to MT (camt.108.001.01 to MT111) Translation Example

This example demonstrates the CBPR+ MX to MT (camt.108.001.01 to MT111) message translation using maps.

- **What the example contains**

Files included in this example are as follows:

- cbpr\_translation.zip
  - Files:
    - env\_camt\_108\_valid.xml
    - trx\_config.xml
  - Schemas:
    - head.001.001.02\_camt\_108.xsd
    - camt.108.001.01.xsd
    - trx\_config.xsd
    - infoset.mtt
    - swift\_iso7775\_ccyy.mtt
    - swiftroute\_funds.mtt
  - Maps:
    - cbpr2630\_camt108\_framework
    - cbpr2631\_camt108\_translate
    - cbpr2632\_camt108\_mt111
    - cbpr2501\_mxmt\_setvarlog
- cbprJnodesConfigIBM.tar.gz

---

## What the example contains

Files included in this example are as follows:

- cbpr\_translation.zip
  - Files:
    - env\_camt\_108\_valid.xml
    - trx\_config.xml
  - Schemas:
    - head.001.001.02\_camt\_108.xsd
    - camt.108.001.01.xsd
    - trx\_config.xsd
    - infoset.mtt
    - swift\_iso7775\_ccyy.mtt
    - swiftroute\_funds.mtt
  - Maps:
    - cbpr2630\_camt108\_framework
    - cbpr2631\_camt108\_translate
    - cbpr2632\_camt108\_mt111
    - cbpr2501\_mxmt\_setvarlog
- cbprJnodesConfigIBM.tar.gz

---

## How to run the example

This mx-mt translation will use the sample files to demonstrate the generation of SWIFT MT111 message output from a CBPR+ camt.108.001.01 XML message.

Extract from <packs\_install\_dir>/UIProjectImports/cbprJnodesConfigIBM.tar.gz file.

- jnodes0.jar

The following jars needs to be copied from <TX\_install\_dir>/jars:

- jackson-core-n.n.n.jar
- jackson-annotations-n.n.n.jar
- jackson-databind-n.n.n.jar

Or visit <https://repo1.maven.org/maven2/com/fasterxml/jackson/core/> for download.

Note: Recommend using the latest version.

For the non-Docker environments,

- Copy jars to <TX\_install\_dir>/extjar

For TX V11.0.1 and up, native based Design Server installation,

Copy the following into the directory defined in config.yaml server.persistence.libs, by default, this is set to /opt/tlxlibs:

- jvcwrap.jar
- jvalccyy.jar

Restart the running application ./ITX stop and then ./ITX start.

For the Docker environments,

- docker cp jnodes0.jar <brand>-server:/opt/<company>/<brand>/libs/
- Restart the design server, that is, **docker restart** <brand>-server

Note: For RHEL, Docker is not supported. Podman can be used as a substitute since it provides a command line interface similar to Docker. Following is an example of using Podman.

- podman cp jnodes0.jar <brand>-server:/opt/<company>/<brand>/libs/
- Restart the design server, that is, **podman restart** <brand>-server

1. Import the cbpr\_translation.zip project into the Design Server.
2. Open the cbpr\_translation project in Design Server and build the following maps:
  - cbpr2630\_camt108\_framework

- cbpr2631\_camt108\_translate
  - cbpr2632\_camt108\_mt111
  - cbpr2501\_mxmt\_setvarlog
3. Run the main map cbpr2630\_camt108\_framework.
- The following output files will be generated:
- mt\_out.txt  
File contains the translated MT111 SWIFT message from CBPR+ camt.108 message.
  - audit\_msg.json  
Reports translation logs, including failure if the input message is not camt.108.001.01

## CBPR+ MX to MT (camt.109.001.01 to MT112) Translation Example

This example demonstrates the CBPR+ MX to MT (camt.109.001.01 to MT112) message translation using maps.

- **What the example contains**  
Files included in this example are as follows:
- **How to run the example**  
This mx-xt translation will use the sample files to demonstrate the generation of SWIFT MT112 message output from a CBPR+ camt.109.001.01 XML message.

## What the example contains

Files included in this example are as follows:

- cbpr\_translation.zip
  - Files:
    - env\_camt\_109\_valid.xml
    - trx\_config.xml
  - Schemas:
    - head.001.001.02\_camt\_109.xsd
    - camt.109.001.01.xsd
    - trx\_config.xsd
    - infoset.mtt
    - swift\_iso7775\_ccyy.mtt
    - swiftroute\_funds.mtt
  - Maps:
    - cbpr2640\_camt109\_framework
    - cbpr2641\_camt109\_translate
    - cbpr2642\_camt109\_mt112
    - cbpr2501\_mxmt\_setvarlog
- cbprJnodesConfigIBM.tar.gz

## How to run the example

This mx-xt translation will use the sample files to demonstrate the generation of SWIFT MT112 message output from a CBPR+ camt.109.001.01 XML message.

Extract from <packs\_install\_dir>/UIProjectImports/cbprJnodesConfigIBM.tar.gz file.

- jnodes0.jar

The following jars needs to be copied from <TX\_install\_dir>/jars:

- jackson-core-n.n.n.jar
- jackson-annotations-n.n.n.jar
- jackson-databind-n.n.n.jar

Or visit <https://repo1.maven.org/maven2/com/fasterxml/jackson/core/> for download.

Note: Recommend using the latest version.

For the non-Docker environments,

- Copy jars to <TX\_install\_dir>/extjar

For TX V11.0.1 and up, native based Design Server installation,

Copy the following into the directory defined in config.yaml server.persistence.libs, by default, this is set to /opt/txlibs:

- jvcwrap.jar
- jvalccyy.jar

Restart the running application ./ITX stop and then ./ITX start.

For the Docker environments,

- docker cp jnodes0.jar <brand>-server:/opt/<company>/<brand>/libs/

- Restart the design server, that is, **docker restart <brand>-server**

Note: For RHEL, Docker is not supported. Podman can be used as a substitute since it provides a command line interface similar to Docker. Following is an example of using Podman.

- podman cp jnodes0.jar <brand>-server:/opt/<company>/<brand>/libs/
- Restart the design server, that is, **podman restart <brand>-server**

1. Import the cbpr\_translation.zip project into the Design Server.
2. Open the cbpr\_translation project in Design Server and build the following maps:

- cbpr2640\_camt109\_framework
- cbpr2641\_camt109\_translate
- cbpr2642\_camt109\_mt112
- cbpr2501\_mxmt\_setvarlog

3. Run the main map cbpr2640\_camt109\_framework.

The following output files will be generated:

- mt\_out.txt  
File contains the translated MT112 from SWIFT message from CBPR+ camt.109 message.
- audit\_msg.json  
Reports translation logs, including failure if the input message is not camt.109.001.01

## CBPR+ MX to MT (pacs.002.001.10 to MT199/MT299) Translation Example

This example demonstrates the CBPR+ MX to MT (pacs.002.001.10 to MT199/MT299) message translation using maps.

- **What the example contains**

Files included in this example are as follows:

- **How to run the example**
- This mx-xt translation will use the sample files to demonstrate the generation of SWIFT MT199 or MT299 message output from a CBPR+ pacs.002.001.10 XML message.

## What the example contains

Files included in this example are as follows:

- cbpr\_translation.zip
  - Files:
    - env\_pacs\_002\_199.xml
    - env\_pacs\_002\_299.xml
    - env\_pacs\_002\_bad.xml
    - trx\_config.xml
  - Schemas:
    - head.001.001.02\_pacs\_002.xsd
    - pacs.002.001.10.xsd
    - trx\_config.xsd
    - infoset.mtt
    - swift\_iso7775\_ccyy.mtt
    - swiftroute\_funds.mtt
  - Maps:
    - cbpr2500\_pacs002\_framework
    - cbpr2501\_mxmt\_setvarlog
    - cbpr2502\_pacs002\_translate
    - cbpr2503\_pacs002\_mtn99
- cbprJnodesConfigIBM.tar.gz

## How to run the example

This mx-xt translation will use the sample files to demonstrate the generation of SWIFT MT199 or MT299 message output from a CBPR+ pacs.002.001.10 XML message.

Extract from <pacs\_install\_dir>/UIProjectImports/cbprJnodesConfigIBM.tar.gz file.

- jnodes0.jar

The following jars needs to be copied from <TX\_install\_dir>/jars:

- jackson-core-n.n.n.jar
- jackson-annotations-n.n.n.jar
- jackson-databind-n.n.n.jar

Or visit <https://repo1.maven.org/maven2/com/fasterxml/jackson/core/> for download.

Note: Recommend using the latest version.

For the non-Docker environments,

- Copy jars to <TX\_install\_dir>/extjar

For TX V11.0.1 and up, native based Design Server installation,

Copy the following into the directory defined in config.yaml server.persistence.libs, by default, this is set to /opt/txlibs:

- jvcwrap.jar
- jvalccyy.jar

Restart the running application ./ITX stop and then ./ITX start.

For the Docker environments,

- docker cp jnodes0.jar <brand>-server:/opt/<company>/<brand>/libs/
- Restart the design server, that is, **docker restart <brand>-server**

Note: For RHEL, Docker is not supported. Podman can be used as a substitute since it provides a command line interface similar to Docker. Following is an example of using Podman.

- podman cp jnodes0.jar <brand>-server:/opt/<company>/<brand>/libs/
- Restart the design server, that is, **podman restart <brand>-server**

1. Import the cbpr\_translation.zip project into the Design Server.

2. Open the cbpr\_translation project in Design Server and build the following maps:

- cbpr2500\_pacs002\_framework
- cbpr2501\_mxmt\_setvarlog
- cbpr2502\_pacs002\_translate
- cbpr2503\_pacs002\_mtn99

3. Run the main map cbpr2500\_pacs002\_framework.

The following output files will be generated:

- mt\_out.txt

File contains the translated MT199 or MT299 SWIFT message from CBPR+ pacs.002 message based on the <OrgnlMsgNmId> which could be pacs.008 or MT103 (will generate MT199) or pacs.009 or MT202 or MT205 (will generate MT299).

- audit\_msg.json

Reports translation logs, including failure due to the pre-conversion checks as:

- Either <OrgnlMsgNmId> is not one of these (pacs.008, MT103, pacs.009, MT202, MT205) or <TxSts> transaction status is not RJCT.

## CBPR+ MX to MT (pacs.008.001.08 SWIFT Go to MT103) Translation Example

This example demonstrates the CBPR+ MX to MT (pacs.008.001.08 SWIFT Go to MT103) message translation using maps.

- [What the example contains](#)

Files included in this example are as follows:

- [How to run the example](#)

This mx-mt translation will use the sample files to demonstrate the generation of SWIFT MT103 message output from a CBPR+ pacs.008.001.08 SWIFT Go XML message.

### What the example contains

Files included in this example are as follows:

- cbpr\_translation.zip
  - Files:
    - env\_pacs\_008\_SGO.xml
    - trx\_config.xml
  - Schemas:
    - head.001.001.02\_pacs\_008.xsd
    - pacs.008.001.08.xsd
    - trx\_config.xsd
    - infoset.mtt
    - swift\_iso7775\_ccyy.mtt
    - swiftroute\_funds.mtt
  - Maps:
    - cbpr2700\_pacs008\_framework
    - cbpr2701\_pacs008\_translate
    - cbpr2704\_pacs008\_mt103\_sgo
    - cbpr2501\_mxmt\_setvarlog
- cbprJnodesConfigIBM.tar.gz

### How to run the example

This mx-mt translation will use the sample files to demonstrate the generation of SWIFT MT103 message output from a CBPR+ pacs.008.001.08 SWIFT Go XML message.

Extract from <pacs\_install\_dir>/UIProjectImports/cbprJnodesConfigIBM.tar.gz file.

- jnodes0.jar

The following jars needs to be copied from <TX\_install\_dir>/jars:

- jackson-core-n.n.n.jar
- jackson-annotations-n.n.n.jar
- jackson-databind-n.n.n.jar

Or visit <https://repo1.maven.org/maven2/com/fasterxml/jackson/core/> for download.

Note: Recommend using the latest version.

For the non-Docker environments,

- Copy jars to <TX\_install\_dir>/extjar

For TX V11.0.1 and up, native based Design Server installation,

Copy the following into the directory defined in config.yaml server.persistence.libs, by default, this is set to /opt/txls:

- jvcwrap.jar
- jvalccyy.jar

Restart the running application ./ITX stop and then ./ITX start.

For the Docker environments,

- docker cp jnodes0.jar <brand>-server:/opt/<company>/<brand>/libs/
- Restart the design server, that is, **docker restart** <brand>-server

Note: For RHEL, Docker is not supported. Podman can be used as a substitute since it provides a command line interface similar to Docker. Following is an example of using Podman.

- podman cp jnodes0.jar <brand>-server:/opt/<company>/<brand>/libs/
- Restart the design server, that is, **podman restart** <brand>-server

1. Import the cbpr\_translation.zip project into the Design Server.
2. Open the cbpr\_translation project in Design Server and build the following maps:
  - cbpr2700\_pacs008\_framework
  - cbpr2701\_pacs008\_translate
  - cbpr2704\_pacs008\_mt103\_sgo
  - cbpr2501\_mxmt\_setvarlog
3. Run the main map cbpr2700\_pacs008\_framework.

The following output files will be generated:

- mt\_out.txt  
File contains the translated MT103 SWIFT message from CBPR+ pacs.008 SWIFT Go message.
- audit\_msg.json  
Reports translation logs, including failure due to the pre-conversion checks as:
  - <IntrBkSttlmAmt> Commodities currencies {XAU, XAG, XPD, XPT} Not allowed in Field 32A and 32B.
  - Number Of Occurrences of <CreditTransferTransactionInformation> is greater than 1.

## CBPR+ MX to MT (pacs.009.001.08 to MT202 (CORE/ADV/COVE)) Translation Example

This example demonstrates the CBPR+ MX to MT (pacs.009.001.08 to MT202 (CORE/ADV/COVE)) message translation using maps.

- **[What the example contains](#)**

Files included in this example are as follows:

- **[How to run the example](#)**

This mx-mt translation will use the sample files to demonstrate the generation of SWIFT MT202 (CORE/ADV/COVE) message output from a CBPR+ pacs.009.001.08 XML message.

### What the example contains

Files included in this example are as follows:

- cbpr\_translation.zip
  - Files:
    - env\_pacs\_009\_adv\_valid.xml
    - env\_pacs\_009\_adv\_bad.xml
    - env\_pacs\_009\_mt202cor\_valid.xml
    - env\_pacs\_009\_cor\_bad.xml
    - env\_pacs\_009\_mt202cov\_valid.xml
    - env\_pacs\_009\_cov\_bad.xml
    - trx\_config.xml
  - Schemas:
    - head.001.001.02\_pacs\_009\_adv.xsd
    - head.001.001.02\_pacs\_009\_cov.xsd

- head.001.001.02\_pacs\_009.xsd
  - pacs.009.001.08\_adv.xsd
  - pacs.009.001.08\_cov.xsd
  - pacs.009.001.08.xsd
  - trx\_config.xsd
  - infoset.mtt
  - swift\_iso7775\_ccyy.mtt
  - swiftroute\_funds.mtt
  - Maps:
    - cbpr2710\_pacs009\_mt202\_framework
    - cbpr2501\_mxmt\_setvarlog
    - cbpr2713\_pacs009\_mt202adv\_trx
    - cbpr2714\_pacs009\_mt202cov\_trx
    - cbpr2712\_pacs009\_mt202\_trx
    - cbpr2715\_pacs009\_mt202
  - cbprJnodesConfigIBM.tar.gz
- 

## How to run the example

This mx-xt translation will use the sample files to demonstrate the generation of SWIFT MT202 (CORE/ADV/COVE) message output from a CBPR+ pacs.009.001.08 XML message.

Extract from <packs\_install\_dir>/UIProjectImports/cbprJnodesConfigIBM.tar.gz file.

- jnodes0.jar

The following jars needs to be copied from <TX\_install\_dir>/jars:

- jackson-core-n.n.n.jar
- jackson-annotations-n.n.n.jar
- jackson-databind-n.n.n.jar

Or visit <https://repo1.maven.org/maven2/com/fasterxml/jackson/core/> for download.

Note: Recommend using the latest version.

For the non-Docker environments,

- Copy jars to <TX\_install\_dir>/extjar

For TX V11.0.1 and up, native based Design Server installation,

Copy the following into the directory defined in config.yaml server.persistence.libs, by default, this is set to /opt/tlxlibs:

- jvcwrap.jar
- jvalccyy.jar

Restart the running application ./ITX stop and then ./ITX start.

For the Docker environments,

- docker cp jnodes0.jar <brand>-server:/opt/<company>/<brand>/libs/
- Restart the design server, that is, **docker restart <brand>-server**

Note: For RHEL, Docker is not supported. Podman can be used as a substitute since it provides a command line interface similar to Docker. Following is an example of using Podman.

- podman cp jnodes0.jar <brand>-server:/opt/<company>/<brand>/libs/
- Restart the design server, that is, **podman restart <brand>-server**

1. Import the cbpr\_translation.zip project into the Design Server.
2. Open the cbpr\_translation project in Design Server and build the following maps:

- cbpr2710\_pacs009\_mt202\_framework
- cbpr2501\_mxmt\_setvarlog
- cbpr2713\_pacs009\_mt202adv\_trx
- cbpr2714\_pacs009\_mt202cov\_trx
- cbpr2712\_pacs009\_mt202\_trx
- cbpr2715\_pacs009\_mt202

3. Run the main map cbpr2710\_pacs009\_mt202\_framework.

The following output files will be generated:

- mt\_out.txt  
File contains the translated MT202 (CORE/ADV/COVE) SWIFT message from CBPR+ pacs.009 message.

- audit\_msg.json  
Reports translation logs, including failure due to the pre-conversion checks as:

• <**IntrBkSttlmAmt**> Commodities currencies {XAU, XAG, XPD, XPT} Not allowed in Field 32A and 32B.

## CBPR+ MX to MT (pacs.009.001.08 to MT205 (CORE/COVE)) Translation Example

This example demonstrates the CBPR+ MX to MT (pacs.009.001.08 to MT205 (CORE/COVE)) message translation using maps.

- [What the example contains](#)

Files included in this example are as follows:

- [How to run the example](#)

This mx-mt translation will use the sample files to demonstrate the generation of SWIFT MT205 (CORE/COVE) message output from a CBPR+ pacs.009.001.08 XML message.

---

## What the example contains

Files included in this example are as follows:

- cbpr\_translation.zip
  - Files:
    - env\_pacs\_009\_mt205cor\_valid.xml
    - env\_pacs\_009\_cor\_bad.xml
    - env\_pacs\_009\_mt205cov\_valid.xml
    - env\_pacs\_009\_cov\_bad.xml
    - trx\_config.xml
  - Schemas:
    - head.001.001.02\_pacs\_009\_adv.xsd
    - head.001.001.02\_pacs\_009\_cov.xsd
    - head.001.001.02\_pacs\_009.xsd
    - pacs.009.001.08\_adv.xsd
    - pacs.009.001.08\_cov.xsd
    - pacs.009.001.08.xsd
    - trx\_config.xsd
    - infoset.mtt
    - swift\_iso7775\_ccyy.mtt
    - swiftroute\_funds.mtt
  - Maps:
    - cbpr2724\_pacs009\_mt205\_framework
    - cbpr2501\_mxmt\_setvarlog
    - cbpr2726\_pacs009\_mt205cov\_trx
    - cbpr2727\_pacs009\_mt205
- cbprJnodesConfigIBM.tar.gz

---

## How to run the example

This mx-mt translation will use the sample files to demonstrate the generation of SWIFT MT205 (CORE/COVE) message output from a CBPR+ pacs.009.001.08 XML message.

Extract from <packs\_install\_dir>/UIProjectImports/cbprJnodesConfigIBM.tar.gz file.

- jnodes0.jar

The following jars needs to be copied from <TX\_install\_dir>/jars:

- jackson-core-n.n.n.jar
- jackson-annotations-n.n.n.jar
- jackson-databind-n.n.n.jar

Or visit <https://repo1.maven.org/maven2/com/fasterxml/jackson/core/> for download.

Note: Recommend using the latest version.

For the non-Docker environments,

- Copy jars to <TX\_install\_dir>/extjar

For TX V11.0.1 and up, native based Design Server installation,

Copy the following into the directory defined in config.yaml server.persistence.libs, by default, this is set to /opt/tlxlibs:

- jvcwrap.jar
- jvalccyy.jar

Restart the running application ./ITX stop and then ./ITX start.

For the Docker environments,

- docker cp jnodes0.jar <brand>-server:/opt/<company>/<brand>/libs/
- Restart the design server, that is, **docker restart** <brand>-server

Note: For RHEL, Docker is not supported. Podman can be used as a substitute since it provides a command line interface similar to Docker. Following is an example of using Podman.

- podman cp jnodes0.jar <brand>-server:/opt/<company>/<brand>/libs/
- Restart the design server, that is, **podman restart** <brand>-server

1. Import the cbpr\_translation.zip project into the Design Server.
2. Open the cbpr\_translation project in Design Server and build the following maps:

- cbpr2724\_pacs009\_mt205\_framework
  - cbpr2501\_mxmt\_setvarlog
  - cbpr2726\_pacs009\_mt205cov\_trx
  - cbpr2727\_pacs009\_mt205
3. Run the main map cbpr2724\_pacs009\_mt205\_framework.
- The following output files will be generated:
- mt\_out.txt  
File contains the translated MT205 (CORE/COVE) SWIFT message from CBPR+ pacs.009 message.
  - audit\_msg.json  
Reports translation logs, including failure due to the pre-conversion checks as:
    - <IntrBkSttlmAmt> Commodities currencies {XAU, XAG, XPD, XPT} Not allowed in Field 32A and 32B.

## CBPR+ MX to MT (pacs.004.001.09 to MT103 RETN/MT202 RETN/MT205 RETN) Translation Example

This example demonstrates the CBPR+ MX to MT (pacs.004.001.09 to MT103 RETN/MT202 RETN/MT205 RETN) message translation using maps.

- [What the example contains](#)  
Files included in this example are as follows:
- [How to run the example](#)  
This mt-mx translation will use the sample files to demonstrate the generation of SWIFT MTnnn RETN message output from a CBPR+ pacs.004.001.09 XML message.

### What the example contains

Files included in this example are as follows:

- cbpr\_translation.zip
  - Files:
    - env\_pacs\_004\_mt202 rtn\_valid.xml
    - env\_pacs\_004\_mt205 rtn\_valid.xml
    - env\_pacs\_004\_mt103 rtn\_valid.xml
    - env\_pacs\_004\_mt202 rtn\_bad.xml
    - env\_pacs\_004\_mt205 rtn\_bad.xml
    - env\_pacs\_004\_mt103 rtn\_bad.xml
    - trx\_config.xml
  - Schemas:
    - head.001.001.02\_pacs\_004.xsd
    - pacs.004.001.09.xsd
    - trx\_config.xsd
    - infoset.mtt
    - swift\_iso7775\_ccyy.mtt
    - swiftroute\_funds.mtt
  - Maps:
    - cbpr2850\_pacs004\_mttnnn\_RTN\_frw
    - cbpr2851\_pacs004RTN\_trx
    - cbpr2852\_pacs004\_mt202RTN
    - cbpr2853\_pacs004\_mt205RTN
    - cbpr2857\_pacs004\_mt103RTN
    - cbpr2501\_mxmt\_setvarlog
- cbprJnodesConfigIBM.tar.gz

### How to run the example

This mt-mx translation will use the sample files to demonstrate the generation of SWIFT MTnnn RETN message output from a CBPR+ pacs.004.001.09 XML message.

Extract from <pacs\_install\_dir>/UIProjectImports/cbprJnodesConfigIBM.tar.gz file.

- jnodes0.jar

The following jars needs to be copied from <TX\_install\_dir>/jars:

- jackson-core-n.n.n.jar
- jackson-annotations-n.n.n.jar
- jackson-databind-n.n.n.jar

Or visit <https://repo1.maven.org/maven2/com/fasterxml/jackson/core/> for download.

Note: Recommend using the latest version.

For the non-Docker environments,

- Copy jars to <TX\_install\_dir>/extjar

For TX V11.0.1 and up, native based Design Server installation,

Copy the following into the directory defined in config.yaml server.persistence.libs, by default, this is set to /opt/txlbs:

- jvcwrap.jar
- jvalccyy.jar

Restart the running application ./ITX stop and then ./ITX start.

For the Docker environments,

- docker cp jnodes0.jar <brand>-server:/opt/<company>/<brand>/libs/
- Restart the design server, that is, **docker restart** <brand>-server

Note: For RHEL, Docker is not supported. Podman can be used as a substitute since it provides a command line interface similar to Docker. Following is an example of using Podman.

- podman cp jnodes0.jar <brand>-server:/opt/<company>/<brand>/libs/
- Restart the design server, that is, **podman restart** <brand>-server

1. Import the cbpr\_translation.zip project into the Design Server.

2. Open the cbpr\_translation project in Design Server and build the following maps:

- cbpr2850\_pacs004\_mtnnn\_RTN\_frw
- cbpr2851\_pacs004RTN\_trx
- cbpr2852\_pacs004\_mt202RTN
- cbpr2853\_pacs004\_mt205RTN
- cbpr2857\_pacs004\_mt103RTN
- cbpr2501\_mxmt\_setvarlog

3. Run the main map cbpr2850\_pacs004\_mtnnn\_RTN\_frw.

The following output files will be generated:

- mt\_out.txt

File contains the translated MT103 RETN/MT202 RETN/MT205 RETN SWIFT message from CBPR+ pacs.004 message.

- audit\_msg.json

Reports translation logs, including failure due to the pre-conversion checks as:

- <IntrBkSttlmAmt> Commodities currencies {XAU, XAG, XPD, XPT} Not allowed in Field 32A and 32B.
- Count of <TxInf> is greater than 1.
- <RtrChain><Dbtr><Pty> is present.
- <RtrChain><Cdtr><Pty> is present.

## CBPR+ MX to MT (pacs.008.001.08 to MT103) Translation Example

This example demonstrates the CBPR+ MX to MT (pacs.008.001.08 to MT103) message translation using maps.

- [What the example contains](#)

Files included in this example are as follows:

- [How to run the example](#)

This mx-mt translation will use the sample files to demonstrate the generation of SWIFT MT103 message output from a CBPR+ pacs.008.001.08 XML message.

## What the example contains

Files included in this example are as follows:

- cbpr\_translation.zip
  - Files:
    - env\_pacs\_008\_103.xml
    - env\_pacs\_008\_STP.xml
    - env\_pacs\_008\_bad.xml
    - trx\_config.xml
  - Schemas:
    - head.001.001.02\_pacs\_008.xsd
    - pacs.008.001.08.xsd
    - trx\_config.xsd
    - infoset.mtt
    - swift\_iso7775\_ccyy.mtt
    - swiftroute\_funds.mtt
  - Maps:
    - cbpr2700\_pacs008\_framework
    - cbpr2701\_pacs008\_translate
    - cbpr2703\_pacs008\_mt103
    - cbpr2709\_pacs008\_stp\_translate
    - cbpr2501\_mxmt\_setvarlog
- cbprJnodesConfigIBM.tar.gz

## How to run the example

This mx-mt translation will use the sample files to demonstrate the generation of SWIFT MT103 message output from a CBPR+ pacs.008.001.08 XML message.

Extract from <packs\_install\_dir>/UIProjectImports/cbprJnodesConfigIBM.tar.gz file.

- jnodes0.jar

The following jars needs to be copied from <TX\_install\_dir>/jars:

- jackson-core-n.n.n.jar
- jackson-annotations-n.n.n.jar
- jackson-databind-n.n.n.jar

Or visit <https://repo1.maven.org/maven2/com/fasterxml/jackson/core/> for download.

Note: Recommend using the latest version.

For the non-Docker environments,

- Copy jars to <TX\_install\_dir>/extjar

For TX V11.0.1 and up, native based Design Server installation,

Copy the following into the directory defined in config.yaml server.persistence.libs, by default, this is set to /opt/txlbs:

- jvcwrap.jar
- jvalccyy.jar

Restart the running application ./ITX stop and then ./ITX start.

For the Docker environments,

- docker cp jnodes0.jar <brand>-server:/opt/<company>/<brand>/libs/
- Restart the design server, that is, **docker restart** <brand>-server

Note: For RHEL, Docker is not supported. Podman can be used as a substitute since it provides a command line interface similar to Docker. Following is an example of using Podman.

- podman cp jnodes0.jar <brand>-server:/opt/<company>/<brand>/libs/
- Restart the design server, that is, **podman restart** <brand>-server

1. Import the cbpr\_translation.zip project into the Design Server.

2. Open the cbpr\_translation project in Design Server and build the following maps:

- cbpr2700\_pacs008\_framework
- cbpr2701\_pacs008\_translate
- cbpr2703\_pacs008\_mt103
- cbpr2709\_pacs008\_stp\_translate
- cbpr2501\_mxmt\_setvarlog

3. Run the main map cbpr2700\_pacs008\_framework.

The following output files will be generated:

- mt\_out.txt  
File contains the translated MT103 SWIFT message from CBPR+ pacs.008 message.
- audit\_msg.json  
Reports translation logs, including failure due to the pre-conversion checks as:
  - <IntrBkSttlmAmt> Commodities currencies {XAU, XAG, XPD, XPT} Not allowed in Field 32A and 32B.

## CBPR+ MT to MX (MT900/MT910 to camt.054.001.08) Translation Example

This example demonstrates the CBPR+ MT to MX (MT900/MT910 to camt.054.001.08) message translation using maps.

- **What the example contains**

Files included in this example are as follows:

- **How to run the example**

This mt-mx translation will use the sample files to demonstrate the generation of CBPR+ camt.054.001.08 XML message output from a SWIFT MT900 or MT910 message.

## What the example contains

Files included in this example are as follows:

- cbpr\_translation.zip
  - Files:
    - MT900.inp
    - MT910.inp
  - Schemas:
    - head.001.001.02\_camt\_054.xsd
    - camt.054.001.08.xsd
    - swift\_iso7775\_ccyy.mtt
    - swiftroute\_funds.mtt
    - infoset.mtt

- Maps:
  - cbpr1500\_mt9n0\_framework
  - cbpr1501\_mt9n0\_camt\_054
  - cbpr1502\_mtmx\_setvarlog
- cbprJnodesConfigIBM.tar.gz

---

## How to run the example

This mt-mx translation will use the sample files to demonstrate the generation of CBPR+ camt.054.001.08 XML message output from a SWIFT MT900 or MT910 message.

Extract from <packs\_install\_dir>/UIProjectImports/cbprJnodesConfigIBM.tar.gz file.

- jnodes0.jar

The following jars needs to be copied from <TX\_install\_dir>/jars:

- jackson-core-n.n.n.jar
- jackson-annotations-n.n.n.jar
- jackson-databind-n.n.n.jar

Or visit <https://repo1.maven.org/maven2/com/fasterxml/jackson/core/> for download.

Note: Recommend using the latest version.

For the non-Docker environments,

- Copy jars to <TX\_install\_dir>/extjar

For TX V11.0.1 and up, native based Design Server installation,

Copy the following into the directory defined in config.yaml server.persistence.libs, by default, this is set to /opt/tlibs:

- jvcwrap.jar
- jvalccyy.jar

Restart the running application ./ITX stop and then ./ITX start.

For the Docker environments,

- docker cp jnodes0.jar <brand>-server:/opt/<company>/<brand>/libs/
- Restart the design server, that is, **docker restart** <brand>-server

Note: For RHEL, Docker is not supported. Podman can be used as a substitute since it provides a command line interface similar to Docker. Following is an example of using Podman.

- podman cp jnodes0.jar <brand>-server:/opt/<company>/<brand>/libs/
- Restart the design server, that is, **podman restart** <brand>-server

1. Import the cbpr\_translation.zip project into the Design Server.
2. Open the cbpr\_translation project in Design Server and build the following maps:
  - cbpr1500\_mt9n0\_framework
  - cbpr1501\_mt9n0\_camt\_054
  - cbpr1502\_mtmx\_setvarlog
3. Run the main map cbpr1500\_mt9n0\_framework.

The following output files will be generated:

- mx\_out.xml  
File contains the translated CBPR+ camt.054.001.08 message from MT900 or MT910 SWIFT message.
- audit\_msg.json  
Reports translation logs, including failure due to the pre-conversion checks as:
  - If the input file is a SWIFT message other than MT900 or MT910 or not a SWIFT message.

---

## CBPR+ MT to MX (MT192/MT292 to camt.056.001.08) Translation Example

This example demonstrates the CBPR+ MT to MX (MT192/MT292 to camt.056.001.08) message translation using maps.

- [What the example contains](#)

Files included in this example are as follows:

- [How to run the example](#)

This mt-mx translation will use the sample files to demonstrate the generation of CBPR+ camt.056.001.08 XML message output from a SWIFT MT192 or MT292 message.

---

## What the example contains

Files included in this example are as follows:

- cbpr\_translation.zip

- Files:
  - MT192.inp
  - MT292.inp
- Schemas:
  - head.001.001.02\_camt\_056.xsd
  - camt.056.001.08.xsd
  - swift\_iso7775\_ccyy.mtt
  - swiftroute\_funds.mtt
  - infoset.mtt
- Maps:
  - cbpr2840\_mtn92\_framework
  - cbpr2841\_mtn92\_camt056
  - cbpr1502\_mttx\_setvarlog
- cbprJnodesConfigIBM.tar.gz

## How to run the example

This mt-mx translation will use the sample files to demonstrate the generation of CBPR+ camt.056.001.08 XML message output from a SWIFT MT192 or MT292 message.

Extract from <packs\_install\_dir>/UIProjectImports/cbprJnodesConfigIBM.tar.gz file.

- jnodes0.jar

The following jars needs to be copied from <TX\_install\_dir>/jars:

- jackson-core-n.n.n.jar
- jackson-annotations-n.n.n.jar
- jackson-databind-n.n.n.jar

Or visit <https://repo1.maven.org/maven2/com/fasterxml/jackson/core/> for download.

Note: Recommend using the latest version.

For the non-Docker environments,

- Copy jars to <TX\_install\_dir>/extjar

For TX V11.0.1 and up, native based Design Server installation,

Copy the following into the directory defined in config.yaml server.persistence.libs, by default, this is set to /opt/tlibs:

- jvcwrap.jar
- jvalccyy.jar

Restart the running application ./ITX stop and then ./ITX start.

For the Docker environments,

- docker cp jnodes0.jar <brand>-server:/opt/<company>/<brand>/libs/
- Restart the design server, that is, **docker restart** <brand>-server

Note: For RHEL, Docker is not supported. Podman can be used as a substitute since it provides a command line interface similar to Docker. Following is an example of using Podman.

- podman cp jnodes0.jar <brand>-server:/opt/<company>/<brand>/libs/
- Restart the design server, that is, **podman restart** <brand>-server

1. Import the cbpr\_translation.zip project into the Design Server.
2. Open the cbpr\_translation project in Design Server and build the following maps:

- cbpr2840\_mtn92\_framework
- cbpr2841\_mtn92\_camt056
- cbpr1502\_mttx\_setvarlog

3. Run the main map cbpr2840\_mtn92\_framework.

The following output files will be generated:

- mx\_out.xml  
File contains the translated CBPR+ camt.056.001.08 message from MT192 or MT292 SWIFT message.

- audit\_msg.json

Reports translation logs, including failure due to the pre-conversion checks as:

- If the input file is a SWIFT message other than MT192 or MT292 or not a SWIFT message.

## CBPR+ MT to MX (MT196/MT296 to camt.029.001.09) Translation Example

This example demonstrates the CBPR+ MT to MX (MT196/MT296 to camt.029.001.09) message translation using maps.

- **What the example contains**

Files included in this example are as follows:

- **How to run the example**

This mt-mx translation will use the sample files to demonstrate the generation of CBPR+ camt.029.001.09 XML message output from a SWIFT MT196 or MT296 message.

## What the example contains

Files included in this example are as follows:

- cbpr\_translation.zip
  - Files:
    - MT196.inp
    - MT296.inp
  - Schemas:
    - head.001.001.02\_camt\_029.xsd
    - camt.029.001.09.xsd
    - swift\_iso7775\_ccyy.mtt
    - swiftroute\_funds.mtt
    - infoset.mtt
  - Maps:
    - cbpr2820\_mtn96\_framework
    - cbpr2821\_mtn96\_camt029
    - cbpr1502\_mttx\_setvarlog
- cbprJnodesConfigIBM.tar.gz

## How to run the example

This mt-mx translation will use the sample files to demonstrate the generation of CBPR+ camt.029.001.09 XML message output from a SWIFT MT196 or MT296 message.

Extract from <packs\_install\_dir>/UIProjectImports/cbprJnodesConfigIBM.tar.gz file.

- jnodes0.jar

The following jars needs to be copied from <TX\_install\_dir>/jars:

- jackson-core-n.n.n.jar
- jackson-annotations-n.n.n.jar
- jackson-databind-n.n.n.jar

Or visit <https://repo1.maven.org/maven2/com/fasterxml/jackson/core/> for download.

Note: Recommend using the latest version.

For the non-Docker environments,

- Copy jars to <TX\_install\_dir>/extjar

For TX V11.0.1 and up, native based Design Server installation,

Copy the following into the directory defined in config.yaml server.persistence.libs, by default, this is set to /opt/tlxlibs:

- jvcwrap.jar
- jvalccyy.jar

Restart the running application ./ITX stop and then ./ITX start.

For the Docker environments,

- docker cp jnodes0.jar <brand>-server:/opt/<company>/<brand>/libs/
- Restart the design server, that is, **docker restart** <brand>-server

Note: For RHEL, Docker is not supported. Podman can be used as a substitute since it provides a command line interface similar to Docker. Following is an example of using Podman.

- podman cp jnodes0.jar <brand>-server:/opt/<company>/<brand>/libs/
- Restart the design server, that is, **podman restart** <brand>-server

1. Import the cbpr\_translation.zip project into the Design Server.

2. Open the cbpr\_translation project in Design Server and build the following maps:

- cbpr2820\_mtn96\_framework
- cbpr2821\_mtn96\_camt029
- cbpr1502\_mttx\_setvarlog

3. Run the main map cbpr2820\_mtn96\_framework.

The following output files will be generated:

- mx\_out.xml  
File contains the translated CBPR+ camt.029.001.09 message from MT196 or MT296 SWIFT message.

- audit\_msg.json

Reports translation logs, including failure due to the pre-conversion checks as:

- If the input file is a SWIFT message other than MT196 or MT296 or not a SWIFT message.

---

## CBPR+ MT to MX (MT103\_COR/MT103\_STP to pacs.008.001.08 or MT103\_COR RETN to pacs.004.001.09) Translation Example

This example demonstrates the CBPR+ MT to MX (MT103\_COR/MT103\_STP to pacs.008.001.08 or MT103\_COR RETN to pacs.004.001.09) message translation using maps.

- **What the example contains**

Files included in this example are as follows:

- **How to run the example**

This mt-mx translation will use the sample files to demonstrate the generation of CBPR+ pacs.008.001.08/pacs.004.001.09 XML message output from a SWIFT MT103 or MT103\_STP or MT103\_COR RETN message.

---

## What the example contains

Files included in this example are as follows:

- cbpr\_translation.zip
  - Files:
    - MT103.inp
    - MT103\_RETN.inp
    - MT103\_STP.inp
  - Schemas:
    - head.001.001.02\_pacs\_004.xsd
    - head.001.001.02\_pacs\_008.xsd
    - pacs.004.001.09.xsd
    - pacs.008.001.08.xsd
    - swift\_iso7775\_ccyy.mtt
    - swiftroute\_funds.mtt
    - infoset.mtt
  - Maps:
    - cbpr2800\_mt103\_framework
    - cbpr2801\_mt103\_pacs008
    - cbpr2802\_mt103\_pacs008\_stp
    - cbpr2799\_mt103\_pacs004
    - cbpr1502\_mtnx\_setvarlog
- cbprJnodesConfigIBM.tar.gz

---

## How to run the example

This mt-mx translation will use the sample files to demonstrate the generation of CBPR+ pacs.008.001.08/pacs.004.001.09 XML message output from a SWIFT MT103 or MT103\_STP or MT103\_COR RETN message.

Extract from <packs\_install\_dir>/UIProjectImports/cbprJnodesConfigIBM.tar.gz file.

- jnodes0.jar

The following jars needs to be copied from <TX\_install\_dir>/jars:

- jackson-core-n.n.n.jar
- jackson-annotations-n.n.n.jar
- jackson-databind-n.n.n.jar

Or visit <https://repo1.maven.org/maven2/com/fasterxml/jackson/core/> for download.

Note: Recommend using the latest version.

For the non-Docker environments,

- Copy jars to <TX\_install\_dir>/extjar

For TX V11.0.1 and up, native based Design Server installation,

Copy the following into the directory defined in config.yaml server.persistence.libs, by default, this is set to /opt/txlibs:

- jvcwrap.jar
- jvalccyy.jar

Restart the running application ./ITX stop and then ./ITX start.

For the Docker environments,

- docker cp jnodes0.jar <brand>-server:/opt/<company>/<brand>/libs/
- Restart the design server, that is, **docker restart <brand>-server**

Note: For RHEL, Docker is not supported. Podman can be used as a substitute since it provides a command line interface similar to Docker. Following is an example of using Podman.

- podman cp jnodes0.jar <brand>-server:/opt/<company>/<brand>/libs/
  - Restart the design server, that is, **podman restart** <brand>-server
1. Import the cbpr\_translation.zip project into the Design Server.
  2. Open the cbpr\_translation project in Design Server and build the following maps:
    - cbpr2800\_mt103\_framework
    - cbpr2801\_mt103\_pacs008
    - cbpr2802\_mt103\_pacs008\_stp
    - cbpr2799\_mt103\_pacs004
    - cbpr1502\_mtmx\_setvarlog
  3. Run the main map cbpr2800\_mt103\_framework.
- The following output files will be generated:
- mx\_out.xml  
File contains the translated CBPR+ pacs.008.001.08/pacs.004.001.09 message from MT103\_COR or MT103\_STP/MT103\_COR RETN SWIFT message.
  - audit\_msg.json  
Reports translation logs, including failure due to the pre-conversion checks as:
    - If the input file is a SWIFT message other than MT103 or MT103\_STP or MT103 COR RETN or not a SWIFT message.

## CBPR+ MT to MX (MT103 SWIFT Go to pacs.008.001.08) Translation Example

This example demonstrates the CBPR+ MT to MX (MT103 SWIFT Go to pacs.008.001.08) message translation using maps.

- **What the example contains**  
Files included in this example are as follows:
- **How to run the example**  
This mt-mx translation will use the sample files to demonstrate the generation of CBPR+ pacs.008.001.08 XML message output from a MT103 SWIFT Go message.

### What the example contains

Files included in this example are as follows:

- cbpr\_translation.zip
  - Files:
    - MT103\_SGO.inp
  - Schemas:
    - head.001.001.02\_pacs\_008.xsd
    - pacs.008.001.08.xsd
    - swift\_iso7775\_ccyy.mtt
    - swiftroute\_funds.mtt
    - infoset.mtt
  - Maps:
    - cbpr2800\_mt103\_framework
    - cbpr\_t9n\_mt103\_pacs008\_sgo
    - cbpr1502\_mtmx\_setvarlog
- cbprJnodesConfigIBM.tar.gz

### How to run the example

This mt-mx translation will use the sample files to demonstrate the generation of CBPR+ pacs.008.001.08 XML message output from a MT103 SWIFT Go message.

Extract from <packs\_install\_dir>/UIProjectImports/cbprJnodesConfigIBM.tar.gz file.

- jnodes0.jar

The following jars needs to be copied from <TX\_install\_dir>/jars:

- jackson-core-n.n.n.jar
- jackson-annotations-n.n.n.jar
- jackson-databind-n.n.n.jar

Or visit <https://repo1.maven.org/maven2/com/fasterxml/jackson/core/> for download.

Note: Recommend using the latest version.

For the non-Docker environments,

- Copy jars to <TX\_install\_dir>/extjar

For TX V11.0.1 and up, native based Design Server installation,

Copy the following into the directory defined in config.yaml server.persistence.libs, by default, this is set to /opt/tlxlibs:

- jvcwrap.jar
- jvalccyy.jar

Restart the running application ./ITX stop and then ./ITX start.

For the Docker environments,

- docker cp jnodes0.jar <brand>-server:/opt/<company>/<brand>/libs/
- Restart the design server, that is, **docker restart** <brand>-server

Note: For RHEL, Docker is not supported. Podman can be used as a substitute since it provides a command line interface similar to Docker. Following is an example of using Podman.

- podman cp jnodes0.jar <brand>-server:/opt/<company>/<brand>/libs/
- Restart the design server, that is, **podman restart** <brand>-server

1. Import the cbpr\_translation.zip project into the Design Server.

2. Open the cbpr\_translation project in Design Server and build the following maps:

- cbpr2800\_mt103\_framework
- cbpr\_t9n\_mt103\_pacs008\_sgo
- cbpr1502\_mtnx\_setvarlog

3. Run the main map cbpr2800\_mt103\_framework.

The following output files will be generated:

- mx\_out.xml

File contains the translated CBPR+ pacs.008.001.08 message from MT103 SWIFT Go SWIFT message.

- audit\_msg.json

Reports translation logs, including failure due to the pre-conversion checks as:

- If the input file is a SWIFT message other than MT103 SWIFT Go or not a SWIFT message.
- IF Field 72, Line 1 starts with "/REJT/" OR "/RETN/".

## CBPR+ MT to MX (MT202 (core/adv/cove) to pacs.009.001.08 (core/adv/cove) or MT202 (RETN) to pacs.004.001.09) Translation Example

This example demonstrates the CBPR+ MT to MX (MT202 (core/adv/cove) to pacs.009.001.08 (core/adv/cove) or MT202 (RETN) to pacs.004.001.09) message translation using maps.

- **What the example contains**

Files included in this example are as follows:

- **How to run the example**

This mt-mx translation will use the sample files to demonstrate the generation of CBPR+ pacs.009.001.08 (core/adv/cove) XML message from a SWIFT MT202 (core/adv/cove) message or CBPR+ pacs.004.001.09 output from a MT202 RETN message.

## What the example contains

Files included in this example are as follows:

- cbpr\_translation.zip
  - Files:
    - MT202\_COR.inp
    - MT202\_COR\_bad.inp
    - MT202\_ADV.inp
    - MT202\_ADV\_bad.inp
    - MT202\_COV.inp
    - MT202\_COV\_bad.inp
    - MT202\_RTN.inp
    - trx\_config.xml
  - Schemas:
    - head.001.001.02\_pacs\_004.xsd
    - head.001.001.02\_pacs\_009\_adv.xsd
    - head.001.001.02\_pacs\_009\_cov.xsd
    - head.001.001.02\_pacs\_009.xsd
    - pacs.004.001.09.xsd
    - pacs.009.001.08\_adv.xsd
    - pacs.009.001.08\_cov.xsd
    - pacs.009.001.08.xsd
    - trx\_config.xsd
    - infoset.mtt
    - swift\_iso7775\_ccyy.mtt
    - swiftroute\_funds.mtt
  - Maps:
    - cbpr2810\_mt202\_framework
    - cbpr1502\_mtnx\_setvarlog
    - cbpr2814\_mt202\_pacs009\_adv
    - cbpr2815\_mt202\_pacs009\_cov
    - cbpr2813\_mt202\_pacs009
    - cbpr2824\_mt202RTN\_pacs004
- cbprJnodesConfigIBM.tar.gz

## How to run the example

This mt-mx translation will use the sample files to demonstrate the generation of CBPR+ pacs.009.001.08 (core/adv/cove) XML message from a SWIFT MT202 (core/adv/cove) message or CBPR+ pacs.004.001.09 output from a MT202 RETN message.

Extract from <packs\_install\_dir>/UIProjectImports/cbprJnodesConfigIBM.tar.gz file.

- jnodes0.jar

The following jars needs to be copied from <TX\_install\_dir>/jars:

- jackson-core-n.n.n.jar
- jackson-annotations-n.n.n.jar
- jackson-databind-n.n.n.jar

Or visit <https://repo1.maven.org/maven2/com/fasterxml/jackson/core/> for download.

Note: Recommend using the latest version.

For the non-Docker environments,

- Copy jars to <TX\_install\_dir>/extjar

For TX V11.0.1 and up, native based Design Server installation,

Copy the following into the directory defined in config.yaml server.persistence.libs, by default, this is set to /opt/tlibs:

- jvcwrap.jar
- jvalccyy.jar

Restart the running application ./ITX stop and then ./ITX start.

For the Docker environments,

- docker cp jnodes0.jar <brand>-server:/opt/<company>/<brand>/libs/
- Restart the design server, that is, **docker restart** <brand>-server

Note: For RHEL, Docker is not supported. Podman can be used as a substitute since it provides a command line interface similar to Docker. Following is an example of using Podman.

- podman cp jnodes0.jar <brand>-server:/opt/<company>/<brand>/libs/
- Restart the design server, that is, **podman restart** <brand>-server

1. Import the cbpr\_translation.zip project into the Design Server.

2. Open the cbpr\_translation project in Design Server and build the following maps:

- cbpr2810\_mt202\_framework
- cbpr1502\_mtmx\_setvarlog
- cbpr2814\_mt202\_pacs009\_adv
- cbpr2815\_mt202\_pacs009\_cov
- cbpr2813\_mt202\_pacs009
- cbpr2824\_mt202RTN\_pacs004

3. Run the main map cbpr2810\_mt202\_framework.

The following output files will be generated:

- mx\_out.xml  
File contains the translated CBPR+ pacs.009.001.08 (core/adv/cove) XML message from SWIFT MT202 (core/adv/cove) message or pacs.004.001.09 XML message from SWIFT MT202 (RETN) message.

- audit\_msg.json

Reports translation logs, including failure due to the pre-conversion checks as:

- Field 72, Line[1] should not start with /REJT/.
- If Field 72, Line[1] starts with /RETN/, <Tag121><EndToEndTxnRef> should not be absent.

## CBPR+ MT to MX (MT205 (core/cove) to pacs.009.001.08 (core/cove) or MT205 (RETN) to pacs.004.001.09) Translation Example

This example demonstrates the CBPR+ MT to MX (MT205 (core/cove) to pacs.009.001.08 (core/cove) or MT205 (RETN) to pacs.004.001.09) message translation using maps.

- **What the example contains**

Files included in this example are as follows:

- **How to run the example**

This mt-mx translation will use the sample files to demonstrate the generation of CBPR+ pacs.009.001.08 (core/cove) XML message from a SWIFT MT205 (core/cove) message or CBPR+ pacs.004.001.09 output from a MT205 (RETN) message.

## What the example contains

Files included in this example are as follows:

- cbpr\_translation.zip
  - Files:
    - MT205\_COR.inp
    - MT205\_COR\_bad.inp
    - MT205\_COV.inp
    - MT205\_COV\_bad.inp
    - MT205\_RTN.inp
    - trx\_config.xml
  - Schemas:
    - head.001.001.02\_pacs\_004.xsd
    - head.001.001.02\_pacs\_009\_adv.xsd
    - head.001.001.02\_pacs\_009\_cov.xsd
    - head.001.001.02\_pacs\_009.xsd
    - pacs.004.001.09.xsd
    - pacs.009.001.08\_adv.xsd
    - pacs.009.001.08\_cov.xsd
    - pacs.009.001.08.xsd
    - trx\_config.xsd
    - infoSet.mtt
    - swift\_iso7775\_ccyy.mtt
    - swiftRoute\_funds.mtt
  - Maps:
    - cbpr2810\_mt205\_framework
    - cbpr1502\_mtmx\_setvarlog
    - cbpr2820\_mt205\_pacs009\_cov
    - cbpr2819\_mt205\_pacs009
    - cbpr2825\_mt205RTN\_pacs004
- cbprJnodesConfigIBM.tar.gz

## How to run the example

This mt-mx translation will use the sample files to demonstrate the generation of CBPR+ pacs.009.001.08 (core/cove) XML message from a SWIFT MT205 (core/cove) message or CBPR+ pacs.004.001.09 output from a MT205 (RTN) message.

Extract from <packs\_install\_dir>/UIProjectImports/cbprJnodesConfigIBM.tar.gz file.

- jnodes0.jar

The following jars needs to be copied from <TX\_install\_dir>/jars:

- jackson-core-n.n.n.jar
- jackson-annotations-n.n.n.jar
- jackson-databind-n.n.n.jar

Or visit <https://repo1.maven.org/maven2/com/fasterxml/jackson/core/> for download.

Note: Recommend using the latest version.

For the non-Docker environments,

- Copy jars to <TX\_install\_dir>/extjar

For TX V11.0.1 and up, native based Design Server installation,

Copy the following into the directory defined in config.yaml server.persistence.libs, by default, this is set to /opt/txls:

- jvcwrap.jar
- jvalccyy.jar

Restart the running application ./ITX stop and then ./ITX start.

For the Docker environments,

- docker cp jnodes0.jar <brand>-server:/opt/<company>/<brand>/libs/
- Restart the design server, that is, **docker restart** <brand>-server

Note: For RHEL, Docker is not supported. Podman can be used as a substitute since it provides a command line interface similar to Docker. Following is an example of using Podman.

- podman cp jnodes0.jar <brand>-server:/opt/<company>/<brand>/libs/
- Restart the design server, that is, **podman restart** <brand>-server

1. Import the cbpr\_translation.zip project into the Design Server.
2. Open the cbpr\_translation project in Design Server and build the following maps:

- cbpr2811\_mt205\_framework
- cbpr1502\_mtmx\_setvarlog
- cbpr2820\_mt205\_pacs009\_cov
- cbpr2819\_mt205\_pacs009
- cbpr2825\_mt205RTN\_pacs004

3. Run the main map cbpr2811\_mt205\_framework.

The following output files will be generated:

- mx\_out.xml

File contains the translated CBPR+ pacs.009.001.08 (core/cove) XML message from SWIFT MT205 (core/cove) message or pacs.004.001.09 XML message from SWIFT MT205 (RETN) message.

- audit\_msg.json  
Reports translation logs, including failure due to the pre-conversion checks as:
  - Field 72, Line[1] should not start with /REJT/ OR /RETN/.
  - Field 72, Line[1] should not start with /REJT/.
  - If Field 72, Line[1] starts with /RETN/, <Tag121><EndToEndTxnRef> should not be absent.

---

## CBPR+ MT to MX Translation (using TX maps) Example

This example shows how TX maps can be used for translating CBPR+ MT to MX message formats.

- [What the example contains](#)  
Files included in this example are as follows:
- [Run the example in Design Studio](#)
- [Run the example in Design Server User Interface](#)

---

### What the example contains

Files included in this example are as follows:

- Maps:  
The maps directory contains the following map source:
  - cbpr\_t9n\_mt\_frmwrk\_maps.mms
    - cbpr1500\_mt9n0\_framework  
Main map to transform MT900/MT910 into camt.054.001.08
    - cbpr2800\_mt103\_framework  
Main map to transform MT103 COR/STP into pacs.008.001.08 COR or MT103 COR RETN to pacs.004.001.09
    - cbpr2808\_mt103sgo\_framework  
Main map to transform MT103 SGO into pacs.008.001.08 SGO
    - cbpr2810\_mt202\_framework  
Main map to transform MT202 COR/COV into pacs.009.001.08 ADV/COR/COV or MT202 RETN into pacs.004.001.09
    - cbpr2811\_mt205\_framework  
Main map to transform MT205 COR/COV into pacs.009.001.08 COR/COV or MT205 RETN into pacs.004.001.09
    - cbpr2820\_mtn96\_framework  
Main map to transform MT196/296 into camt.029.001.09
    - cbpr2840\_mtn92\_framework  
Main map to transform MT192/292 into camt.056.001.08
  - cbpr\_t9n\_mt\_setvarlog.mms  
Utility used to report translation logs.
  - cbpr\_t9n\_mt103\_pacs004.mms  
Utility used to populate pacs.004.001.09
  - cbpr\_t9n\_mt103\_pacs008.mms  
Utility used to populate pacs.008.001.08
  - cbpr\_t9n\_mt103\_pacs008\_sgo.mms  
Utility used to populate pacs.008.001.08 SGO
  - cbpr\_t9n\_mt103\_pacs008\_stp.mms  
Utility used to populate pacs.008.001.08 STP
  - cbpr\_t9n\_mt202\_pacs009\_adv.mms  
Utility used to populate pacs.009.001.08 ADV
  - cbpr\_t9n\_mt20n\_pacs009.mms  
Utility used to populate pacs.009.001.08 COR
  - cbpr\_t9n\_mt20n\_pacs009\_cov.mms  
Utility used to populate pacs.009.001.08 COV
  - cbpr\_t9n\_mt20nRTN\_pacs004.mms  
Utility used to populate pacs.004.001.09
  - cbpr\_t9n\_mt9n0\_camt054.mms  
Utility used to populate camt.054.001.08
  - cbpr\_t9n\_mtn92\_camt056.mms  
Utility used to populate camt.056.001.08

- cbpr\_t9n\_mtn96\_camt029.mms  
Utility used to populate camt.029.001.09

- Schemas:

The schemas directory contains the following files downloaded from SWIFT MyStandards MT/ISO 20022 Translation portal:

- camt.029.001.09.xsd
- camt.054.001.08.xsd
- camt.056.001.08.xsd
- head.001.001.02\_camt\_029.xsd
- head.001.001.02\_camt\_054.xsd
- head.001.001.02\_camt\_056.xsd
- head.001.001.02\_pacs\_004.xsd
- head.001.001.02\_pacs\_008.xsd
- head.001.001.02\_pacs\_008\_stp.xsd
- head.001.001.02\_pacs\_009.xsd
- head.001.001.02\_pacs\_009\_adv.xsd
- head.001.001.02\_pacs\_009\_cov.xsd
- pacs.004.001.09.xsd
- pacs.008.001.08.xsd
- pacs.008.001.08\_stp.xsd
- pacs.009.001.08.xsd
- pacs.009.001.08\_adv.xsd
- pacs.009.001.08\_cov.xsd
- trx\_config.xsd

Schema for the translation configuration file.

Each will have corresponding .mtx files used internally to manage RESTART attributes.

- Trees:

The trees directory contains the following files:

- infoset.mtt  
Metadata used to represent input messages into JSON.
- swift\_iso7775\_ccyy.mtt  
Metadata supporting SWIFT MT ISO7775 FIN structures.
- swiftroute\_funds.mtt  
Multi-purpose metadata structures.

- Data:

The data directory contains the following file:

- Sample SWIFT MT messages:
  - MT103.inp
  - MT103\_RETN.inp
  - MT103\_SGO.inp
  - MT103\_STP.inp
  - MT103\_STP\_RETN.inp
  - MT192.inp
  - MT196.inp
  - MT196\_bad.inp
  - MT202\_ADV.inp
  - MT202\_ADV\_bad.inp
  - MT202\_COR.inp
  - MT202\_COR\_bad.inp
  - MT202\_COV.inp
  - MT202\_RTN.inp
  - MT205\_COR.inp
  - MT205\_COR\_bad.inp
  - MT205\_COV.inp
  - MT205\_COV\_bad.inp
  - MT205\_RTN.inp
  - MT292.inp
  - MT296.inp
  - MT296\_bad.inp
  - MT900.inp
  - MT910.inp
  - trx\_config.xml

Translation configuration file.

## Run the example in Design Studio

Extract from <packs\_install\_dir>/UIProjectImports/cbprJnodesConfigIBM.tar.gz file.

- jnodes0.jar

The following jars needs to be copied from <TX\_install\_dir>/jars:

- jackson-core-n.n.n.jar
- jackson-annotations-n.n.n.jar
- jackson-databind-n.n.n.jar

Or visit <https://repo1.maven.org/maven2/com/fasterxml/jackson/core/> for download.

Note: Recommend using the latest version.

For the non-Docker environments,

- Copy jars to <TX\_install\_dir>/extjar

For TX V11.0.1 and up, native based Design Server installation,

Copy the following into the directory defined in config.yaml server.persistence.libs, by default, this is set to /opt/tlxlibs:

- jvcwrap.jar
- jvalccyy.jar

Restart the running application ./ITX stop and then ./ITX start.

For the Docker environments,

- docker cp jnodes0.jar <brand>-server:/opt/<company>/<brand>/libs/
- Restart the design server, that is, **docker restart** <brand>-server

Note: For RHEL, Docker is not supported. Podman can be used as a substitute since it provides a command line interface similar to Docker. Following is an example of using Podman.

- podman cp jnodes0.jar <brand>-server:/opt/<company>/<brand>/libs/
- Restart the design server, that is, **podman restart** <brand>-server

1. Build all the maps in the .mms files listed in What the example contains section.

2. Replace the input card 1 in any of the main maps (under cbpr\_t9n\_mt\_frmwrk\_maps.mms file) with desired input file.

3. Run any of the main router map:

- cbpr1500\_mt9n0\_framework (MT900/MT910 into camt.054.001.08)
- cbpr2800\_mt103\_framework (MT103 COR/STP into pacs.008.001.08 COR)
- cbpr2808\_mt103sgo\_framework (MT103 SGO into pacs.008.001.08 SGO)
- cbpr2810\_mt202\_framework (MT202 COR/COV/ADV into pacs.009.001.08 COR/COV or MT202 RETN into pacs.004.001.09)
- cbpr2811\_mt205\_framework (MT205 COR/COV into pacs.009.001.08 COR/COV or MT205 RETN into pacs.004.001.09)
- cbpr2820\_mtn96\_framework (MT196/MT296 into camt.029.001.09)
- cbpr2840\_mtn92\_framework (MT192/MT292 into camt.056.001.08)

You will see the output file in the data folder called mx\_out.xml and translation logs in the data folder called audit\_msg.json.

## Run the example in Design Server User Interface

Extract from <packs\_install\_dir>/UIProjectImports/cbprJnodesConfigIBM.tar.gz file.

- jnodes0.jar

The following jars needs to be copied from <TX\_install\_dir>/jars:

- jackson-core-n.n.n.jar
- jackson-annotations-n.n.n.jar
- jackson-databind-n.n.n.jar

Or visit <https://repo1.maven.org/maven2/com/fasterxml/jackson/core/> for download.

Note: Recommend using the latest version.

For the non-Docker environments,

- Copy jars to <TX\_install\_dir>/extjar

For TX V11.0.1 and up, native based Design Server installation,

Copy the following into the directory defined in config.yaml server.persistence.libs, by default, this is set to /opt/tlxlibs:

- jvcwrap.jar
- jvalccyy.jar

Restart the running application ./ITX stop and then ./ITX start.

For the Docker environments,

- docker cp jnodes0.jar <brand>-server:/opt/<company>/<brand>/libs/
- Restart the design server, that is, **docker restart** <brand>-server

Note: For RHEL, Docker is not supported. Podman can be used as a substitute since it provides a command line interface similar to Docker. Following is an example of using Podman.

- podman cp jnodes0.jar <brand>-server:/opt/<company>/<brand>/libs/
- Restart the design server, that is, **podman restart** <brand>-server

1. Open a browser pointing to the installation of the IBM Sterling Transformation Extender.
2. Enter user and password credentials.
3. If the project does not exist, import cbpr\_translation.zip.

4. If absent, create a package containing all the Maps, Files, and Flows.
5. Build package in desired server (to build all maps).
6. Open project.
7. In the Maps tab, open any of the main maps:
  - cbpr1500\_mt9n0\_framework (MT900/MT910 into camt.054.001.08)
  - cbpr2800\_mt103\_framework (MT103 COR/STP into pacs.008.001.08 COR)
  - cbpr2808\_mt103sgo\_framework (MT103 SGO into pacs.008.001.08 SGO)
  - cbpr2810\_mt202\_framework (MT202 COR/COV/ADV into pacs.009.001.08 COR/COV or MT202 RETN into pacs.004.001.09)
  - cbpr2811\_mt205\_framework (MT205 COR/COV into pacs.009.001.08 COR/COV or MT205 RETN into pacs.004.001.09)
  - cbpr2820\_mtn96\_framework (MT196/MT296 into camt.029.001.09)
  - cbpr2840\_mtn92\_framework (MT192/MT292 into camt.056.001.08)
8. Modify the settings for input card 1 on design canvas to point to the desired test file.
9. Save, build, and run the map from Step 8.
10. Right click on the output card 3 on canvas and select View translation results.
11. Right click on the output card 4 on canvas and select View translation logs.

## CBPR+ MX to MT Translation (using TX maps) Example

This example shows how TX maps can be used for translating CBPR+ MX to MT message formats.

- [What the example contains](#)
- Files included in this example are as follows:
- [Run the example in Design Studio](#)
- [Run the example in Design Server User Interface](#)

### What the example contains

Files included in this example are as follows:

- Maps:  
The maps directory contains the following map source:
  - cbpr\_t9n\_mx\_frmwrk\_maps.mms
    - cbpr2510\_camt054\_framework  
Main map to transform camt.054.001.08 into MT900/MT910.
    - cbpr2520\_camt029\_framework  
Main map to transform camt.029.001.09 into MT196/MT296.
    - cbpr2530\_camt053\_framework  
Main map to transform camt.053.001.08 into MT940.
    - cbpr2540\_camt056\_framework  
Main map to transform camt.056.001.08 into MT192/MT292.
    - cbpr2546\_camt052\_framework  
Main map to transform camt.052.001.08 into MT942.
    - cbpr2600\_camt057\_framework  
Main map to transform camt.057.001.06 into MT210.
    - cbpr2610\_camt058\_framework  
Main map to transform camt.058.001.08 into MT292.
    - cbpr2620\_camt107\_framework  
Main map to transform camt.107.001.01 into MT110.
    - cbpr2630\_camt108\_framework  
Main map to transform camt.108.001.01 into MT111.
    - cbpr2640\_camt109\_framework  
Main map to transform camt.109.001.01 into MT112.
    - cbpr2500\_pacs002\_framework  
Main map to transform pacs.002.001.10 into MT199/MT299.
    - cbpr2700\_pacs008\_framework  
Main map to transform pacs.008.001.08 COR/SGO into MT103 COR/STP/SGO.
    - cbpr2710\_pacs009\_mt202\_framework  
Main map to transform pacs.009.001.08 ADV/COR/COV into MT202 COR/COV.
    - cbpr2724\_pacs009\_mt205\_framework  
Main map to transform pacs.009.001.08 COR/COV into MT205 COR/COV.
    - cbpr2850\_pacs004\_mtnnn\_RTN\_frw  
Main map to transform pacs.004.001.09 into MT103RETN/MT202RETN/MT205RETN.
  - cbpr\_t9n\_mx\_setvarlog.mms

Utility used to report translation logs.

- cbpr\_t9n\_mx\_camt029\_mtn96.mms  
Utility used to populate MT196/MT296.
- cbpr\_t9n\_mx\_camt029\_translate.mms  
Utility used to normalize camt.029.001.09 into JSON.
- cbpr\_t9n\_mx\_camt052\_mt942.mms  
Utility used to populate MT942.
- cbpr\_t9n\_mx\_camt052\_translate.mms  
Utility used to normalize camt.052.001.08 into JSON.
- cbpr\_t9n\_mx\_camt053\_mt940.mms  
Utility used to populate MT940.
- cbpr\_t9n\_mx\_camt053\_translate.mms  
Utility used to normalize camt.053.001.08 into JSON.
- cbpr\_t9n\_mx\_camt054\_mt900.mms  
Utility used to populate MT900.
- cbpr\_t9n\_mx\_camt054\_mt910.mms  
Utility used to populate MT910.
- cbpr\_t9n\_mx\_camt054\_translate.mms  
Utility used to normalize camt.054.001.08 into JSON.
- cbpr\_t9n\_mx\_camt056\_mtn92.mms  
Utility used to populate MT192/MT292.
- cbpr\_t9n\_mx\_camt056\_translate.mms  
Utility used to normalize camt.056.001.08 into JSON.
- cbpr\_t9n\_mx\_camt057\_mt210.mms  
Utility used to populate MT210.
- cbpr\_t9n\_mx\_camt057\_translate.mms  
Utility used to normalize camt.057.001.06 into JSON.
- cbpr\_t9n\_mx\_camt058\_mt292.mms  
Utility used to populate MT292.
- cbpr\_t9n\_mx\_camt058\_translate.mms  
Utility used to normalize camt.058.001.08 into JSON.
- cbpr\_t9n\_mx\_camt107\_mt110.mms  
Utility used to populate MT110.
- cbpr\_t9n\_mx\_camt107\_translate.mms  
Utility used to normalize camt.107.001.01 into JSON.
- cbpr\_t9n\_mx\_camt108\_mt111.mms  
Utility used to populate MT111.
- cbpr\_t9n\_mx\_camt108\_translate.mms  
Utility used to normalize camt.108.001.01 into JSON.
- cbpr\_t9n\_mx\_camt109\_mt112.mms  
Utility used to populate MT112.
- cbpr\_t9n\_mx\_camt109\_translate.mms  
Utility used to normalize camt.109.001.01 into JSON.
- cbpr\_t9n\_mx\_pacs002\_mtn99.mms  
Utility used to populate MT199/MT299.
- cbpr\_t9n\_mx\_pacs002\_translate.mms  
Utility used to normalize pacs.002.001.10 into JSON.
- cbpr\_t9n\_mx\_pacs004\_mt103RTN.mms  
Utility used to populate MT103RTN.
- cbpr\_t9n\_mx\_pacs004\_mt20nRTN.mms  
Utility used to populate MT202RTN/MT205RTN.
- cbpr\_t9n\_mx\_pacs004\_translate.mms  
Utility used to normalize pacs.004.001.09 into JSON.
- cbpr\_t9n\_mx\_pacs008\_mt103.mms  
Utility used to populate MT103 COR/STP.
- cbpr\_t9n\_mx\_pacs008\_mt103\_sgo.mms  
Utility used to populate MT103 SGO.
- cbpr\_t9n\_mx\_pacs008\_stp\_translate.mms

Utility used to normalize pacs.008.001.08 STP into JSON.

- cbpr\_t9n\_mx\_pacs008\_translate.mms  
Utility used to normalize pacs.008.001.08 into JSON.
- cbpr\_t9n\_mx\_pacs009\_adv\_translate.mms  
Utility used to normalize pacs.009.001.08 ADV into JSON.
- cbpr\_t9n\_mx\_pacs009\_cov\_translate.mms  
Utility used to normalize pacs.009.001.08 COV into JSON.
- cbpr\_t9n\_mx\_pacs009\_mt20n.mms  
Utility used to populate MT202 COR/COV/ADV.
- cbpr\_t9n\_mx\_pacs009\_translate.mms  
Utility used to normalize pacs.009.001.08 into JSON.

- Schemas:

The schemas directory contains the following files downloaded from SWIFT MyStandards MT/ISO 20022 Translation portal:

- camt.029.001.09.xsd
- camt.052.001.08.xsd
- camt.053.001.08.xsd
- camt.054.001.08.xsd
- camt.056.001.08.xsd
- camt.057.001.06.xsd
- camt.058.001.08.xsd
- camt.107.001.01.xsd
- camt.108.001.01.xsd
- camt.109.001.01.xsd
- head.001.001.02\_camt\_029.xsd
- head.001.001.02\_camt\_052.xsd
- head.001.001.02\_camt\_053.xsd
- head.001.001.02\_camt\_054.xsd
- head.001.001.02\_camt\_056.xsd
- head.001.001.02\_camt\_057.xsd
- head.001.001.02\_camt\_058.xsd
- head.001.001.02\_camt\_107.xsd
- head.001.001.02\_camt\_108.xsd
- head.001.001.02\_camt\_109.xsd
- head.001.001.02\_pacs\_002.xsd
- head.001.001.02\_pacs\_004.xsd
- head.001.001.02\_pacs\_008.xsd
- head.001.001.02\_pacs\_008\_stp.xsd
- head.001.001.02\_pacs\_009.xsd
- head.001.001.02\_pacs\_009\_adv.xsd
- head.001.001.02\_pacs\_009\_cov.xsd
- pacs.002.001.10.xsd
- pacs.004.001.09.xsd
- pacs.008.001.08.xsd
- pacs.008.001.08\_stp.xsd
- pacs.009.001.08.xsd
- pacs.009.001.08\_adv.xsd
- pacs.009.001.08\_cov.xsd
- trx\_config.xsd  
Schema for the translation configuration file.

Each will have corresponding .mtx files used internally to manage RESTART attributes.

- Trees:

The trees directory contains the following files:

- infoset.mtt  
Metadata used to represent input messages into JSON.
- swift\_iso7775\_ccyy.mtt  
Metadata supporting SWIFT MT ISO7775 FIN structures.
- swiftroute\_funds.mtt  
Multi-purpose metadata structures.

- Data:

The data directory contains the following file:

- Sample CBPR+ MX messages:
  - env\_camt\_029\_mt196\_valid.xml
  - env\_camt\_029\_mt296\_valid.xml
  - env\_camt\_052\_mt942\_valid.xml
  - env\_camt\_052\_mt942\_bad.xml
  - env\_camt\_053\_mt940\_valid.xml

- env\_camt\_053\_mt940\_bad.xml
  - env\_camt\_054\_crdt.xml
  - env\_camt\_054\_dbit.xml
  - env\_camt\_054\_bad.xml
  - env\_camt\_056\_mt192\_valid.xml
  - env\_camt\_056\_mt292\_valid.xml
  - env\_camt\_057\_valid.xml
  - env\_camt\_057\_bad.xml
  - env\_camt\_058\_valid.xml
  - env\_camt\_058\_bad.xml
  - env\_camt\_107\_valid.xml
  - env\_camt\_108\_valid.xml
  - env\_camt\_109\_valid.xml
  - env\_pacs\_002\_199.xml
  - env\_pacs\_002\_299.xml
  - env\_pacs\_002\_bad.xml
  - env\_pacs\_004\_mt202rtn\_valid.xml
  - env\_pacs\_004\_mt205rtn\_valid.xml
  - env\_pacs\_004\_mt202rtn\_bad.xml
  - env\_pacs\_004\_mt205rtn\_bad.xml
  - env\_pacs\_004\_mt103rtn\_valid.xml
  - env\_pacs\_004\_mt103rtn\_bad.xml
  - env\_pacs\_008\_103.xml
  - env\_pacs\_008\_bad.xml
  - env\_pacs\_008\_STP.xml
  - env\_pacs\_008\_SGO.xml
  - env\_pacs\_009\_adv\_bad.xml
  - env\_pacs\_009\_adv\_valid.xml
  - env\_pacs\_009\_cor\_bad.xml
  - env\_pacs\_009\_mt202cor\_valid.xml
  - env\_pacs\_009\_cov\_bad.xml
  - env\_pacs\_009\_mt202cov\_valid.xml
  - env\_pacs\_009\_mt205cor\_valid.xml
  - env\_pacs\_009\_mt205cov\_valid.xml
  - trx\_config.xml
- Translation configuration file.

## Run the example in Design Studio

Extract from <packs\_install\_dir>/UIProjectImports/cbprJnodesConfigIBM.tar.gz file.

- jnodes0.jar

The following jars needs to be copied from <TX\_install\_dir>/jars:

- jackson-core-n.n.n.jar
- jackson-annotations-n.n.n.jar
- jackson-databind-n.n.n.jar

Or visit <https://repo1.maven.org/maven2/com/fasterxml/jackson/core/> for download.

Note: Recommend using the latest version.

For the non-Docker environments,

- Copy jars to <TX\_install\_dir>/extjar

For TX V11.0.1 and up, native based Design Server installation,

Copy the following into the directory defined in config.yaml server.persistence.libs, by default, this is set to /opt/txlbs:

- jvcwrap.jar
- jvalccyy.jar

Restart the running application ./ITX stop and then ./ITX start.

For the Docker environments,

- docker cp jnodes0.jar <brand>-server:/opt/<company>/<brand>/libs/
- Restart the design server, that is, **docker restart <brand>-server**

Note: For RHEL, Docker is not supported. Podman can be used as a substitute since it provides a command line interface similar to Docker. Following is an example of using Podman.

- podman cp jnodes0.jar <brand>-server:/opt/<company>/<brand>/libs/
- Restart the design server, that is, **podman restart <brand>-server**

1. Build all the maps in the .mms files listed in What the example contains section.

2. Replace the input card 1 in any of the main maps (under cbpr\_t9n\_mx\_frmwrk\_maps.mms) with desired input file.

3. Run any of the main router map:

- cbpr2510\_camt054\_framework (camt.054.001.08 into MT900/MT910)
- cbpr2520\_camt029\_framework (camt.029.001.09 into MT196/MT296)
- cbpr2530\_camt053\_framework (camt.053.001.08 into MT940)
- cbpr2540\_camt056\_framework (camt.056.001.08 into MT192/MT292)
- cbpr2546\_camt052\_framework (camt.052.001.08 into MT942)
- cbpr2600\_camt057\_framework (camt.057.001.06 into MT210)
- cbpr2610\_camt058\_framework (camt.058.001.08 into MT292)
- cbpr2620\_camt107\_framework (camt.107.001.01 into MT110)
- cbpr2630\_camt108\_framework (camt.108.001.01 into MT111)
- cbpr2640\_camt109\_framework (camt.109.001.01 into MT112)
- cbpr2500\_pacs002\_framework (pacs.002.001.10 into MT199/MT299)
- cbpr2700\_pacs008\_framework (pacs.008.001.08 COR/SGO into MT103 COR/STP/SGO)
- cbpr2710\_pacs009\_mt202\_framework (pacs.009.001.08 into MT202 COR/COV/ADV)
- cbpr2724\_pacs009\_mt205\_framework (pacs.009.001.08 into MT205 COR/COV)
- cbpr2850\_pacs004\_mtnnn\_RTN\_frw (pacs.004.001.09 into MT103RETN/MT202RETN/MT205RETN)

You will see the output file in the data folder called mt\_out.xml and translation logs in the data folder called audit\_msg.json.

## Run the example in Design Server User Interface

Extract from <packs\_install\_dir>/UIProjectImports/cbprJnodesConfigIBM.tar.gz file.

- jnodes0.jar

The following jars needs to be copied from <TX\_install\_dir>/jars:

- jackson-core-n.n.n.jar
- jackson-annotations-n.n.n.jar
- jackson-databind-n.n.n.jar

Or visit <https://repo1.maven.org/maven2/com/fasterxml/jackson/core/> for download.

Note: Recommend using the latest version.

For the non-Docker environments,

- Copy jars to <TX\_install\_dir>/extjar

For TX V11.0.1 and up, native based Design Server installation,

Copy the following into the directory defined in config.yaml server.persistence.libs, by default, this is set to /opt/txlbs:

- jvcwrap.jar
- jvalccyy.jar

Restart the running application ./ITX stop and then ./ITX start.

For the Docker environments,

- docker cp jnodes0.jar <brand>-server:/opt/<company>/<brand>/libs/
- Restart the design server, that is, **docker restart** <brand>-server

Note: For RHEL, Docker is not supported. Podman can be used as a substitute since it provides a command line interface similar to Docker. Following is an example of using Podman.

- podman cp jnodes0.jar <brand>-server:/opt/<company>/<brand>/libs/
  - Restart the design server, that is, **podman restart** <brand>-server
1. Open a browser pointing to the installation of the IBM Sterling Transformation Extender.
  2. Enter user and password credentials.
  3. If the project does not exist, import cbpr\_translation.zip.
  4. If absent, create a package containing all the Maps, Files, and Flows.
  5. Build package in desired server (to build all maps).
  6. Open project.
  7. In the Maps tab, open any of the main maps:
    - cbpr2510\_camt054\_framework (camt.054.001.08 into MT900/MT910)
    - cbpr2520\_camt029\_framework (camt.029.001.09 into MT196/MT296)
    - cbpr2530\_camt053\_framework (camt.053.001.08 into MT940)
    - cbpr2540\_camt056\_framework (camt.056.001.08 into MT192/MT292)
    - cbpr2545\_camt052\_framework (camt.052.001.08 into MT942)
    - cbpr2600\_camt057\_framework (camt.057.001.06 into MT210)
    - cbpr2610\_camt058\_framework (camt.058.001.08 into MT292)
    - cbpr2620\_camt107\_framework (camt.107.001.01 into MT110)
    - cbpr2630\_camt108\_framework (camt.108.001.01 into MT111)
    - cbpr2640\_camt109\_framework (camt.109.001.01 into MT112)
    - cbpr2500\_pacs002\_framework (pacs.002.001.10 into MT199/MT299)
    - cbpr2700\_pacs008\_framework (pacs.008.001.08 COR/SGO into MT103 COR/STP/SGO)
    - cbpr2710\_pacs009\_mt202\_framework (pacs.009.001.08 into MT202 COR/COV/ADV)
    - cbpr2724\_pacs009\_mt205\_framework (pacs.009.001.08 into MT205 COR/COV)
    - cbpr2850\_pacs004\_mtnnn\_RTN\_frw (pacs.004.001.09 into MT103RETN/MT202RETN/MT205RETN)
  8. Modify the settings for input card 1 on design canvas to point to the desired test file.
  9. Save, build, and run the map from Step 8.
  10. Right click on the output card 2 on canvas and select View translation results.
  11. Right click on the output card 3 on canvas and select View translation logs.

## CHAPS L4L/ENH Examples:

- **[CHAPS L4L Examples: \(Deprecated\)](#)**  
CHAPS L4L examples are as follow:
- **[CHAPS ENH Examples:](#)**  
CHAPS ENH examples are as follow:

## CHAPS L4L Examples: (Deprecated)

CHAPS L4L examples are as follow:

- **[CHAPS L4L enhanced MX validation \(using flows\) Example](#)**  
This example demonstrates the CHAPS L4L enhanced MX validation for CHAPS L4L messages using the flow server.
- **[CHAPS L4L enhanced MX validation \(using maps\) Example](#)**  
This example demonstrates the CHAPS L4L enhanced MX validation for CHAPS L4L messages using maps only.
- **[CHAPS L4L schema validation \(using maps\) Example](#)**  
This example demonstrates the CHAPS L4L schema validation for CHAPS L4L messages using maps only.

## CHAPS L4L enhanced MX validation (using flows) Example

This example demonstrates the CHAPS L4L enhanced MX validation for CHAPS L4L messages using the flow server.

The flow can perform following level of validation:

- MX extended validation includes following checks:
  - BIC lookup
  - Country code lookup
  - Currency code lookup
  - IBAN format
  - Allowed maximum fractional digits per currency
  - Usage guideline rules, see mxconfig.xml for list of rules
- Schema validation
- **[What the example contains](#)**  
Files included in this example are as follows:
- **[How to run the example](#)**  
This CHAPS L4L MX Validation will use the sample files to demonstrate the generation of validation report as output from a CHAPS L4L XML message.

- [How to customize the mxconfig.xml file](#)

Rules enforced by the MX validation framework can be customized by editing the mxconfig.xml located under the validation/mx\_extended/data folder.

---

## What the example contains

Files included in this example are as follows:

- chaps\_l4l.zip
  - Test Data Files:
    - chlf\_pacs\_009\_cov.xml
    - chlf\_pacs\_009\_cov\_bad\_rule.xml
    - chlf\_pacs\_009\_cov\_bad\_schema.xml
  - Validation Files:
    - mxconfig.xml
    - bic.xml
    - currencycodedecimals.xml
  - Schemas:
    - bic.xsd  
Metadata that represents the bic.xml repository file structure.
    - ccy.xsd  
Metadata that represents the currencycodedecimals.xml repository file structure.
    - mxconfig.xsd  
Metadata that represents the mxconfig.xml configuration file structure.
    - mxvalErrorReport.mtt  
Metadata that represents the xml based structure of the validation report.
    - swiftroute\_funds.mtt  
Metadata that is used as an internal element placeholder.
    - admi.004.001.02\_fqsm.xsd
    - admi.004.001.02\_rsr.xsd
    - camt.052.001.08.xsd
    - camt.053.001.08.xsd
    - camt.054.001.08\_ca.xsd
    - camt.054.001.08\_cld.xsd
    - camt.054.001.08\_cli.xsd
    - camt.054.001.08\_lpa.xsd
    - camt.060.001.05.xsd
    - mirs.095.001.01.xsd
    - pacs.002.001.10.xsd
    - pacs.004.001.09.xsd
    - pacs.008.001.08.xsd
    - pacs.009.001.08\_core.xsd
    - pacs.009.001.08\_cov.xml
    - head.001.001.02\_admi\_004\_fqsm.xsd
    - head.001.001.02\_admi\_004\_rsr.xsd
    - head.001.001.02\_camt\_052.xsd
    - head.001.001.02\_camt\_053.xsd
    - head.001.001.02\_camt\_054\_ca.xsd
    - head.001.001.02\_camt\_054\_cld.xsd
    - head.001.001.02\_camt\_054\_cli.xsd
    - head.001.001.02\_camt\_054\_lpa.xsd
    - head.001.001.02\_camt\_060.xsd
    - head.001.001.02\_mirs\_095.xsd
    - head.001.001.02\_pacs\_002.xsd
    - head.001.001.02\_pacs\_004.xsd
    - head.001.001.02\_pacs\_008.xsd
    - head.001.001.02\_pacs\_009\_core.xsd
    - head.001.001.02\_pacs\_009\_cov.xml

Note: XML schemas were downloaded from SWIFT MyStandards Readiness CHAPS L4L portal.

- Maps:
  - For Extended Validation:
    - chlf8000\_val
    - chlf8100\_val
    - chlf8056\_head112\_camt052
    - chlf8006\_camt\_052\_001\_08
    - chlf8055\_head112\_camt053
    - chlf8005\_camt\_053\_001\_08
    - chlf8051\_head112\_camt054\_ca
    - chlf8001\_camt\_054\_001\_08\_ca
    - chlf8052\_head112\_camt054\_cld
    - chlf8002\_camt\_054\_001\_08\_cld
    - chlf8053\_head112\_camt054\_cli
    - chlf8003\_camt\_054\_001\_08\_cli
    - chlf8054\_head112\_camt054\_lpa

- chlf8004\_camt\_054\_001\_08\_lpa
- chlf8060\_head112\_camt060
- chlf8009\_camt\_060\_001\_05
- chlf8057\_head112\_pacs\_002
- chlf8007\_pacs\_002\_001\_10
- chlf8058\_head112\_pacs\_008
- chlf8008\_pacs\_008\_001\_08
- chlf8059\_head112\_pacs\_004
- chlf8010\_pacs\_004\_001\_09
- chlf8061\_head112\_pacs\_009\_core
- chlf8011\_pacs\_009\_001\_08\_core
- chlf8062\_head112\_pacs\_009\_cov
- chlf8012\_pacs\_009\_001\_08\_cov
- chlf8401\_head112\_mirs095
- chlf8301\_mirs\_095\_001\_01
- chlf8601\_head112\_admi004\_fqsm
- chlf8501\_admi\_004\_001\_02\_fqsm
- chlf8602\_head112\_admi004\_rsr
- chlf8502\_admi\_004\_001\_02\_rsr
- For Schema Validation:
  - chlf8200\_val
  - chlf8801\_chaps\_l4l\_bah\_val
  - chlf8802\_chaps\_l4l\_doc\_val
  - chlf8201\_camt\_054\_001\_08\_ca
  - chlf8202\_camt\_054\_001\_08\_cld
  - chlf8203\_camt\_054\_001\_08\_cli
  - chlf8204\_camt\_054\_001\_08\_lpa
  - chlf8205\_camt\_053\_001\_08
  - chlf8206\_camt\_052\_001\_08
  - chlf8207\_pacs\_002\_001\_10
  - chlf8208\_pacs\_008\_001\_08
  - chlf8209\_camt\_060\_001\_05
  - chlf8210\_pacs\_004\_001\_09
  - chlf8211\_pacs\_009\_001\_08\_core
  - chlf8212\_pacs\_009\_001\_08\_cov
  - chlf8213\_mirs\_095\_001\_01
  - chlf8214\_admi\_004\_001\_02\_fqsm
  - chlf8215\_admi\_004\_001\_02\_rsr
  - chlf8251\_head112\_camt054\_ca
  - chlf8252\_head112\_camt054\_cld
  - chlf8253\_head112\_camt054\_cli
  - chlf8254\_head112\_camt054\_lpa
  - chlf8255\_head112\_camt053
  - chlf8256\_head112\_camt052
  - chlf8257\_head112\_pacs\_002\_001\_10
  - chlf8258\_head112\_pacs\_008\_001\_08
  - chlf8259\_head112\_camt060
  - chlf8260\_head112\_pacs004
  - chlf8261\_head112\_pacs009\_core
  - chlf8262\_head112\_pacs009\_cov
  - chlf8263\_head112\_mirs095
  - chlf8264\_head112\_admi004\_fqsm
  - chlf8265\_head112\_admi004\_rsr
- Common:
  - mxut1002\_bizsvc\_chaps
- Flows:
  - chaps\_l4l\_validation\_flow

## How to run the example

This CHAPS L4L MX Validation will use the sample files to demonstrate the generation of validation report as output from a CHAPS L4L XML message.

The stopValidation.json file will report the validation failure due to the pre-conversion checks:

**If input file is other than any of the supported CHAPS L4L messages as per schema list indicated above.**

1. Import the chaps\_l4l.zip project into the Design Server.
  2. Open the chaps\_l4l project in Design Server and view the flow chaps\_l4l\_validation\_flow.
- a. The Flow Description points to all the maps to be executed during the flow process, which will be called from within some of the map nodes in the flow. The following is a list of maps invoked using RUN built-in function during the flow process:

- @packagemap=chaps\_l4l/validation/mx\_extended/maps/chaps\_l4l\_mx\_camt\_validation\_enh/chlf8056\_head112\_camt052
- @packagemap=chaps\_l4l/validation/mx\_extended/maps/chaps\_l4l\_mx\_camt\_validation\_enh/chlf8006\_camt\_052\_001\_08
- @packagemap=chaps\_l4l/validation/mx\_extended/maps/chaps\_l4l\_mx\_camt\_validation\_enh/chlf8055\_head112\_camt053
- @packagemap=chaps\_l4l/validation/mx\_extended/maps/chaps\_l4l\_mx\_camt\_validation\_enh/chlf8005\_camt\_053\_001\_08
- @packagemap=chaps\_l4l/validation/mx\_extended/maps/chaps\_l4l\_mx\_camt\_validation\_enh/chlf8051\_head112\_camt054\_ca
- @packagemap=chaps\_l4l/validation/mx\_extended/maps/chaps\_l4l\_mx\_camt\_validation\_enh/chlf8001\_camt\_054\_001\_08\_ca
- @packagemap=chaps\_l4l/validation/mx\_extended/maps/chaps\_l4l\_mx\_camt\_validation\_enh/chlf8052\_head112\_camt054\_cld
- @packagemap=chaps\_l4l/validation/mx\_extended/maps/chaps\_l4l\_mx\_camt\_validation\_enh/chlf8002\_camt\_054\_001\_08\_cld

- @packagemap=chaps\_l4l/validation/mx\_extended/maps/chaps\_l4l\_mx\_camt\_validation\_enh/chlf8053\_head112\_camt054\_cli
- @packagemap=chaps\_l4l/validation/mx\_extended/maps/chaps\_l4l\_mx\_camt\_validation\_enh/chlf8003\_camt\_054\_001\_08\_cli
- @packagemap=chaps\_l4l/validation/mx\_extended/maps/chaps\_l4l\_mx\_camt\_validation\_enh/chlf8054\_head112\_camt054\_lpa
- @packagemap=chaps\_l4l/validation/mx\_extended/maps/chaps\_l4l\_mx\_camt\_validation\_enh/chlf8004\_camt\_054\_001\_08\_lpa
- @packagemap=chaps\_l4l/validation/mx\_extended/maps/chaps\_l4l\_mx\_camt\_validation\_enh/chlf8060\_head112\_camt060
- @packagemap=chaps\_l4l/validation/mx\_extended/maps/chaps\_l4l\_mx\_camt\_validation\_enh/chlf8009\_camt\_060\_001\_05
- @packagemap=chaps\_l4l/validation/mx\_extended/maps/chaps\_l4l\_mx\_pacs\_validation\_enh/chlf8057\_head112\_pacs\_002
- @packagemap=chaps\_l4l/validation/mx\_extended/maps/chaps\_l4l\_mx\_pacs\_validation\_enh/chlf8007\_pacs\_002\_001\_10
- @packagemap=chaps\_l4l/validation/mx\_extended/maps/chaps\_l4l\_mx\_pacs\_validation\_enh/chlf8058\_head112\_pacs\_008
- @packagemap=chaps\_l4l/validation/mx\_extended/maps/chaps\_l4l\_mx\_pacs\_validation\_enh/chlf8008\_pacs\_008\_001\_08
- @packagemap=chaps\_l4l/validation/mx\_extended/maps/chaps\_l4l\_mx\_pacs\_validation\_enh/chlf8059\_head112\_pacs\_004
- @packagemap=chaps\_l4l/validation/mx\_extended/maps/chaps\_l4l\_mx\_pacs\_validation\_enh/chlf8010\_pacs\_004\_001\_09
- @packagemap=chaps\_l4l/validation/mx\_extended/maps/chaps\_l4l\_mx\_pacs\_validation\_enh/chlf8061\_head112\_pacs\_009\_core
- @packagemap=chaps\_l4l/validation/mx\_extended/maps/chaps\_l4l\_mx\_pacs\_validation\_enh/chlf8011\_pacs\_009\_001\_08\_core
- @packagemap=chaps\_l4l/validation/mx\_extended/maps/chaps\_l4l\_mx\_pacs\_validation\_enh/chlf8062\_head112\_pacs\_009\_cov
- @packagemap=chaps\_l4l/validation/mx\_extended/maps/chaps\_l4l\_mx\_pacs\_validation\_enh/chlf8012\_pacs\_009\_001\_08\_cov
- @packagemap=chaps\_l4l/validation/mx\_extended/maps/chaps\_l4l\_mx\_mirs\_validation\_enh/chlf8401\_head112\_mirs095
- @packagemap=chaps\_l4l/validation/mx\_extended/maps/chaps\_l4l\_mx\_mirs\_validation\_enh/chlf8301\_mirs\_095\_001\_01
- @packagemap=chaps\_l4l/validation/mx\_extended/maps/chaps\_l4l\_mx\_admi\_validation\_enh/chlf8601\_head112\_admi004\_fqsm
- @packagemap=chaps\_l4l/validation/mx\_extended/maps/chaps\_l4l\_mx\_admi\_validation\_enh/chlf8501\_admi\_004\_001\_02\_fqsm
- @packagemap=chaps\_l4l/validation/mx\_extended/maps/chaps\_l4l\_mx\_admi\_validation\_enh/chlf8602\_head112\_admi004\_rsr
- @packagemap=chaps\_l4l/validation/mx\_extended/maps/chaps\_l4l\_mx\_admi\_validation\_enh/chlf8502\_admi\_004\_001\_02\_rsr
- @packagemap=chaps\_l4l/validation/schema\_only/maps/chaps\_l4l\_mx\_schema\_validation\_xsd/chlf8201\_camt\_054\_001\_08\_ca
- @packagemap=chaps\_l4l/validation/schema\_only/maps/chaps\_l4l\_mx\_schema\_validation\_xsd/chlf8202\_camt\_054\_001\_08\_cld
- @packagemap=chaps\_l4l/validation/schema\_only/maps/chaps\_l4l\_mx\_schema\_validation\_xsd/chlf8203\_camt\_054\_001\_08\_cli
- @packagemap=chaps\_l4l/validation/schema\_only/maps/chaps\_l4l\_mx\_schema\_validation\_xsd/chlf8204\_camt\_054\_001\_08\_lpa
- @packagemap=chaps\_l4l/validation/schema\_only/maps/chaps\_l4l\_mx\_schema\_validation\_xsd/chlf8205\_camt\_053\_001\_08
- @packagemap=chaps\_l4l/validation/schema\_only/maps/chaps\_l4l\_mx\_schema\_validation\_xsd/chlf8206\_camt\_052\_001\_08
- @packagemap=chaps\_l4l/validation/schema\_only/maps/chaps\_l4l\_mx\_schema\_validation\_xsd/chlf8207\_pacs\_002\_001\_10
- @packagemap=chaps\_l4l/validation/schema\_only/maps/chaps\_l4l\_mx\_schema\_validation\_xsd/chlf8208\_pacs\_008\_001\_08
- @packagemap=chaps\_l4l/validation/schema\_only/maps/chaps\_l4l\_mx\_schema\_validation\_xsd/chlf8209\_camt\_060\_001\_05
- @packagemap=chaps\_l4l/validation/schema\_only/maps/chaps\_l4l\_mx\_schema\_validation\_xsd/chlf8210\_pacs\_004\_001\_09
- @packagemap=chaps\_l4l/validation/schema\_only/maps/chaps\_l4l\_mx\_schema\_validation\_xsd/chlf8211\_pacs\_009\_001\_08\_core
- @packagemap=chaps\_l4l/validation/schema\_only/maps/chaps\_l4l\_mx\_schema\_validation\_xsd/chlf8212\_pacs\_009\_001\_08\_cov
- @packagemap=chaps\_l4l/validation/schema\_only/maps/chaps\_l4l\_mx\_schema\_validation\_xsd/chlf8213\_mirs\_095\_001\_01
- @packagemap=chaps\_l4l/validation/schema\_only/maps/chaps\_l4l\_mx\_schema\_validation\_xsd/chlf8214\_admi\_004\_001\_02\_fqsm
- @packagemap=chaps\_l4l/validation/schema\_only/maps/chaps\_l4l\_mx\_schema\_validation\_xsd/chlf8215\_admi\_004\_001\_02\_rsr
- @packagemap=chaps\_l4l/validation/schema\_only/maps/chaps\_l4l\_mx\_schema\_validation\_xsd/chlf8251\_head112\_camt054\_ca
- @packagemap=chaps\_l4l/validation/schema\_only/maps/chaps\_l4l\_mx\_schema\_validation\_xsd/chlf8252\_head112\_camt054\_cld
- @packagemap=chaps\_l4l/validation/schema\_only/maps/chaps\_l4l\_mx\_schema\_validation\_xsd/chlf8253\_head112\_camt054\_cli
- @packagemap=chaps\_l4l/validation/schema\_only/maps/chaps\_l4l\_mx\_schema\_validation\_xsd/chlf8254\_head112\_camt054\_lpa
- @packagemap=chaps\_l4l/validation/schema\_only/maps/chaps\_l4l\_mx\_schema\_validation\_xsd/chlf8255\_head112\_camt053
- @packagemap=chaps\_l4l/validation/schema\_only/maps/chaps\_l4l\_mx\_schema\_validation\_xsd/chlf8256\_head112\_camt052
- @packagemap=chaps\_l4l/validation/schema\_only/maps/chaps\_l4l\_mx\_schema\_validation\_xsd/chlf8257\_head112\_pacs\_002\_001\_10
- @packagemap=chaps\_l4l/validation/schema\_only/maps/chaps\_l4l\_mx\_schema\_validation\_xsd/chlf8258\_head112\_pacs\_008\_001\_08
- @packagemap=chaps\_l4l/validation/schema\_only/maps/chaps\_l4l\_mx\_schema\_validation\_xsd/chlf8259\_head112\_camt060
- @packagemap=chaps\_l4l/validation/schema\_only/maps/chaps\_l4l\_mx\_schema\_validation\_xsd/chlf8260\_head112\_pacs004
- @packagemap=chaps\_l4l/validation/schema\_only/maps/chaps\_l4l\_mx\_schema\_validation\_xsd/chlf8261\_head112\_pacs009\_core
- @packagemap=chaps\_l4l/validation/schema\_only/maps/chaps\_l4l\_mx\_schema\_validation\_xsd/chlf8262\_head112\_pacs009\_cov
- @packagemap=chaps\_l4l/validation/schema\_only/maps/chaps\_l4l\_mx\_schema\_validation\_xsd/chlf8263\_head112\_mirs095
- @packagemap=chaps\_l4l/validation/schema\_only/maps/chaps\_l4l\_mx\_schema\_validation\_xsd/chlf8264\_head112\_admi004\_fqsm
- @packagemap=chaps\_l4l/validation/schema\_only/maps/chaps\_l4l\_mx\_schema\_validation\_xsd/chlf8265\_head112\_admi004\_rsr

a. It utilizes the following nodes:

i. Source Nodes:

- mx\_input

This node identifies the input data to be validated in the flow. It uses the INPUT\_FILE variable to set the location of the data.

ii. Map Nodes:

- MX pre-check

Runs map mxut1002\_bizsvc\_chaps which sets flow variables, checks pre-validation conditions and creates infoset.json for validation.

- EXT\_VAL

Runs map chlf8100\_val which calls the appropriate map to perform extended validation of a CHAPS L4L message.

- XSD\_VAL

Runs map chlf8200\_val which calls the appropriate map to perform schema validation of a CHAPS L4L message.

Note: The maps chlf8100\_val and chlf8200\_val internally call all the other maps identified in the above Flow Description step.

iii. Decision Nodes:

- pre-valid chk

This node checks the flow variable stopValidation to decide if further validation/processing is not required.

- EXT\_RESULT\_FORMAT

This node checks the value of the flow variable REPORT\_FORMAT to determine if the resulting report for extended validation should be generated in XML (default) or JSON.

- XSD\_RESULT\_FORMAT

This node checks the value of the flow variable REPORT\_FORMAT to determine if the resulting report for schema-only validation should be generated in XML (default) or JSON.

iv. Route Nodes:

- **VAL\_TYPE**  
This node checks the variable VALIDATION\_TYPE to determine if the process will do extended validation or schema-only validation on the data.
- v. Log Nodes:
  - **FAILURE**  
In case of no validation due to precondition check failures, this node creates a log file specified by the flow variable FAILURE\_LOG.
- vi. Fail Node:
  - **STOP**  
It generates error message setup in the node in case of no validation performed.
- vii. Passthrough Nodes:
  - **EXT\_XML\_CONVERT**  
This node receives an extended validation report in XML format and does not modify the content, thus passing it as is to the appropriate target node.
  - **XSD\_XML\_CONVERT**  
This node receives a schema-only validation report in XML format and does not modify the content, thus passing it as is to the appropriate target node.
- viii. Format Converter Nodes:
  - **EXT\_JSON\_CONVERT**  
This node receives an extended validation report in XML format and converts it to JSON before passing it to the appropriate target node.
  - **XSD\_JSON\_CONVERT**  
This node receives a schema-only validation report in XML format and converts it to JSON before passing it to the appropriate target node.
- ix. Target Nodes:
  - **ext\_xml**  
This node contains the resulting extended validation report in XML format and creates the output as defined by the variable OUTPUT\_RESULT\_XML.
  - **ext\_json**  
This node contains the resulting extended validation report in JSON format and creates the output as defined by the variable OUTPUT\_RESULT\_JSON.
  - **xsd\_xml**  
This node contains the resulting schema-only validation report in XML format and creates the output as defined by the variable OUTPUT\_RESULT\_XML.
  - **xsd\_json**  
This node contains the resulting schema-only validation report in JSON format and creates the output as defined by the variable OUTPUT\_RESULT\_JSON.

b. It utilizes the following variables:

Flow variables:

- **VALIDATION\_TYPE**  
The default value is extended.  
  
For schema-only validation, it can be changed to schema. This is used in Route node VAL\_TYPE.
- **REPORT\_FORMAT**  
The default value is xml.  
  
It can be changed to json. This is used in Decision nodes EXT\_RESULT\_FORMAT and XSD\_RESULT\_FORMAT.
- **INPUT\_FILE**  
The default value is ..../tools/mx\_service/data/chlf\_pacs\_009\_cov.xml.  
  
This is the data file to be used for validation and can be customized. This is used in Source node mx\_input.
- **BIC\_FILE**  
The default value is ..../data/bic.xml.  
  
This is the location of the bic cross-reference file that is used in all the maps executed by the map in node EXT\_VAL. It can be customized.
- **CCY\_FILE**  
The default value is ..../data/currencycodedecimals.xml.  
  
This is the location of the country code cross-reference file that is used in all the maps executed by the map in node EXT\_VAL. It can be customized.
- **MXCONFIG\_FILE**  
The default value is ..../data/mxconfig.xml.  
  
This is the location of the file containing the rule validation settings that is used in all the maps executed by the map in node EXT\_VAL. It can be customized.
- **OUTPUT\_RESULT\_XML**  
The default value is validation\_result.xml.  
  
This is the location of the validation report file in xml format. It is used in Target nodes ext\_xml and xsd\_xml. It can be customized.
- **OUTPUT\_RESULT\_JSON**  
The default value is validation\_result.json.  
  
This is the location of the validation report file in json format. It is used in Target nodes ext\_json and xsd\_json. It can be customized.

- FAILURE\_LOG  
The default value is stopMXValidation.json.  
  
This is the location of the failure log. It is used in Log node FAILURE. It can be customized.
  - bizSvc  
For internal use in the Map node MX pre-check to determine the type of data being read in the input file.
  - stopValidation  
For internal use in the Map node MX pre-check and is checked in Decision node pre-valid chk to cause a Failure log in case a data file was not recognized as a valid CHAPS L4L message.
3. Open the main flow chaps\_l4l\_validation\_flow in the Design Server. It utilizes above one source node, three map nodes, three decision nodes, one route node, one log node, one fail node, two passthrough nodes, two format converter nodes and four target nodes
4. In Design Server, create a package chaps\_l4l\_mx\_validation that contains the input files and one flow chaps\_l4l\_validation\_flow. The maps will automatically be included during deployment of the package onto the runtime server.  
Note: Not required for running flows on Design Server user interface directly.
- [Run the example using three different methods](#)  
You can run the example using three different methods:

## Run the example using three different methods

You can run the example using three different methods:

1. Running the example from the Design Server user interface.
  2. Deploying the example to tx-rest and running it using the Swagger interface.
  3. Deploying locally or to ftp server and using the flow command server process (flowcmdserver).
- [Run the example from the Design Server user interface](#)  
Use the following steps to run the example from the Design Server user interface:
  - [Run the example using the Swagger interface](#)  
To run the example, deploy to tx-rest and run it using the Swagger interface.
  - [Run the example using the flow command server process \(flowcmdserver\)](#)  
To run the example, deploy locally or to ftp server and use the flow command server process (flowcmdserver).

## Run the example from the Design Server user interface

Use the following steps to run the example from the Design Server user interface:

1. In the Design Server, open chaps\_l4l\_validation\_flow flow and click on Run button.
2. Set toggle switch Run On Design Server and select flow input file. Example is chlf\_pacs\_009\_cov.xml.  
Flow will run and report with the green check box. The reports list execution of all nodes.
3. Right click and select View Link Data on each link to examine the data.
4. Right click on the node and select View Log to see the logs.

## Run the example using the Swagger interface

To run the example, deploy to tx-rest and run it using the Swagger interface.

1. Create a Web type server definition where the runtime tx-rest is installed if you do not have a server definition to deploy in Design Server.
2. If not running, start tx-rest on the server: <DTX\_HOME>/restapi/tomcat/dtxtomcat start tx-rest.
3. Deploy the created package to the server definition in Design Server.
4. Bring up the tx-rest Swagger UI: <DTX\_HOME>/restapi/tomcat/dtxtomcat showdocs tx-rest.
5. In the Swagger Explore window, enter /tx-rest/v2/docs and click on the Explore button to view the deployed package.  
You should see a Miscellaneous section having two actions: A **PUT** /v2/run/chaps\_l4l\_validation\_flow and a **POST** /v2/run/chaps\_l4l\_validation\_flow.
6. In the **PUT** action:
  - a. Expand the **PUT** action and select the Try it out button.
  - b. Leave the input and output sections empty.
  - c. Under flow\_vars section, delete the entire content and replace with the commands to run the flow, for example:

```
{
 "INPUT_FILE": "C:/mydir/data/chlf_pacs_009_cov.xml",
 "OUTPUT_RESULT_JSON": "C:/mydir/data/validation_result.json",
 "REPORT_FORMAT": "json",
 "VALIDATION_TYPE": "extended"
}

or

{
 "INPUT_FILE": "C:/mydir/data/chlf_pacs_009_cov.xml",
 "OUTPUT_RESULT_XML": "C:/mydir/data/validation_result.xml",
 "REPORT_FORMAT": "xml",
 "VALIDATION_TYPE": "extended"
}
```

- d. Click Execute blue bar for running the flow. When flow completes execution, 200 response code is shown in the Server Response section. This run should generate the output same as above.
7. Refresh the browser to delete the deployed package and get the Swagger Explore back to /tx-rest/openapi.json.  
There will be a DELETE /v2/packages/{name} action.
8. Expand the DELETE /v2/packages/{name} action and select Try it out.
9. In the Name field, enter the name you gave to the created package.
10. Select the true option from the stop query drop down, select the blue Execute bar.  
You can see a response of 204 with a timestamp, if it is successful.

## Run the example using the flow command server process (flowcmdserver)

To run the example, deploy locally or to ftp server and use the flow command server process (flowcmdserver).

1. Create a server definition where the runtime is installed, if you do not have a server local definition or ftp execution definition to deploy to, in Design Server.
2. In Design Server, deploy the created package to the server definition. For example, deploy to c:\ftpserver\deployment directory.
3. Execute the flow using the flow command server. The below command is for CHAPS L4L chlf\_pacs\_009\_cov.xml message.

```
flowcmdserver.exe --dir c:\ftpserver\deployment\flows
--flow chaps_141_validation_flow.json
--input "1;FILE;C:\ftpserver\deployment\tools\mx_service\data\chlf_pacs_009_cov.xml"
--audit "C:\ftpserver\deployment\tools\mx_service\data\chlf_pacs_009_cov_adt.json" -ad
```

The results will be in C:\ftpserver\deployment\flows as validation\_results.xml or validation\_results.json depending on the value of the REPORT\_FORMAT variable.

4. Use the below command to execute with the variable VALIDATION\_TYPE value defined in the Flow settings in Design Server ('extended' or 'schema').

```
flowcmdserver.exe --dir c:\ftpserver\deployment\flows
--flow chaps_141_validation_flow.json
--var "INPUT_FILE=C:\ftpserver\deployment\tools\mx_service\data\chlf_pacs_009_cov.xml"
--var "VALIDATION_TYPE=schema"
--var "REPORT_FORMAT=xml"
--audit "C:\ftpserver\deployment\tools\mx_service\data\chlf_pacs_009_cov_adt.json" -ad
```

5. The flow will report status similar to:

```
***Starting flow command server

Flow UUID: 85eabe2c-2302-4543-b347-9e18d65c6816
The flow audit file is: "C:\ftpserver\deployment\tools\mx_service\data\chlf_pacs_009_cov_adt.json"
Flow completed successfully
Elapsed time: 3048ms
```

6. Examine the flow output result file validation\_results.xml and the audit file chlf\_pacs\_009\_cov\_adt.json.

## How to customize the mxconfig.xml file

Rules enforced by the MX validation framework can be customized by editing the mxconfig.xml located under the validation/mx\_extended/data folder.

All the rules enforced by the MX validation map are under element ExtendedValidation.

Any of the sub-elements under ExtendedValidation can be enabled by setting the value to T or disabled by setting to F.

### Example

To disable the BIC lookup validation feature.

### About this task

Change sub-element ExtendedValidation/BICValidation value from T to F.

### Procedure

1. Download the mxconfig.xml file from the Files tab.
2. Open the mxconfig.xml file and edit the sub-element ExtendedValidation/BICValidation value setting from T to F, then save the file.
3. Upload the edited mxconfig.xml file to the project.
4. Run the flow.  
Run using an input with BIC lookup validation issue.

Right click on the output card on canvas and select View data. No BIC lookup validation reported.

## CHAPS L4L enhanced MX validation (using maps) Example

This example demonstrates the CHAPS L4L enhanced MX validation for CHAPS L4L messages using maps only.

The maps can perform following level of validation:

- MX extended validation includes following checks:
  - BIC lookup
  - Country code lookup
  - Currency code lookup
  - IBAN format
  - Allowed maximum fractional digits per currency
  - Usage guideline rules, see mxconfig.xml for list of rules
- Schema validation
- [\*\*What the example contains\*\*](#)  
Files and directories included in this example are as follows:
- [\*\*Run the example in Design Studio\*\*](#)
- [\*\*Run the example in Design Server User Interface\*\*](#)
- [\*\*How to customize the mxconfig.xml file\*\*](#)  
Rules enforced by the MX validation framework can be customized by editing the mxconfig.xml located under the validation/mx\_extended/data folder.

## What the example contains

Files and directories included in this example are as follows:

- Maps:  
The maps directory contains the following map sources:
  - chaps\_l4l\_mx\_admi\_validation\_enh.mms  
Utility maps called by main map for all the admi CHAPS L4L xml message MX validation.
  - chaps\_l4l\_mx\_camt\_validation\_enh.mms  
Utility maps called by main map for all the camt CHAPS L4L xml message MX validation.
  - chaps\_l4l\_mx\_mirs\_validation\_enh.mms  
Utility maps called by main map for all the mirs CHAPS L4L xml message MX validation.
  - chaps\_l4l\_mx\_pacs\_validation\_enh.mms  
Utility maps called by main map for all the pacs CHAPS L4L xml message MX validation.
  - chaps\_l4l\_mx\_validation\_frmwrk\_map\_enh.mms  
The main map used to apply MX validation to CHAPS L4L xml messages.
- Schemas:  
The schemas directory contains the following files:
  - bic.xsd  
Metadata that represents the bic.xml repository file structure.
  - ccy.xsd  
Metadata that represents the currencycodedecimals.xml repository file structure.
  - mxconfig.xsd  
Metadata that represents the mxconfig.xml configuration file structure.
  - adm1.004.001.02\_fqsm.xsd
  - adm1.004.001.02\_rsr.xsd
  - camt.052.001.08.xsd
  - camt.053.001.08.xsd
  - camt.054.001.08\_ca.xsd
  - camt.054.001.08\_cld.xsd
  - camt.054.001.08\_cli.xsd
  - camt.054.001.08\_lpa.xsd
  - camt.060.001.05.xsd
  - mirs.095.001.01.xsd
  - pacs.002.001.10.xsd
  - pacs.004.001.09.xsd
  - pacs.008.001.08.xsd
  - pacs.009.001.08\_core.xsd
  - pacs.009.001.08\_cov.xsd
  - head.001.001.02\_admi\_004\_fqsm.xsd
  - head.001.001.02\_admi\_004\_rsr.xsd
  - head.001.001.02\_camt\_052.xsd
  - head.001.001.02\_camt\_053.xsd
  - head.001.001.02\_camt\_054\_ca.xsd
  - head.001.001.02\_camt\_054\_cld.xsd
  - head.001.001.02\_camt\_054\_cli.xsd
  - head.001.001.02\_camt\_054\_lpa.xsd
  - head.001.001.02\_camt\_060.xsd
  - head.001.001.02\_mirs\_095.xsd
  - head.001.001.02\_pacs\_002.xsd
  - head.001.001.02\_pacs\_004.xsd
  - head.001.001.02\_pacs\_008.xsd
  - head.001.001.02\_pacs\_009\_core.xsd
  - head.001.001.02\_pacs\_009\_cov.xsd

Note: XML schemas were downloaded from SWIFT MyStandards Readiness CHAPS L4L portal.

- Trees:  
The trees directory contains the following files:
  - mxvalErrorReport.mtt  
Metadata that represents the xml based structure of the validation report.
  - swiftroute\_funds.mtt  
Metadata that is used as an internal element placeholder.
- Data:  
The data directory contains the following files:
  - bic.xml  
Repository file listing all BICs which are used during validation.
  - currencycodedecimals.xml  
Repository file listing country codes, currency codes and corresponding maximum fractionally digits, used as reference for validation.
  - mxconfig.xml  
Contains the MX configuration information on how to process the message.

Sample CHAPS L4L valid files for test purpose with header envelope:

- bah\_admi\_004\_001\_02\_fqsm\_valid.xml
- bah\_admi\_004\_001\_02\_rsr\_valid.xml
- bah\_camt\_052\_001\_08\_valid.xml
- bah\_camt\_053\_001\_08\_valid.xml
- bah\_camt\_054\_001\_08\_ca\_valid.xml
- bah\_camt\_054\_001\_08\_cld\_valid.xml
- bah\_camt\_054\_001\_08\_cli\_valid.xml
- bah\_camt\_054\_001\_08\_lpa\_valid.xml
- bah\_camt\_060\_001\_05\_valid.xml
- bah\_mirs\_095\_001\_01\_valid.xml
- bah\_pacs\_002\_001\_10\_valid.xml
- bah\_pacs\_004\_001\_09\_valid.xml
- bah\_pacs\_008\_001\_08\_valid.xml
- bah\_pacs\_009\_001\_08\_core\_valid.xml
- bah\_pacs\_009\_001\_08\_cov\_valid.xml

---

## Run the example in Design Studio

1. Build all the maps in the .mms files listed in What the example contains section.
2. Replace the input card 1 in router map, chlf8000\_val (under chaps\_l4l\_mx\_validation\_frmwrk\_map\_enh.mms file) with the desired input file.
3. Run the main router map, chlf8000\_val.  
You will see the output file in the data folder called chaps\_error\_report.xml.

---

## Run the example in Design Server User Interface

1. Open a browser pointing to the installation of the IBM Sterling Transformation Extender.
2. Enter the user and password credentials.
3. If the project does not exist, import chaps\_l4l.zip.
4. If absent, create a package containing all the Maps, Files and Flows.
5. Build the package in the desired server (to build all maps).
6. Open the project.
7. In the Maps tab, open the map chlf8000\_val.
8. Modify the settings for input card 1 on design canvas to point to the desired test file.
9. Save, build, and run the map chlf8000\_val.
10. Right click on the output card on canvas and select View data.

---

## How to customize the mxconfig.xml file

Rules enforced by the MX validation framework can be customized by editing the mxconfig.xml located under the validation/mx\_extended/data folder.

All the rules enforced by the MX validation map are under element ExtendedValidation.

Any of the sub-elements under ExtendedValidation can be enabled by setting the value to T or disabled by setting to F.

---

## Example

To disable the BIC lookup validation feature.

---

## About this task

Change sub-element ExtendedValidation/BICValidation value from T to F.

## Procedure

---

1. Download the mxconfig.xml file from the Files tab.
2. Open the mxconfig.xml file and edit the sub-element ExtendedValidation/BICValidation value setting from T to F, then save the file.
3. Upload the edited mxconfig.xml file to the project.
4. Run the map.  
Run using an input with BIC lookup validation issue.

Right click on the output card on canvas and select View data. No BIC lookup validation reported.

---

## CHAPS L4L schema validation (using maps) Example

This example demonstrates the CHAPS L4L schema validation for CHAPS L4L messages using maps only.

The maps can perform following level of validation:

- Schema validation
  - [\*\*What the example contains\*\*](#)  
Files and directories included in this example are as follows:
  - [\*\*Run the example in Design Studio\*\*](#)
  - [\*\*Run the example in Design Server User Interface\*\*](#)
- 

## What the example contains

Files and directories included in this example are as follows:

- Maps:  
The maps directory contains the following map sources:
  - chaps\_l4l\_mx\_schema\_validation\_xsd.mms  
Contains maps for schema validation of any CHAPS L4L xml message.
  - chaps\_l4l\_schema\_validation\_frmwrk\_map\_xsd.mms  
The main map used to apply schema validation to CHAPS L4L xml messages.

- Schemas:  
The schemas directory contains the following files:

- admi.004.001.02\_fqsm.xsd
- admi.004.001.02\_rsr.xsd
- camt.052.001.08.xsd
- camt.053.001.08.xsd
- camt.054.001.08\_ca.xsd
- camt.054.001.08\_cld.xsd
- camt.054.001.08\_cli.xsd
- camt.054.001.08\_lpa.xsd
- camt.060.001.05.xsd
- mirs.095.001.01.xsd
- pacs.002.001.10.xsd
- pacs.004.001.09.xsd
- pacs.008.001.08.xsd
- pacs.009.001.08\_core.xsd
- pacs.009.001.08\_cov.xsd
- head.001.001.02\_admi\_004\_fqsm.xsd
- head.001.001.02\_admi\_004\_rsr.xsd
- head.001.001.02\_camt\_052.xsd
- head.001.001.02\_camt\_053.xsd
- head.001.001.02\_camt\_054\_ca.xsd
- head.001.001.02\_camt\_054\_cld.xsd
- head.001.001.02\_camt\_054\_cli.xsd
- head.001.001.02\_camt\_054\_lpa.xsd
- head.001.001.02\_camt\_060.xsd
- head.001.001.02\_mirs\_095.xsd
- head.001.001.02\_pacs\_002.xsd
- head.001.001.02\_pacs\_004.xsd
- head.001.001.02\_pacs\_008.xsd
- head.001.001.02\_pacs\_009\_core.xsd
- head.001.001.02\_pacs\_009\_cov.xsd

Note: XML schemas were downloaded from SWIFT MyStandards Readiness CHAPS L4L portal.

- Trees:  
The trees directory contains the following files:

- mxvalErrorReport.mtt  
Metadata that represents the xml based structure of the validation report.
  - swiftroute\_funds.mtt  
Metadata that is used as an internal element placeholder.
  - Data:  
The data directory contains the following file:
- Sample CHAPS L4L valid files for test purpose with header envelope:
- bah\_admi\_004\_001\_02\_fqsm\_valid.xml
  - bah\_admi\_004\_001\_02\_rsr\_valid.xml
  - bah\_camt\_052\_001\_08\_valid.xml
  - bah\_camt\_053\_001\_08\_valid.xml
  - bah\_camt\_054\_001\_08\_ca\_valid.xml
  - bah\_camt\_054\_001\_08\_cld\_valid.xml
  - bah\_camt\_054\_001\_08\_cli\_valid.xml
  - bah\_camt\_054\_001\_08\_lpa\_valid.xml
  - bah\_camt\_060\_001\_05\_valid.xml
  - bah\_mirs\_095\_001\_01\_valid.xml
  - bah\_pacs\_002\_001\_10\_valid.xml
  - bah\_pacs\_004\_001\_09\_valid.xml
  - bah\_pacs\_008\_001\_08\_valid.xml
  - bah\_pacs\_009\_001\_08\_core\_valid.xml
  - bah\_pacs\_009\_001\_08\_cov\_valid.xml

## Run the example in Design Studio

1. Build all the maps in the .mms files listed in What the example contains section.
2. Replace the input card 1 in router map, chlf8200\_val (under chaps\_l4l\_schema\_validation\_frmwrk\_map\_xsd.mms file) with the desired input file.
3. Run the main router map, chlf8200\_val.  
You will see the output file in the data folder called chaps\_error\_report.xml.

## Run the example in Design Server User Interface

1. Open a browser pointing to the installation of the IBM Sterling Transformation Extender.
2. Enter the user and password credentials.
3. If the project does not exist, import chaps\_l4l.zip.
4. If absent, create a package containing all the Maps, Files and Flows.
5. Build the package in the desired server (to build all maps).
6. Open the project.
7. In the Maps tab, open the map chlf8200\_val.
8. Modify the settings for input card 1 on design canvas to point to the desired test file.
9. Save, build, and run the map chlf8200\_val.
10. Right click on the output card on canvas and select View data.

## CHAPS ENH Examples:

CHAPS ENH examples are as follow:

- [\*\*CHAPS ENH enhanced MX validation \(using flows\) Example\*\*](#)  
This example demonstrates the CHAPS ENH enhanced MX validation for CHAPS ENH messages using the flow server.
- [\*\*CHAPS ENH enhanced MX validation \(using maps\) Example\*\*](#)  
This example demonstrates the CHAPS ENH enhanced MX validation for CHAPS ENH messages using maps.
- [\*\*CHAPS ENH schema validation \(using maps\) Example\*\*](#)  
This example demonstrates the CHAPS ENH schema validation for CHAPS ENH messages using maps only.

## CHAPS ENH enhanced MX validation (using flows) Example

This example demonstrates the CHAPS ENH enhanced MX validation for CHAPS ENH messages using the flow server.

The flow can perform following level of validation:

- MX extended validation includes following checks:
  - BIC lookup
  - Country code lookup
  - Currency code lookup
  - IBAN format
  - Allowed maximum fractional digits per currency
  - Usage guideline rules, see mxconfig.xml for list of rules

- Schema validation
  - [\*\*What the example contains\*\*](#)  
Files included in this example are as follows:
  - [\*\*How to run the example\*\*](#)  
This CHAPS ENH MX Validation will use the sample files to demonstrate the generation of validation reports as output from a CHAPS ENH XML message.
  - [\*\*How to customize the mxconfig.xml file\*\*](#)  
Rules enforced by the MX validation framework can be customized by editing the mxconfig.xml located under the validation/mx\_extended/data folder.
- 

## What the example contains

Files included in this example are as follows:

- chaps\_enh.zip
  - Test Data Files:
    - chen\_pacs\_009\_cov.xml
    - chen\_pacs\_009\_cov\_bad\_rule.xml
    - chen\_pacs\_009\_cov\_bad\_schema.xml
  - Validation Files:
    - mxconfig.xml
    - bic.xml
    - currencycodedecimals.xml
  - Schemas:
    - bic.xsd  
Metadata that represents the bic.xml repository file structure.
    - ccy.xsd  
Metadata that represents the currencycodedecimals.xml repository file structure.
    - mxconfig.xsd  
Metadata that represents the mxconfig.xml configuration file structure.
    - mxvalErrorReport.mtt  
Metadata that represents the xml based structure of the validation report.
    - swiftroute\_funds.mtt  
Metadata that is used as an internal element placeholder.
    - admi.004.001.02\_fqsm.xsd
    - admi.004.001.02\_rsr.xsd
    - camt.052.001.08.xsd
    - camt.053.001.08.xsd
    - camt.054.001.08\_ca.xsd
    - camt.054.001.08\_cld.xsd
    - camt.054.001.08\_cli.xsd
    - camt.054.001.08\_lpa.xsd
    - camt.060.001.05.xsd
    - mirs.095.001.01.xsd
    - pacs.002.001.10.xsd
    - pacs.004.001.09.xsd
    - pacs.008.001.08.xsd
    - pacs.009.001.08\_core.xsd
    - pacs.009.001.08\_cov.xsd
    - head.001.001.02\_admi\_004\_fqsm.xsd
    - head.001.001.02\_admi\_004\_rsr.xsd
    - head.001.001.02\_camt\_052.xsd
    - head.001.001.02\_camt\_053.xsd
    - head.001.001.02\_camt\_054\_ca.xsd
    - head.001.001.02\_camt\_054\_cld.xsd
    - head.001.001.02\_camt\_054\_cli.xsd
    - head.001.001.02\_camt\_054\_lpa.xsd
    - head.001.001.02\_camt\_060.xsd
    - head.001.001.02\_mirs\_095.xsd
    - head.001.001.02\_pacs\_002.xsd
    - head.001.001.02\_pacs\_004.xsd
    - head.001.001.02\_pacs\_008.xsd
    - head.001.001.02\_pacs\_009\_core.xsd
    - head.001.001.02\_pacs\_009\_cov.xsd

Note: XML schemas were downloaded from SWIFT MyStandards Readiness CHAPS ENH Portal.

- Maps:
  - For Extended Validation:
    - chen8000\_val
    - chen8100\_val
    - chen8056\_head112\_camt052
    - chen8006\_camt\_052\_001\_08
    - chen8055\_head112\_camt053
    - chen8005\_camt\_053\_001\_08
    - chen8051\_head112\_camt054\_ca
    - chen8001\_camt\_054\_001\_08\_ca

- chen8052\_head112\_camt054\_cld
- chen8002\_camt\_054\_001\_08\_cld
- chen8053\_head112\_camt054\_cli
- chen8003\_camt\_054\_001\_08\_cli
- chen8054\_head112\_camt054\_lpa
- chen8004\_camt\_054\_001\_08\_lpa
- chen8060\_head112\_camt060
- chen8009\_camt\_060\_001\_05
- chen8057\_head112\_pacs\_002
- chen8007\_pacs\_002\_001\_10
- chen8058\_head112\_pacs\_008
- chen8008\_pacs\_008\_001\_08
- chen8059\_head112\_pacs\_004
- chen8010\_pacs\_004\_001\_09
- chen8061\_head112\_pacs\_009\_core
- chen8011\_pacs\_009\_001\_08\_core
- chen8062\_head112\_pacs\_009\_cov
- chen8012\_pacs\_009\_001\_08\_cov
- chen8401\_head112\_mirs095
- chen8301\_mirs\_095\_001\_01
- chen8601\_head112\_admi004\_fqsm
- chen8501\_admi\_004\_001\_02\_fqsm
- chen8602\_head112\_admi004\_rsr
- chen8502\_admi\_004\_001\_02\_rsr
- For Schema Validation:
  - chen8200\_val
  - chen8801\_chaps\_enh\_bah\_val
  - chen8802\_chaps\_enh\_doc\_val
  - chen8201\_camt\_054\_001\_08\_ca
  - chen8202\_camt\_054\_001\_08\_cld
  - chen8203\_camt\_054\_001\_08\_cli
  - chen8204\_camt\_054\_001\_08\_lpa
  - chen8205\_camt\_053\_001\_08
  - chen8206\_camt\_052\_001\_08
  - chen8207\_pacs\_002\_001\_10
  - chen8208\_pacs\_008\_001\_08
  - chen8209\_camt\_060\_001\_05
  - chen8210\_pacs\_004\_001\_09
  - chen8211\_pacs\_009\_001\_08\_core
  - chen8212\_pacs\_009\_001\_08\_cov
  - chen8213\_mirs\_095\_001\_01
  - chen8214\_admi\_004\_001\_02\_fqsm
  - chen8215\_admi\_004\_001\_02\_rsr
  - chen8251\_head112\_camt054\_ca
  - chen8252\_head112\_camt054\_cld
  - chen8253\_head112\_camt054\_cli
  - chen8254\_head112\_camt054\_lpa
  - chen8255\_head112\_camt053
  - chen8256\_head112\_camt052
  - chen8257\_head112\_pacs\_002\_001\_10
  - chen8258\_head112\_pacs\_008\_001\_08
  - chen8259\_head112\_camt060
  - chen8260\_head112\_pacs004
  - chen8261\_head112\_pacs009\_core
  - chen8262\_head112\_pacs009\_cov
  - chen8263\_head112\_mirs095
  - chen8264\_head112\_admi004\_fqsm
  - chen8265\_head112\_admi004\_rsr
- Common
  - mxut1002\_bizsvc\_chaps
- Flows:
  - chaps\_enh\_validation\_flow

## How to run the example

This CHAPS ENH MX Validation will use the sample files to demonstrate the generation of validation reports as output from a CHAPS ENH XML message.

The stopValidation.json file will report the validation failure due to the pre-conversion checks:

**If input file is other than any of the supported CHAPS ENH messages as per schema list indicated above.**

1. Import the chaps\_enh.zip project into the Design Server.
2. Open the chaps\_enh project in Design Server and view the flow chaps\_enh\_validation\_flow.
  - a. The Flow Description points to all the maps to be executed during the flow process, which will be called from within some of the map nodes in the flow. The following is a list of maps:
    - @packagemap=chaps\_enh/validation/mx\_extended/maps/chaps\_enh\_mx\_camt\_validation\_enh/chen8056\_head112\_camt052
    - @packagemap=chaps\_enh/validation/mx\_extended/maps/chaps\_enh\_mx\_camt\_validation\_enh/chen8056\_head112\_camt052
    - @packagemap=chaps\_enh/validation/mx\_extended/maps/chaps\_enh\_mx\_camt\_validation\_enh/chen8006\_camt\_052\_001\_08

- @packagemap=chaps\_enh/validation/mx\_extended/maps/chaps\_enh\_mx\_camt\_validation\_enh/chen8055\_head112\_camt053
- @packagemap=chaps\_enh/validation/mx\_extended/maps/chaps\_enh\_mx\_camt\_validation\_enh/chen8005\_camt\_053\_001\_08
- @packagemap=chaps\_enh/validation/mx\_extended/maps/chaps\_enh\_mx\_camt\_validation\_enh/chen8051\_head112\_camt054\_ca
- @packagemap=chaps\_enh/validation/mx\_extended/maps/chaps\_enh\_mx\_camt\_validation\_enh/chen8001\_camt\_054\_001\_08\_ca
- @packagemap=chaps\_enh/validation/mx\_extended/maps/chaps\_enh\_mx\_camt\_validation\_enh/chen8052\_head112\_camt054\_cld
- @packagemap=chaps\_enh/validation/mx\_extended/maps/chaps\_enh\_mx\_camt\_validation\_enh/chen8002\_camt\_054\_001\_08\_cld
- @packagemap=chaps\_enh/validation/mx\_extended/maps/chaps\_enh\_mx\_camt\_validation\_enh/chen8053\_head112\_camt054\_cli
- @packagemap=chaps\_enh/validation/mx\_extended/maps/chaps\_enh\_mx\_camt\_validation\_enh/chen8003\_camt\_054\_001\_08\_cli
- @packagemap=chaps\_enh/validation/mx\_extended/maps/chaps\_enh\_mx\_camt\_validation\_enh/chen8054\_head112\_camt054\_lpa
- @packagemap=chaps\_enh/validation/mx\_extended/maps/chaps\_enh\_mx\_camt\_validation\_enh/chen8004\_camt\_054\_001\_08\_lpa
- @packagemap=chaps\_enh/validation/mx\_extended/maps/chaps\_enh\_mx\_camt\_validation\_enh/chen8060\_head112\_camt060
- @packagemap=chaps\_enh/validation/mx\_extended/maps/chaps\_enh\_mx\_camt\_validation\_enh/chen8009\_camt\_060\_001\_05
- @packagemap=chaps\_enh/validation/mx\_extended/maps/chaps\_enh\_mx\_pacs\_validation\_enh/chen8057\_head112\_pacs\_002
- @packagemap=chaps\_enh/validation/mx\_extended/maps/chaps\_enh\_mx\_pacs\_validation\_enh/chen8007\_pacs\_002\_001\_10
- @packagemap=chaps\_enh/validation/mx\_extended/maps/chaps\_enh\_mx\_pacs\_validation\_enh/chen8058\_head112\_pacs\_008
- @packagemap=chaps\_enh/validation/mx\_extended/maps/chaps\_enh\_mx\_pacs\_validation\_enh/chen8008\_pacs\_008\_001\_08
- @packagemap=chaps\_enh/validation/mx\_extended/maps/chaps\_enh\_mx\_pacs\_validation\_enh/chen8059\_head112\_pacs\_004
- @packagemap=chaps\_enh/validation/mx\_extended/maps/chaps\_enh\_mx\_pacs\_validation\_enh/chen8010\_pacs\_004\_001\_09
- @packagemap=chaps\_enh/validation/mx\_extended/maps/chaps\_enh\_mx\_pacs\_validation\_enh/chen8061\_head112\_pacs\_009\_core
- @packagemap=chaps\_enh/validation/mx\_extended/maps/chaps\_enh\_mx\_pacs\_validation\_enh/chen8011\_pacs\_009\_001\_08\_core
- @packagemap=chaps\_enh/validation/mx\_extended/maps/chaps\_enh\_mx\_pacs\_validation\_enh/chen8062\_head112\_pacs\_009\_cov
- @packagemap=chaps\_enh/validation/mx\_extended/maps/chaps\_enh\_mx\_pacs\_validation\_enh/chen8012\_pacs\_009\_001\_08\_cov
- @packagemap=chaps\_enh/validation/mx\_extended/maps/chaps\_enh\_mx\_mirs\_validation\_enh/chen8401\_head112\_mirs095
- @packagemap=chaps\_enh/validation/mx\_extended/maps/chaps\_enh\_mx\_mirs\_validation\_enh/chen8301\_mirs\_095\_001\_01
- @packagemap=chaps\_enh/validation/mx\_extended/maps/chaps\_enh\_mx\_admi\_validation\_enh/chen8601\_head112\_admi004\_fqsm
- @packagemap=chaps\_enh/validation/mx\_extended/maps/chaps\_enh\_mx\_admi\_validation\_enh/chen8501\_admi\_004\_001\_02\_fqsm
- @packagemap=chaps\_enh/validation/mx\_extended/maps/chaps\_enh\_mx\_admi\_validation\_enh/chen8602\_head112\_admi004\_rsr
- @packagemap=chaps\_enh/validation/mx\_extended/maps/chaps\_enh\_mx\_admi\_validation\_enh/chen8502\_admi\_004\_001\_02\_rsr
- @packagemap=chaps\_enh/validation/schema\_only/maps/chaps\_enh\_mx\_schema\_validation\_xsd/chen8201\_camt\_054\_001\_08\_ca
- @packagemap=chaps\_enh/validation/schema\_only/maps/chaps\_enh\_mx\_schema\_validation\_xsd/chen8202\_camt\_054\_001\_08\_cld
- @packagemap=chaps\_enh/validation/schema\_only/maps/chaps\_enh\_mx\_schema\_validation\_xsd/chen8203\_camt\_054\_001\_08\_cli
- @packagemap=chaps\_enh/validation/schema\_only/maps/chaps\_enh\_mx\_schema\_validation\_xsd/chen8204\_camt\_054\_001\_08\_lpa
- @packagemap=chaps\_enh/validation/schema\_only/maps/chaps\_enh\_mx\_schema\_validation\_xsd/chen8205\_camt\_053\_001\_08
- @packagemap=chaps\_enh/validation/schema\_only/maps/chaps\_enh\_mx\_schema\_validation\_xsd/chen8206\_camt\_052\_001\_08
- @packagemap=chaps\_enh/validation/schema\_only/maps/chaps\_enh\_mx\_schema\_validation\_xsd/chen8207\_pacs\_002\_001\_10
- @packagemap=chaps\_enh/validation/schema\_only/maps/chaps\_enh\_mx\_schema\_validation\_xsd/chen8208\_pacs\_008\_001\_08
- @packagemap=chaps\_enh/validation/schema\_only/maps/chaps\_enh\_mx\_schema\_validation\_xsd/chen8209\_camt\_060\_001\_05
- @packagemap=chaps\_enh/validation/schema\_only/maps/chaps\_enh\_mx\_schema\_validation\_xsd/chen8210\_pacs\_004\_001\_09
- @packagemap=chaps\_enh/validation/schema\_only/maps/chaps\_enh\_mx\_schema\_validation\_xsd/chen8211\_pacs\_009\_001\_08\_core
- @packagemap=chaps\_enh/validation/schema\_only/maps/chaps\_enh\_mx\_schema\_validation\_xsd/chen8212\_pacs\_009\_001\_08\_cov
- @packagemap=chaps\_enh/validation/schema\_only/maps/chaps\_enh\_mx\_schema\_validation\_xsd/chen8213\_mirs\_095\_001\_01
- @packagemap=chaps\_enh/validation/schema\_only/maps/chaps\_enh\_mx\_schema\_validation\_xsd/chen8214\_admi\_004\_001\_02\_fqsm
- @packagemap=chaps\_enh/validation/schema\_only/maps/chaps\_enh\_mx\_schema\_validation\_xsd/chen8215\_admi\_004\_001\_02\_rsr
- @packagemap=chaps\_enh/validation/schema\_only/maps/chaps\_enh\_mx\_schema\_validation\_xsd/chen8251\_head112\_camt054\_ca
- @packagemap=chaps\_enh/validation/schema\_only/maps/chaps\_enh\_mx\_schema\_validation\_xsd/chen8252\_head112\_camt054\_cld
- @packagemap=chaps\_enh/validation/schema\_only/maps/chaps\_enh\_mx\_schema\_validation\_xsd/chen8253\_head112\_camt054\_cli
- @packagemap=chaps\_enh/validation/schema\_only/maps/chaps\_enh\_mx\_schema\_validation\_xsd/chen8254\_head112\_camt054\_lpa
- @packagemap=chaps\_enh/validation/schema\_only/maps/chaps\_enh\_mx\_schema\_validation\_xsd/chen8255\_head112\_camt053
- @packagemap=chaps\_enh/validation/schema\_only/maps/chaps\_enh\_mx\_schema\_validation\_xsd/chen8256\_head112\_camt052
- @packagemap=chaps\_enh/validation/schema\_only/maps/chaps\_enh\_mx\_schema\_validation\_xsd/chen8257\_head112\_pacs\_002\_001\_10
- @packagemap=chaps\_enh/validation/schema\_only/maps/chaps\_enh\_mx\_schema\_validation\_xsd/chen8258\_head112\_pacs\_008\_001\_08
- @packagemap=chaps\_enh/validation/schema\_only/maps/chaps\_enh\_mx\_schema\_validation\_xsd/chen8259\_head112\_camt060
- @packagemap=chaps\_enh/validation/schema\_only/maps/chaps\_enh\_mx\_schema\_validation\_xsd/chen8260\_head112\_pacs004
- @packagemap=chaps\_enh/validation/schema\_only/maps/chaps\_enh\_mx\_schema\_validation\_xsd/chen8261\_head112\_pacs009\_core
- @packagemap=chaps\_enh/validation/schema\_only/maps/chaps\_enh\_mx\_schema\_validation\_xsd/chen8262\_head112\_pacs009\_cov
- @packagemap=chaps\_enh/validation/schema\_only/maps/chaps\_enh\_mx\_schema\_validation\_xsd/chen8263\_head112\_mirs095
- @packagemap=chaps\_enh/validation/schema\_only/maps/chaps\_enh\_mx\_schema\_validation\_xsd/chen8264\_head112\_admi004\_fqsm
- @packagemap=chaps\_enh/validation/schema\_only/maps/chaps\_enh\_mx\_schema\_validation\_xsd/chen8265\_head112\_admi004\_rsr

a. It utilizes the following nodes:

i. Source Nodes:

- mx\_input

This node identifies the input data to be validated in the flow. It uses the INPUT\_FILE variable to set the location of the data.

ii. Map Nodes:

- MX pre-check

Runs map mxut1002\_bizsvc\_chaps which sets flow variables, checks pre-validation conditions and creates infoset.json for validation.

- EXT\_VAL

Runs map chen8100\_val which calls the appropriate map to perform extended validation of a CHAPS ENH message.

- XSD\_VAL

Runs map chen8200\_val which calls the appropriate map to perform schema validation of a CHAPS ENH message.

Note: The maps chen8100\_val and chen8200\_val internally call all the other maps identified in the above Flow Description step.

iii. Decision Nodes:

- pre-valid chk

This node checks the flow variable stopValidation to decide if further validation/processing is not required.

- EXT\_RESULT\_FORMAT

This node checks the value of the flow variable REPORT\_FORMAT to determine if the resulting report for extended validation should be generated in XML (default) or JSON.

- XSD\_RESULT\_FORMAT  
This node checks the value of the flow variable REPORT\_FORMAT to determine if the resulting report for schema-only validation should be generated in XML (default) or JSON.

iv. Route Nodes:

- VAL\_TYPE  
This node checks the variable VALIDATION\_TYPE to determine if the process will do extended validation or schema-only validation on the data.

v. Log Nodes:

- FAILURE  
In case of no validation due to precondition check failures, this node creates a log file specified by the flow variable FAILURE\_LOG.

vi. Fail Node:

- STOP  
It generates error message setup in the node in case of no validation performed.

vii. Passthrough Nodes:

- EXT\_XML\_CONVERT  
This node receives an extended validation report in XML format and does not modify the content, thus passing it as is to the appropriate target node.
- XSD\_XML\_CONVERT  
This node receives a schema-only validation report in XML format and does not modify the content, thus passing it as is to the appropriate target node.

viii. Format Converter Nodes:

- EXT\_JSON\_CONVERT  
This node receives an extended validation report in XML format and converts it to JSON before passing it to the appropriate target node.
- XSD\_JSON\_CONVERT  
This node receives a schema-only validation report in XML format and converts it to JSON before passing it to the appropriate target node.

ix. Target Nodes:

- ext\_xml  
This node contains the resulting extended validation report in XML format and creates the output as defined by the variable OUTPUT\_RESULT\_XML.
- ext\_json  
This node contains the resulting extended validation report in JSON format and creates the output as defined by the variable OUTPUT\_RESULT\_JSON.
- xsd\_xml  
This node contains the resulting schema-only validation report in XML format and creates the output as defined by the variable OUTPUT\_RESULT\_XML.
- xsd\_json  
This node contains the resulting schema-only validation report in JSON format and creates the output as defined by the variable OUTPUT\_RESULT\_JSON.

b. It utilizes the following variables:

Flow variables:

- VALIDATION\_TYPE  
The default value is extended.

For schema-only validation, it can be changed to schema. This is used in Route node VAL\_TYPE.

- REPORT\_FORMAT  
The default value is xml.

It can be changed to json. This is used in Decision nodes EXT\_RESULT\_FORMAT and XSD\_RESULT\_FORMAT.

- INPUT\_FILE  
The default value is ..../tools/mx\_service/data/chen\_pacs\_009\_cov.xml.

This is the data file to be used for validation and can be customized. This is used in Source node mx\_input.

- BIC\_FILE  
The default value is ..../data/bic.xml.

This is the location of the bic cross-reference file that is used in all the maps executed by the map in node EXT\_VAL. It can be customized.

- CCY\_FILE  
The default value is ..../data/currencycodedecimals.xml.

This is the location of the country code cross-reference file that is used in all the maps executed by the map in node EXT\_VAL. It can be customized.

- MXCONFIG\_FILE  
The default value is ..../data/mxconfig.xml.

This is the location of the file containing the rule validation settings that is used in all the maps executed by the map in node EXT\_VAL. It can be customized.

- OUTPUT\_RESULT\_XML  
The default value is validation\_result.xml.

This is the location of the validation report file in xml format. It is used in Target nodes ext\_xml and xsd\_xml. It can be customized.

- OUTPUT\_RESULT\_JSON

The default value is validation\_result.json.

This is the location of the validation report file in json format. It is used in Target nodes ext\_json and xsd\_json. It can be customized.

- FAILURE\_LOG

The default value is stopMXValidation.json.

This is the location of the failure log. It is used in Log node FAILURE. It can be customized.

- bizSvc

For internal use in the Map node MX pre-check to determine the type of data being read in the input file.

- stopValidation

For internal use in the Map node MX pre-check and is checked in Decision node pre-valid chk to cause a Failure log in case a data file was not recognized as a valid CHAPS ENH message.

3. Open the main flow chaps\_enh\_validation\_flow in the Design Server. It utilizes above one source node, three map nodes, three decision nodes, one route node, one log node, one fail node, two passthrough nodes, two format converter nodes and four target nodes

4. In Design Server, create a package chaps\_enh\_mx\_validation that contains the input files and one flow chaps\_enh\_validation\_flow. The maps will automatically be included during deployment of the package onto the runtime server.

Note: Not required for running flows on Design Server user interface directly.

- [Run the example using three different methods](#)

You can run the example using three different methods:

## Run the example using three different methods

You can run the example using three different methods:

1. Running the example from the Design Server user interface.

2. Deploying the example to tx-rest and running it using the Swagger interface.

3. Deploying locally or to ftp server and using the flow command server process (flowcmdserver).

- [Run the example from the Design Server user interface](#)

Use the following steps to run the example from the Design Server user interface:

- [Run the example using the Swagger interface](#)

To run the example, deploy to tx-rest and run it using the Swagger interface.

- [Run the example using the flow command server process \(flowcmdserver\)](#)

To run the example, deploy locally or to ftp server and use the flow command server process (flowcmdserver).

## Run the example from the Design Server user interface

Use the following steps to run the example from the Design Server user interface:

1. In the Design Server, open chaps\_enh\_validation\_flow flow and click on Run button.

2. Set toggle switch Run On Design Server and select flow input file. Example is chen\_pacs\_009\_cov.xml.

Flow will run and report with the green check box. The reports list execution of all nodes.

3. Right click and select View Link Data on each link to examine the data.

4. Right click on the node and select View Log to see the logs.

## Run the example using the Swagger interface

To run the example, deploy to tx-rest and run it using the Swagger interface.

1. Create a Web type server definition where the runtime tx-rest is installed if you do not have a server definition to deploy in Design Server.

2. If not running, start tx-rest on the server: <DTX\_HOME>/restapi/tomcat/dtxtomcat start tx-rest.

3. Deploy the created package to the server definition in Design Server.

4. Bring up the tx-rest Swagger UI: <DTX\_HOME>/restapi/tomcat/dtxtomcat showdocs tx-rest.

5. In the Swagger Explore window, enter /tx-rest/v2/docs and click on the Explore button to view the deployed package.

You should see a Miscellaneous section having two actions: A **PUT** /v2/run/chaps\_enh\_validation\_flow and a **POST** /v2/run/chaps\_enh\_validation\_flow.

6. In the **PUT** action:

a. Expand the **PUT** action and select the Try it out button.

b. Leave the input and output sections empty.

c. Under flow\_vars section, delete the entire content and replace with the commands to run the flow, for example:

```
{
 "INPUT_FILE": "C:/mydir/data/chen_pacs_009_cov.xml",
 "OUTPUT_RESULT_JSON": "C:/mydir/data/validation_result.json",
 "REPORT_FORMAT": "json",
 "VALIDATION_TYPE": "extended"
}
```

or

```
{
 "INPUT_FILE": "C:/mydir/data/chen_pacs_009_cov.xml",
 "OUTPUT_RESULT_XML": "C:/mydir/data/validation_result.xml",
 "REPORT_FORMAT": "xml",
 "VALIDATION_TYPE": "extended"
}
```

- d. Click Execute blue bar for running the flow. When flow completes execution, 200 response code is shown in the Server Response section. This run should generate the output same as above.
7. Refresh the browser to delete the deployed package and get the Swagger Explore back to /tx-rest/openapi.json. There will be a DELETE /v2/packages/{name} action.
  8. Expand the DELETE /v2/packages/{name} action and select Try it out.
  9. In the Name field, enter the name you gave to the created package.
  10. Select the true option from the stop query drop down, select the blue Execute bar.  
You can see a response of 204 with a timestamp, if it is successful.

## Run the example using the flow command server process (flowcmdserver)

To run the example, deploy locally or to ftp server and use the flow command server process (flowcmdserver).

1. Create a server definition where the runtime is installed, if you do not have a server local definition or ftp execution definition to deploy to, in Design Server.
2. In Design Server, deploy the created package to the server definition. For example, deploy to c:\ftpserver\deployment directory.
3. Execute the flow using the flow command server. The below command is for CHAPS ENH chen\_pacs\_009\_cov.xml message.

```
flowcmdserver.exe --dir c:\ftpserver\deployment\flows
--flow chaps_enh_validation_flow.json
--input "1;FILE;C:\ftpserver\deployment\tools\mx_service\data\chen_pacs_009_cov.xml"
--audit "C:\ftpserver\deployment\tools\mx_service\data\chen_pacs_009_cov_adt.json" -ad
```

- The results will be in C:\ftpserver\deployment\flows as validation\_results.xml or validation\_results.json depending on the value of the REPORT\_FORMAT variable.
4. Use the below command to execute with the variable VALIDATION\_TYPE value defined in the Flow settings in Design Server ('extended' or 'schema').

```
--flow chaps_enh_validation_flow.json
--var "INPUT_FILE=C:\ftpserver\deployment\tools\mx_service\data\chen_pacs_009_cov.xml"
--var "VALIDATION_TYPE=schema"
--var "REPORT_FORMAT=xml"
--audit "C:\ftpserver\deployment\tools\mx_service\data\chen_pacs_009_cov_adt.json" -ad
```

5. The flow will report status similar to:

```
***Starting flow command server
Flow UUID: 85eabe2c-2302-4543-b347-9e18d65c6816
The flow audit file is: "C:\\ftpserver\\deployment\\tools\\mx_service\\data\\chen_pacs_009_cov_adt.json"
Flow completed successfully
Elapsed time: 3048ms
```

6. Examine the flow output result files validation\_results.xml and the audit file chen\_pacs\_009\_cov\_adt.json.

## How to customize the mxconfig.xml file

Rules enforced by the MX validation framework can be customized by editing the mxconfig.xml located under the validation/mx\_extended/data folder.

All the rules enforced by the MX validation map are under element ExtendedValidation.

Any of the sub-elements under ExtendedValidation can be enabled by setting the value to T or disabled by setting to F.

### Example

To disable the BIC lookup validation feature.

### About this task

Change sub-element ExtendedValidation/BICValidation value from T to F.

### Procedure

1. Download the mxconfig.xml file from the Files tab.
2. Open the mxconfig.xml file and edit the sub-element ExtendedValidation/BICValidation value setting from T to F, then save the file.
3. Upload the edited mxconfig.xml file to the project.
4. Run the flow.  
Run using an input with BIC lookup validation issue.

Right click on the output card on canvas and select View data. No BIC lookup validation reported.

## CHAPS ENH enhanced MX validation (using maps) Example

This example demonstrates the CHAPS ENH enhanced MX validation for CHAPS ENH messages using maps.

The maps can perform following level of validation:

- MX extended validation includes following checks:
  - BIC lookup
  - Country code lookup
  - Currency code lookup
  - IBAN format
  - Allowed maximum fractional digits per currency
  - Usage guideline rules, see mxconfig.xml for list of rules
- Schema validation
- **[What the example contains](#)**  
Files and directories included in this example are as follows:
- **[Run the example in Design Studio](#)**
- **[Run the example in Design Server User Interface](#)**
- **[How to customize the mxconfig.xml file](#)**  
Rules enforced by the MX validation framework can be customized by editing the mxconfig.xml located under the validation/mx\_extended/data folder.

## What the example contains

Files and directories included in this example are as follows:

- Maps:  
The maps directory contains the following map sources:
  - chaps\_enh\_mx\_admi\_validation\_enh.mms  
Utility maps called by main map for all the admi CHAPS ENH xml message MX validation.
  - chaps\_enh\_mx\_camt\_validation\_enh.mms  
Utility maps called by main map for all the camt CHAPS ENH xml message MX validation.
  - chaps\_enh\_mx\_mirs\_validation\_enh.mms  
Utility maps called by main map for all the mirs CHAPS ENH xml message MX validation.
  - chaps\_enh\_mx\_pacs\_validation\_enh.mms  
Utility maps called by main map for all the pacs CHAPS ENH xml message MX validation.
  - chaps\_enh\_mx\_validation\_frmwrk\_map\_enh.mms  
The Main map used to apply MX validation to CHAPS ENH xml messages.
- Schemas:  
The schemas directory contains the following files:
  - bic.xsd  
Metadata that represents the bic.xml repository file structure.
  - ccy.xsd  
Metadata that represents the currencycodedecimals.xml repository file structure.
  - mxconfig.xsd  
Metadata that represents the mxconfig.xml configuration file structure.
  - adm1.004.001.02\_fqsm.xsd
  - adm1.004.001.02\_rsr.xsd
  - camt.052.001.08.xsd
  - camt.053.001.08.xsd
  - camt.054.001.08\_ca.xsd
  - camt.054.001.08\_cld.xsd
  - camt.054.001.08\_cli.xsd
  - camt.054.001.08\_lpa.xsd
  - camt.060.001.05.xsd
  - mirs.095.001.01.xsd
  - pacs.002.001.10.xsd
  - pacs.004.001.09.xsd
  - pacs.008.001.08.xsd
  - pacs.009.001.08\_core.xsd
  - pacs.009.001.08\_cov.xsd
  - head.001.001.02\_admi\_004\_fqsm.xsd
  - head.001.001.02\_admi\_004\_rsr.xsd
  - head.001.001.02\_camt\_052.xsd
  - head.001.001.02\_camt\_053.xsd
  - head.001.001.02\_camt\_054\_ca.xsd
  - head.001.001.02\_camt\_054\_cld.xsd
  - head.001.001.02\_camt\_054\_cli.xsd
  - head.001.001.02\_camt\_054\_lpa.xsd
  - head.001.001.02\_camt\_060.xsd
  - head.001.001.02\_mirs\_095.xsd
  - head.001.001.02\_pacs\_002.xsd

- head.001.001.02\_pacs\_004.xsd
- head.001.001.02\_pacs\_008.xsd
- head.001.001.02\_pacs\_009\_core.xsd
- head.001.001.02\_pacs\_009\_cov.xsd

Note: XML schemas were downloaded from SWIFT MyStandards Readiness CHAPS ENH portal.

- Trees:

The trees directory contains the following files:

- mxvalErrorReport.mtt  
Metadata that represents the xml based structure of the validation report.
- swiftroute\_funds.mtt  
Metadata that is used as an internal element placeholder.

- Data:

The data directory contains the following file:

- bic.xml  
Repository file listing all BICs which are used during validation.
- currencycodedecimals.xml  
Repository file list country codes, currency codes and corresponding maximum fractionally digits, used as reference for validation.
- mxconfig.xml  
Contains the MX configuration information on how to process the message.

Sample CHAPS ENH valid files for test purpose with header envelope:

- bah\_admi\_004\_001\_02\_fqsm\_valid.xml
- bah\_admi\_004\_001\_02\_rsr\_valid.xml
- bah\_camt\_052\_001\_08\_valid.xml
- bah\_camt\_053\_001\_08\_valid.xml
- bah\_camt\_054\_001\_08\_ca\_valid.xml
- bah\_camt\_054\_001\_08\_cld\_valid.xml
- bah\_camt\_054\_001\_08\_cli\_valid.xml
- bah\_camt\_054\_001\_08\_lpa\_valid.xml
- bah\_camt\_060\_001\_05\_valid.xml
- bah\_mirs\_095\_001\_01\_valid.xml
- bah\_pacs\_002\_001\_10\_valid.xml
- bah\_pacs\_004\_001\_09\_valid.xml
- bah\_pacs\_008\_001\_08\_valid.xml
- bah\_pacs\_009\_001\_08\_core\_valid.xml
- bah\_pacs\_009\_001\_08\_cov\_valid.xml

## Run the example in Design Studio

1. Build all the maps in the .mms files listed in What the example contains section.
2. Replace the input card 1 in router map, chen8000\_val (under chaps\_enh\_mx\_validation\_frmwrk\_map\_enh.mms file) with the desired input file.
3. Run the main router map, chen8000\_val.  
You will see the output file in the data folder called chaps\_error\_report.xml.

## Run the example in Design Server User Interface

1. Open a browser pointing to the installation of the IBM Sterling Transformation Extender.
2. Enter the user and password credentials.
3. If the project does not exist, import chaps\_enh.zip.
4. If absent, create a package containing all the Maps, Files and Flows.
5. Build the package in the desired server (to build all maps).
6. Open the project.
7. In the Maps tab, open the map chen8000\_val.
8. Modify the settings for input card 1 on design canvas to point to the desired test file.
9. Save, build, and run the map chen8000\_val.
10. Right click on the output card on canvas and select View data.

## How to customize the mxconfig.xml file

Rules enforced by the MX validation framework can be customized by editing the mxconfig.xml located under the validation/mx\_extended/data folder.

All the rules enforced by the MX validation map are under element ExtendedValidation.

Any of the sub-elements under ExtendedValidation can be enabled by setting the value to T or disabled by setting to F.

## Example

To disable the BIC lookup validation feature.

## About this task

---

Change sub-element ExtendedValidation/BICValidation value from T to F.

## Procedure

---

1. Download the mxconfig.xml file from the Files tab.
2. Open the mxconfig.xml file and edit the sub-element ExtendedValidation/BICValidation value setting from T to F, then save the file.
3. Upload the edited mxconfig.xml file to the project.
4. Run the map.  
Run using an input with BIC lookup validation issue.

Right click on the output card on canvas and select View data. No BIC lookup validation reported.

---

## CHAPS ENH schema validation (using maps) Example

This example demonstrates the CHAPS ENH schema validation for CHAPS ENH messages using maps only.

The maps can perform following level of validation:

- Schema validation
- [What the example contains](#)  
Files and directories included in this example are as follows:
- [Run the example in Design Studio](#)
- [Run the example in Design Server User Interface](#)

---

## What the example contains

Files and directories included in this example are as follows:

- Maps:  
The maps directory contains the following map sources:
  - chaps\_enh\_mx\_schema\_validation\_xsd.mms  
Contains maps for schema validation of any CHAPS ENH xml message.
  - chaps\_enh\_schema\_validation\_frmwrk\_map\_xsd.mms  
The main map used to apply schema validation to CHAPS ENH xml messages.
- Schemas:  
The schemas directory contains the following files:
  - admi.004.001.02\_fqsm.xsd
  - admi.004.001.02\_rsr.xsd
  - camt.052.001.08.xsd
  - camt.053.001.08.xsd
  - camt.054.001.08\_ca.xsd
  - camt.054.001.08\_cld.xsd
  - camt.054.001.08\_cli.xsd
  - camt.054.001.08\_lpa.xsd
  - camt.060.001.05.xsd
  - mirs.095.001.01.xsd
  - pacs.002.001.10.xsd
  - pacs.004.001.09.xsd
  - pacs.008.001.08.xsd
  - pacs.009.001.08\_core.xsd
  - pacs.009.001.08\_cov.xsd
  - head.001.001.02\_admi\_004\_fqsm.xsd
  - head.001.001.02\_admi\_004\_rsr.xsd
  - head.001.001.02\_camt\_052.xsd
  - head.001.001.02\_camt\_053.xsd
  - head.001.001.02\_camt\_054\_ca.xsd
  - head.001.001.02\_camt\_054\_cld.xsd
  - head.001.001.02\_camt\_054\_cli.xsd
  - head.001.001.02\_camt\_054\_lpa.xsd
  - head.001.001.02\_camt\_060.xsd
  - head.001.001.02\_mirs\_095.xsd
  - head.001.001.02\_pacs\_002.xsd
  - head.001.001.02\_pacs\_004.xsd
  - head.001.001.02\_pacs\_008.xsd
  - head.001.001.02\_pacs\_009\_core.xsd
  - head.001.001.02\_pacs\_009\_cov.xsd

Note: XML schemas were downloaded from SWIFT MyStandards Readiness CHAPS ENH Portal.

- Trees:  
The trees directory contains the following files:
  - mxvalErrorReport.mtt  
Metadata that represents the xml based structure of the validation report.
  - swiftroute\_funds.mtt  
Metadata that is used as an internal element placeholder.

- Data:  
The data directory contains the following files:

Sample CHAPS ENH valid files for test purpose with header envelope.

- bah\_admi\_004\_001\_02\_fqsm\_valid.xml
- bah\_admi\_004\_001\_02\_rsr\_valid.xml
- bah\_camt\_052\_001\_08\_valid.xml
- bah\_camt\_053\_001\_08\_valid.xml
- bah\_camt\_054\_001\_08\_ca\_valid.xml
- bah\_camt\_054\_001\_08\_cld\_valid.xml
- bah\_camt\_054\_001\_08\_cli\_valid.xml
- bah\_camt\_054\_001\_08\_lpa\_valid.xml
- bah\_camt\_060\_001\_05\_valid.xml
- bah\_mirs\_095\_001\_01\_valid.xml
- bah\_pacs\_002\_001\_10\_valid.xml
- bah\_pacs\_004\_001\_09\_valid.xml
- bah\_pacs\_008\_001\_08\_valid.xml
- bah\_pacs\_009\_001\_08\_core\_valid.xml
- bah\_pacs\_009\_001\_08\_cov\_valid.xml

---

## Run the example in Design Studio

1. Build all the maps in the .mms files listed in What the example contains section.
2. Replace the input card 1 in router map, chen8200\_val (under chaps\_enh\_schema\_validation\_frmwrk\_map\_xsd.mms file) with the desired input file.
3. Run the main router map, chen8200\_val.  
You will see the output file in the data folder called chaps\_error\_report.xml.

---

## Run the example in Design Server User Interface

1. Open a browser pointing to the installation of the IBM Sterling Transformation Extender.
2. Enter the user and password credentials.
3. If the project does not exist, import chaps\_enh.zip.
4. If absent, create a package containing all the Maps, Files and Flows.
5. Build the package in the desired server (to build all maps).
6. Open the project.
7. In the Maps tab, open the map chen8200\_val.
8. Modify the settings for input card 1 on design canvas to point to the desired test file.
9. Save, build, and run the map chen8200\_val.
10. Right click on the output card on canvas and select View data.

---

## EBA EURO1/STEP1 Examples:

- [\*\*EBA E1S1 enhanced MX validation \(using Flow Server\) Example\*\*](#)  
This example demonstrates the EBA EURO1/STEP1 enhanced MX validation for EBA messages using the flow server.
- [\*\*EBA E1S1 enhanced MX validation \(using maps\) Example\*\*](#)  
This example demonstrates the EBA enhanced MX validation for EBA messages using maps only.
- [\*\*EBA E1S1 schema MX validation \(using maps\) Example\*\*](#)  
This example demonstrates the EBA EURO1/STEP1 schema validation for EBA E1S1 messages using maps only.

---

## EBA E1S1 enhanced MX validation (using Flow Server) Example

This example demonstrates the EBA EURO1/STEP1 enhanced MX validation for EBA messages using the flow server.

The flow can perform following level of validation:

- MX extended validation includes following checks:
  - BIC lookup
  - Country code lookup
  - Currency code lookup
  - IBAN format

- Allowed maximum fractional digits per currency
- Usage guideline rules, see mxconfig.xml for list of rules
- Schema validation
- **[What the example contains](#)**  
Files included in this example are as follows:
- **[How to run the example](#)**  
This EBA EURO1/STEP1 MX Validation will use the sample files to demonstrate the generation of validation reports as output from an EBA EURO1/STEP1 XML message.
- **[How to customize the mxconfig.xml file](#)**  
Rules enforced by the MX validation framework can be customized by editing the mxconfig.xml located under the validation/mx\_extended/data folder.

## What the example contains

Files included in this example are as follows:

- eba\_e1s1.zip
  - Test Data Files:
    - e1s1\_pacs\_009.xml
    - e1s1\_pacs\_009\_bad\_rule.xml
    - e1s1\_pacs\_009\_bad\_schema.xml
  - Validation Files:
    - mxconfig.xml
    - bic.xml
    - currencycodedecimals.xml
  - Schemas:
    - bic.xsd  
Metadata that represents the bic.xml repository file structure.
    - ccy.xsd  
Metadata that represents the currencycodedecimals.xml repository file structure.
    - mxconfig.xsd  
Meta data that represents the mxconfig.xml configuration file structure.
    - mxvalErrorReport.mtt  
Metadata that represent the mxconfig.xml configuration file structure.
    - swiftroute\_funds.mtt  
Metadata that is used as an internal element placeholder.

(Based from SWIFT MyStandards Readiness EBA EURO1/STEP1 portal)

- admi.007.001.01.xsd
- camt.003.001.07.xsd
- camt.004.001.08.xsd
- camt.005.001.08.xsd
- camt.006.001.08.xsd
- camt.009.001.07.xsd
- camt.010.001.08.xsd
- camt.011.001.07.xsd
- camt.029.001.09.xsd
- camt.052.001.08.xsd
- camt.053.001.08.xsd
- camt.054.001.08.xsd
- camt.056.001.08.xsd
- camt.998.001.03\_gar.xsd
- camt.998.001.03\_rar.xsd
- sup\_camt.998.001.03\_gar.xsd
- sup\_camt.998.001.03\_rar.xsd
- pacs.002.001.10.xsd
- pacs.004.001.09.xsd
- pacs.008.001.08.xsd
- pacs.009.001.08.xsd
- pacs.010.001.03.xsd
- head.001.001.01.xsd
- Maps:
  - For Extended Validation:
    - e1s17100\_val
    - e1s17001\_camt\_054\_001\_08
    - e1s17002\_camt\_003\_001\_07
    - e1s17003\_camt\_004\_001\_08
    - e1s17004\_camt\_005\_001\_08
    - e1s17005\_camt\_006\_001\_08
    - e1s17006\_camt\_009\_001\_07
    - e1s17007\_camt\_010\_001\_08
    - e1s17008\_camt\_011\_001\_07
    - e1s17009\_camt\_029\_001\_09
    - e1s17010\_camt\_052\_001\_08

- e1s17011\_camt\_053\_001\_08
- e1s17012\_camt\_056\_001\_08
- e1s17013\_camt\_998\_001\_03\_gar
- e1s17014\_sup\_camt\_998\_001\_03\_gar
- e1s17015\_camt\_998\_001\_03\_rar
- e1s17016\_sup\_camt\_998\_001\_03\_rar
- e1s17051\_head111\_camt
- e1s17101\_pacs\_002\_001\_10
- e1s17102\_pacs\_004\_001\_09
- e1s17103\_pacs\_008\_001\_08
- e1s17104\_pacs\_009\_001\_08
- e1s17105\_pacs\_010\_001\_03
- e1s17151\_head111\_pacs
- e1s17501\_admi\_007\_001\_01
- e1s17551\_head111\_admi
- For Schema Validation:
  - e1s17200\_val
  - e1s17201\_camt\_054\_001\_08
  - e1s17202\_camt\_003\_001\_07
  - e1s17203\_camt\_004\_001\_08
  - e1s17204\_camt\_005\_001\_08
  - e1s17205\_camt\_006\_001\_08
  - e1s17206\_camt\_009\_001\_07
  - e1s17207\_camt\_010\_001\_08
  - e1s17208\_camt\_011\_001\_07
  - e1s17209\_camt\_029\_001\_09
  - e1s17210\_camt\_052\_001\_08
  - e1s17211\_camt\_053\_001\_08
  - e1s17212\_camt\_056\_001\_08
  - e1s17213\_camt\_998\_001\_03\_gar
  - e1s17214\_sup\_camt\_998\_001\_03\_gar
  - e1s17215\_camt\_998\_001\_03\_rar
  - e1s17216\_sup\_camt\_998\_001\_03\_rar
  - e1s17301\_pacs\_002\_001\_10
  - e1s17302\_pacs\_004\_001\_09
  - e1s17303\_pacs\_008\_001\_08
  - e1s17304\_pacs\_009\_001\_08
  - e1s17305\_pacs\_010\_001\_03
  - e1s17401\_admi\_007\_001\_01
  - e1s17651\_head111\_camt
  - e1s17651\_head\_001\_001\_01
- Common:
  - mxut1003\_bizsvc\_e1s1
- Flows:
  - eba\_e1s1\_validation\_flow

## How to run the example

This EBA EURO1/STEP1 MX Validation will use the sample files to demonstrate the generation of validation reports as output from an EBA EURO1/STEP1 XML message.

The stopValidation.json file will report the validation failure due to the pre-conversion checks:

**If input file is other than any of the supported EBA EURO1/STEP1 messages as per schema list indicated above.**

1. Import the eba\_e1s1.zip project into the Design Server.
2. Open the eba\_e1s1 project in Design Server and view the flow eba\_e1s1\_validation\_flow.
  - a. The Flow Description points to all the maps to be executed during the flow process, which will be called from within some of the map nodes in the flow. The following is a list of maps invoked using RUN built-in function during the flow process:
    - @packagemap=eba\_e1s1/validation/mx\_extended/maps/e1s1\_mx\_pacs\_validation\_enh/e1s17101\_pacs\_002\_001\_10
    - @packagemap=eba\_e1s1/validation/mx\_extended/maps/e1s1\_mx\_pacs\_validation\_enh/e1s17102\_pacs\_004\_001\_09
    - @packagemap=eba\_e1s1/validation/mx\_extended/maps/e1s1\_mx\_pacs\_validation\_enh/e1s17103\_pacs\_008\_001\_08
    - @packagemap=eba\_e1s1/validation/mx\_extended/maps/e1s1\_mx\_pacs\_validation\_enh/e1s17104\_pacs\_009\_001\_08
    - @packagemap=eba\_e1s1/validation/mx\_extended/maps/e1s1\_mx\_pacs\_validation\_enh/e1s17105\_pacs\_010\_001\_03
    - @packagemap=eba\_e1s1/validation/mx\_extended/maps/e1s1\_mx\_pacs\_validation\_enh/e1s17151\_head111\_pacs
    - @packagemap=eba\_e1s1/validation/mx\_extended/maps/e1s1\_mx\_camt\_validation\_enh/e1s17051\_head111\_camt
    - @packagemap=eba\_e1s1/validation/mx\_extended/maps/e1s1\_mx\_camt\_validation\_enh/e1s17001\_camt\_054\_001\_08
    - @packagemap=eba\_e1s1/validation/mx\_extended/maps/e1s1\_mx\_camt\_validation\_enh/e1s17002\_camt\_003\_001\_07
    - @packagemap=eba\_e1s1/validation/mx\_extended/maps/e1s1\_mx\_camt\_validation\_enh/e1s17003\_camt\_004\_001\_08
    - @packagemap=eba\_e1s1/validation/mx\_extended/maps/e1s1\_mx\_camt\_validation\_enh/e1s17004\_camt\_005\_001\_08
    - @packagemap=eba\_e1s1/validation/mx\_extended/maps/e1s1\_mx\_camt\_validation\_enh/e1s17005\_camt\_006\_001\_08
    - @packagemap=eba\_e1s1/validation/mx\_extended/maps/e1s1\_mx\_camt\_validation\_enh/e1s17006\_camt\_009\_001\_07
    - @packagemap=eba\_e1s1/validation/mx\_extended/maps/e1s1\_mx\_camt\_validation\_enh/e1s17007\_camt\_010\_001\_08
    - @packagemap=eba\_e1s1/validation/mx\_extended/maps/e1s1\_mx\_camt\_validation\_enh/e1s17008\_camt\_011\_001\_07
    - @packagemap=eba\_e1s1/validation/mx\_extended/maps/e1s1\_mx\_camt\_validation\_enh/e1s17009\_camt\_029\_001\_09
    - @packagemap=eba\_e1s1/validation/mx\_extended/maps/e1s1\_mx\_camt\_validation\_enh/e1s17010\_camt\_052\_001\_08
    - @packagemap=eba\_e1s1/validation/mx\_extended/maps/e1s1\_mx\_camt\_validation\_enh/e1s17011\_camt\_053\_001\_08
    - @packagemap=eba\_e1s1/validation/mx\_extended/maps/e1s1\_mx\_camt\_validation\_enh/e1s17012\_camt\_056\_001\_08
    - @packagemap=eba\_e1s1/validation/mx\_extended/maps/e1s1\_mx\_camt\_validation\_enh/e1s17013\_camt\_998\_001\_03\_gar

- @packagemap=eba\_e1s1/validation/mx\_extended/maps/e1s1\_mx\_camt\_validation\_enh/e1s17014\_sup\_camt\_998\_001\_03\_gar
- @packagemap=eba\_e1s1/validation/mx\_extended/maps/e1s1\_mx\_camt\_validation\_enh/e1s17015\_camt\_998\_001\_03\_rar
- @packagemap=eba\_e1s1/validation/mx\_extended/maps/e1s1\_mx\_camt\_validation\_enh/e1s17016\_sup\_camt\_998\_001\_03\_rar
- @packagemap=eba\_e1s1/validation/mx\_extended/maps/e1s1\_mx\_admi\_validation\_enh/e1s17501\_admi\_007\_001\_01
- @packagemap=eba\_e1s1/validation/mx\_extended/maps/e1s1\_mx\_admi\_validation\_enh/e1s17551\_head111\_admi
- @packagemap=eba\_e1s1/validation/schema\_only/maps/e1s1\_mx\_schema\_validation\_xsd/e1s17201\_camt\_054\_001\_08
- @packagemap=eba\_e1s1/validation/schema\_only/maps/e1s1\_mx\_schema\_validation\_xsd/e1s17202\_camt\_003\_001\_07
- @packagemap=eba\_e1s1/validation/schema\_only/maps/e1s1\_mx\_schema\_validation\_xsd/e1s17203\_camt\_004\_001\_08
- @packagemap=eba\_e1s1/validation/schema\_only/maps/e1s1\_mx\_schema\_validation\_xsd/e1s17204\_camt\_005\_001\_08
- @packagemap=eba\_e1s1/validation/schema\_only/maps/e1s1\_mx\_schema\_validation\_xsd/e1s17205\_camt\_006\_001\_08
- @packagemap=eba\_e1s1/validation/schema\_only/maps/e1s1\_mx\_schema\_validation\_xsd/e1s17206\_camt\_009\_001\_07
- @packagemap=eba\_e1s1/validation/schema\_only/maps/e1s1\_mx\_schema\_validation\_xsd/e1s17207\_camt\_010\_001\_08
- @packagemap=eba\_e1s1/validation/schema\_only/maps/e1s1\_mx\_schema\_validation\_xsd/e1s17208\_camt\_011\_001\_07
- @packagemap=eba\_e1s1/validation/schema\_only/maps/e1s1\_mx\_schema\_validation\_xsd/e1s17209\_camt\_029\_001\_09
- @packagemap=eba\_e1s1/validation/schema\_only/maps/e1s1\_mx\_schema\_validation\_xsd/e1s17210\_camt\_052\_001\_08
- @packagemap=eba\_e1s1/validation/schema\_only/maps/e1s1\_mx\_schema\_validation\_xsd/e1s17211\_camt\_053\_001\_08
- @packagemap=eba\_e1s1/validation/schema\_only/maps/e1s1\_mx\_schema\_validation\_xsd/e1s17212\_camt\_056\_001\_08
- @packagemap=eba\_e1s1/validation/schema\_only/maps/e1s1\_mx\_schema\_validation\_xsd/e1s17213\_camt\_998\_001\_03\_gar
- @packagemap=eba\_e1s1/validation/schema\_only/maps/e1s1\_mx\_schema\_validation\_xsd/e1s17214\_sup\_camt\_998\_001\_03\_gar
- @packagemap=eba\_e1s1/validation/schema\_only/maps/e1s1\_mx\_schema\_validation\_xsd/e1s17215\_camt\_998\_001\_03\_rar
- @packagemap=eba\_e1s1/validation/schema\_only/maps/e1s1\_mx\_schema\_validation\_xsd/e1s17216\_sup\_camt\_998\_001\_03\_rar
- @packagemap=eba\_e1s1/validation/schema\_only/maps/e1s1\_mx\_schema\_validation\_xsd/e1s17301\_pacs\_002\_001\_10
- @packagemap=eba\_e1s1/validation/schema\_only/maps/e1s1\_mx\_schema\_validation\_xsd/e1s17302\_pacs\_004\_001\_09
- @packagemap=eba\_e1s1/validation/schema\_only/maps/e1s1\_mx\_schema\_validation\_xsd/e1s17303\_pacs\_008\_001\_08
- @packagemap=eba\_e1s1/validation/schema\_only/maps/e1s1\_mx\_schema\_validation\_xsd/e1s17304\_pacs\_009\_001\_08
- @packagemap=eba\_e1s1/validation/schema\_only/maps/e1s1\_mx\_schema\_validation\_xsd/e1s17305\_pacs\_010\_001\_03
- @packagemap=eba\_e1s1/validation/schema\_only/maps/e1s1\_mx\_schema\_validation\_xsd/e1s17401\_admi\_007\_001\_01
- @packagemap=eba\_e1s1/validation/schema\_only/maps/e1s1\_mx\_schema\_validation\_xsd/e1s17651\_head\_001\_001\_01

b. It utilizes the following nodes:

- Source Nodes:
  - mx\_input  
This node identifies the input data to be validated in the flow. It uses the INPUT\_FILE variable to set the location of the data.
- Map Nodes:
  - MX pre-check  
This node runs map mxut1003\_bizsvc\_e1s1 which sets the flow variables, checks pre-validation conditions and creates infoset.json for validation.
  - EXT\_VAL  
This node runs map e1s17100\_val which calls the appropriate map to perform extended validation of an EBA message.
  - XSD\_VAL  
This node runs map e1s17200\_val which calls the appropriate map to perform schema validation of an EBA message.

Note: The maps e1s17100\_val and e1s17200\_val internally call all the other maps identified in the above Flow Description step.
- Decision Nodes:
  - pre-valid chk  
This node checks the flow variable stopValidation to decide if further validation/processing is not required.
  - EXT\_RESULT\_FORMAT  
This node checks the value of the flow variable REPORT\_FORMAT to determine if the resulting report for extended validation should be generated in XML (default) or JSON.
  - XSD\_RESULT\_FORMAT  
This node checks the value of the flow variable REPORT\_FORMAT to determine if the resulting report for schema-only validation should be generated in XML (default) or JSON.
- Route Nodes:
  - VAL\_TYPE  
This node checks the variable VALIDATION\_TYPE to determine if the process will do extended validation or schema-only validation on the data.
- Log Nodes:
  - FAILURE  
This node creates a log file specified by the flow variable FAILURE\_LOG in case of no validation due to precondition check failures.
- Fail Node:
  - STOP  
It generates error message setup in the node in case of no validation performed.
- Passthrough Nodes
  - EXT\_XML\_CONVERT  
This node receives an extended validation report in XML format and does not modify the content, thus passing it as is to the appropriate target node.
  - XSD\_XML\_CONVERT  
This node receives a schema-only validation report in XML format and does not modify the content, thus passing it as is to the appropriate target node.
- Format Converter Nodes
  - EXT\_JSON\_CONVERT  
This node receives an extended validation report in XML format and converts it to JSON before passing it to the appropriate target node.
  - XSD\_JSON\_CONVERT

This node receives a schema-only validation report in XML format and converts it to JSON before passing it to the appropriate target node.

ix. Target Nodes

- ext\_xml

This node contains the resulting extended validation report in XML format and creates the output as defined by variable OUTPUT\_RESULT\_XML.

- ext\_json

This node contains the resulting extended validation report in JSON format and creates the output as defined by the variable OUTPUT\_RESULT\_JSON.

- xsd\_xml

This node contains the resulting schema-only validation report in XML format and creates the output as defined by the variable OUTPUT\_RESULT\_XML.

- xsd\_json

This node contains the resulting schema-only validation report in JSON format and creates the output as defined by the variable OUTPUT\_RESULT\_JSON.

c. It utilizes the following variables:

Flow variables:

- VALIDATION\_TYPE

The default value is extended.

For schema-only validation, it can be changed to schema. This is used in Route node VAL\_TYPE.

- REPORT\_FORMAT

The default value is xml.

It can be changed to json. This is used in Decision nodes EXT\_RESULT\_FORMAT and XSD\_RESULT\_FORMAT.

- INPUT\_FILE

The default value is ../tools/mxservice/data/e1s1\_pacs\_009.xml.

This is the data file to be used for validation and can be customized. This is used in Source node mx\_input.

- BIC\_FILE

The default value is ../data/bic.xml.

This is the location of the bic cross-reference file that is used in all the maps executed by the map in node EXT\_VAL. It can be customized.

- CCY\_FILE

The default value is ../data/currencycodedecimals.xml.

This is the location of the country code cross-reference file that is used in all the maps executed by the map in node EXT\_VAL. It can be customized.

- MXCONFIG\_FILE

The default value is ../data/mxconfig.xml.

This is the location of the file containing the rule validation settings that is used in all the maps executed by the map in node EXT\_VAL. It can be customized.

- OUTPUT\_RESULT\_XML

The default value is validation\_result.xml.

This is the location of the validation report file in xml format. It is used in Target nodes ext\_xml and xsd\_xml. It can be customized.

- OUTPUT\_RESULT\_JSON

The default value is validation\_result.json.

This is the location of the validation report file in json format. It is used in Target nodes ext\_json and xsd\_json. It can be customized.

- FAILURE\_LOG

The default value is stopMXValidation.json.

This is the location of the failure log. It is used in Log node FAILURE. It can be customized.

- bizSvc

This is for internal use in the Map node MX pre-check to determine the type of data being read in the input file.

- stopValidation

This is for internal use in the Map node MX pre-check and is checked in Decision node pre-valid chk to cause a Failure log in case a data file was not recognized as a valid EBA message.

3. Open the main flow eba\_e1s1\_validation\_flow in the Design Server. It utilizes above one source node, three map nodes, one route node, three decision nodes, one log node, one fail node, two passthrough nodes, two format converter nodes and four target nodes.

4. In Design Server, create a package eba\_e1s1\_mx\_validation that contains the input files and one flow eba\_e1s1\_validation\_flow. The maps will automatically be included during deployment of the package onto the runtime server.

Note: Not required for running flows on Design Server user interface directly.

- [Run the example using three different methods](#)

You can run the example using three different methods:

---

## Run the example using three different methods

You can run the example using three different methods:

1. Running the example from the Design Server user interface.
2. Deploying the example to tx-rest and running it using the Swagger interface.
3. Deploying locally or to ftp server and using the flow command server process (flowcmdserver).

- [Run the example from the Design Server user interface](#)

Use the following steps to run the example from the Design Server user interface:

- [Run the example using the Swagger interface](#)

To run the example, deploy to tx-rest and run it using the Swagger interface.

- [Run the example using the flow command server process \(flowcmdserver\)](#)

To run the example, deploy locally or to ftp server and use the flow command server process (flowcmdserver).

---

## Run the example from the Design Server user interface

Use the following steps to run the example from the Design Server user interface:

1. In the Design Server, open eba\_e1s1\_validation\_flow flow and click on Run button.
2. Set toggle switch Run On Design Server and select flow input file. Example is e1s1\_pacs\_009.xml.  
Note: If a default path is assigned to variable INPUT\_FILE, not selecting an input file will use the value defined for the variable.  
Flow will run and report with the green check box. The report lists the execution of all nodes.
3. To examine the data passed on each link, right click on each link and select View Link Data.
4. To view the target node, right click on the node input icon and select View Data.

---

## Run the example using the Swagger interface

To run the example, deploy to tx-rest and run it using the Swagger interface.

1. Create a Web type server definition where the runtime tx-rest is installed if you do not have a server definition to deploy in Design Server.
2. If not running, start tx-rest on the server: <DTX\_HOME>/restapi/tomcat/dtxtomcat start tx-rest.
3. Deploy the created package to the server definition in Design Server.
4. Bring up the tx-rest Swagger UI: <DTX\_HOME>/restapi/tomcat/dtxtomcat showdocs tx-rest.
5. In the Swagger Explore window, enter /tx-rest/v2/docs and click on the Explore button to view the deployed package.  
You should see a Miscellaneous section having two actions: A **PUT** /v2/run/eba\_e1s1\_validation\_flow and a **POST** /v2/run/eba\_e1s1\_validation\_flow.
6. In the **PUT** action:
  - a. Expand the **PUT** action and select the Try it out button.
  - b. Leave the input and output files sections empty.
  - c. Under flow\_vars section, delete the entire content and replace it with the commands to run the flow. For example:

```
{
 "INPUT_FILE": "C:/mydir/data/e1s1_pacs_009.xml",
 "OUTPUT_RESULT_JSON": "C:/mydir/data/validation_result.json",
 "REPORT_FORMAT": "json",
 "VALIDATION_TYPE": "extended"
}

or

{
 "INPUT_FILE": "C:/mydir/data/e1s1_pacs_009.xml",
 "OUTPUT_RESULT_XML": "C:/mydir/data/validation_result.xml",
 "REPORT_FORMAT": "xml",
 "VALIDATION_TYPE": "extended"
}
```

- d. Click Execute blue bar for running the flow. When flow completes execution, 200 response code is shown in the Server Response section. This run should generate the output same as above.
7. Refresh the browser to delete the deployed package and get the Swagger Explore back to /tx-rest/openapi.json.  
There will be a DELETE /v2/packages/{name} action.
8. Expand the DELETE /v2/packages/{name} action and select Try it out.
9. In the Name field, enter the name you gave to the created package.
10. Select the true option from the stop query drop down, select the blue Execute bar.

You can see a response of 204 with a timestamp, if it is successful.

---

## Run the example using the flow command server process (flowcmdserver)

To run the example, deploy locally or to ftp server and use the flow command server process (flowcmdserver).

1. Create a server definition where the runtime is installed, if you do not have a server local definition or ftp execution definition to deploy to, in Design Server.
2. In Design Server, deploy the created package to the server definition. For example, deploy to c:\ftpserver\deployment directory.
3. Execute the flow using the flow command server. The below command is for EBA EURO1/STEP1 e1s1\_pacs\_009.xml message.

```
--flow eba_e1s1_validation_flow.json
--var "INPUT_FILE=C:/ftpserver/deployment/tools/mx_service/data/e1s1_pacs_009.xml"
--audit "C:/ftpserver/deployment/tools/mx_service/data/e1s1_pacs_009.adt.json" -ad
```

The results will be in C:\ftpserver\deployment\flows as validation\_results.xml or validation\_results.json depending on the value of the REPORT\_FORMAT variable.

4. Use the following command to execute with the variable VALIDATION\_TYPE value defined in the Flow settings in Design Server (extended or schema).

```
--flow eba_e1s1_validation_flow.json
--var "INPUT_FILE=C:\ftpserver\deployment\tools\mx_service\data\e1s1_pacs_009.xml"
--var "VALIDATION_TYPE=schema"
--var "REPORT_FORMAT=json"
--audit "C:\ftpserver\deployment\tools\mx_service\data\e1s1_pacs_009_adt.json" -ad
```

5. The flow will report status similar to:

```
***Starting flow command server

Flow UUID: 85eabeb2c-2302-4543-b347-9e18d65c6816
The flow audit file is: "C:\\\\ftpserver\\\\deployment\\\\tools\\\\mx_service\\\\data\\\\e1s1_pacs_009_adt.json"
Flow completed successfully
Elapsed time: 3048ms
```

6. Examine the flow output result file validation\_results.xml and the audit file e1s1\_pacs\_009\_adt.json.

## How to customize the mxconfig.xml file

Rules enforced by the MX validation framework can be customized by editing the mxconfig.xml located under the validation/mx\_extended/data folder.

All the rules enforced by the MX validation map are under element ExtendedValidation.

Any of the sub-elements under ExtendedValidation can be enabled by setting the value to T or disabled by setting to F.

### Example

To disable the BIC lookup validation feature.

### About this task

Change sub-element ExtendedValidation/BICValidation value from T to F.

### Procedure

1. Download the mxconfig.xml file from the Files tab.
2. Open the mxconfig.xml file and edit the sub-element ExtendedValidation/BICValidation value setting from T to F, then save the file.
3. Upload the edited mxconfig.xml file to the project.
4. Run the flow.

Run using an input with BIC lookup validation issue.

Right click on the output card on canvas and select View data. No BIC lookup validation reported.

## EBA E1S1 enhanced MX validation (using maps) Example

This example demonstrates the EBA enhanced MX validation for EBA messages using maps only.

The maps can perform following level of validation:

- MX enhanced validation includes following checks:
  - BIC lookup
  - Country code lookup
  - Currency code lookup
  - IBAN format
  - Allowed maximum fractional digits per currency
  - Usage guideline rules, see mxconfig.xml for list of rules
- Schema validation:
  - Enforces validation based on XSD.
- [What the example contains](#)  
Files and directories included in this example are as follows:
- [Run the example in Design Studio](#)
- [Run the example in Design Server User Interface](#)
- [How to customize the mxconfig.xml file](#)

Rules enforced by the MX validation framework can be customized by editing the mxconfig.xml located under the validation/mx\_extended/data folder.

### What the example contains

Files and directories included in this example are as follows:

- Maps:  
The maps directory contains the following map sources to use when running under Design Studio.

- e1s1\_mx\_camt\_validation\_enh.mms  
Utility map called by main map for all the camt EBA E1S1 xml message MX validation.
- e1s1\_mx\_pacs\_validation\_enh.mms  
Utility map called by main map for all the pacs EBA E1S1 xml message MX validation.
- e1s1\_mx\_admi\_validation\_enh.mms  
Utility map called by main map for all the admi EBA E1S1 xml message MX validation.
- e1s1\_mx\_validation\_frmwrk\_map\_enh.mms  
The Main map used to apply MX validation to EBA E1S1 xml messages.

Note: When running under Design Server, the main framework map is e1s1\_mx\_validation\_frmwrk\_map\_flw\_enh.mms

- Schemas:

The schemas directory contains the following files:

- bic.xsd  
Metadata that represents the bic.xml repository file structure.
- ccy.xsd  
Metadata that represents the currencycodedecimals.xml repository file structure.
- mxconfig.xsd  
Metadata that represents the mxconfig.xml configuration file structure.

(Based from SWIFT MyStandards Readiness EBA Portal)

- admi.007.001.01.xsd
- camt.003.001.07.xsd
- camt.004.001.08.xsd
- camt.005.001.08.xsd
- camt.006.001.08.xsd
- camt.009.001.07.xsd
- camt.010.001.08.xsd
- camt.011.001.07.xsd
- camt.029.001.09.xsd
- camt.052.001.08.xsd
- camt.053.001.08.xsd
- camt.054.001.08.xsd
- camt.056.001.08.xsd
- camt.998.001.03\_gar.xsd
- camt.998.001.03\_rar.xsd
- camt.998.001.03\_gar.xsd
- sup\_camt.998.001.03\_gar.xsd
- sup\_camt.998.001.03\_rar.xsd
- pacs.002.001.10.xsd
- pacs.004.001.09.xsd
- pacs.008.001.08.xsd
- pacs.009.001.08.xsd
- pacs.010.001.03.xsd
- head.001.001.01.xsd

- Trees:

The trees directory contains the following files:

- swiftroute\_funds.mtt  
Metadata that is used as an internal element placeholder.
- mxvalErrorReport.mtt  
Metadata that represents the xml based structure of the validation report.

- Data:

The data directory contains the following file:

- bic.xml  
Repository file listing all BICs which are used during validation.
- currencycodedecimals.xml  
Repository file listing country codes, currency codes and corresponding maximum fractionally digits, used as reference for validation.
- mxconfig.xml  
Contains the MX configuration information on how to process the message.

Sample EBA EURO1/STEP1 valid files without header envelope for test purposes:

- admi\_007\_001\_01\_valid.xml

Sample EBA EURO1/STEP1 valid files with header envelope for test purposes:

- bah\_admi\_007\_001\_01\_valid.xml
- bah\_camt\_003\_001\_07\_valid.xml
- bah\_camt\_004\_001\_08\_valid.xml
- bah\_camt\_005\_001\_08\_valid.xml
- bah\_camt\_006\_001\_08\_valid.xml
- bah\_camt\_009\_001\_07\_valid.xml
- bah\_camt\_010\_001\_08\_valid.xml

- bah\_camt\_011\_001\_07\_valid.xml
  - bah\_camt\_029\_001\_09\_valid.xml
  - bah\_camt\_052\_001\_08\_valid.xml
  - bah\_camt\_053\_001\_08\_valid.xml
  - bah\_camt\_054\_001\_08\_valid.xml
  - bah\_camt\_056\_001\_08\_valid.xml
  - bah\_camt\_998\_001\_03\_gar\_valid.xml
  - bah\_camt\_998\_001\_03\_rar\_valid.xml
  - bah\_pacs\_002\_001\_10\_valid.xml
  - bah\_pacs\_004\_001\_09\_valid.xml
  - bah\_pacs\_008\_001\_08\_valid.xml
  - bah\_pacs\_009\_001\_08\_valid.xml
  - bah\_pacs\_010\_001\_03\_valid.xml
- 

## Run the example in Design Studio

1. Build all the maps in the .mms files listed in What the example contains section.
  2. Replace the input card 1 in router map, e1s17000\_val (under e1s1\_mx\_validation\_frmwrk\_map\_enh.mms file or e1s1\_mx\_validation\_frmwrk\_map\_flw\_enh.mms file if running under Design Server) with the desired input file.
  3. Run the main router map, e1s17000\_val.  
You will see the output file in the data folder called e1s1\_error\_report.xml.
- 

## Run the example in Design Server User Interface

1. Open a browser pointing to the installation of the IBM Sterling Transformation Extender.
  2. Enter the user and password credentials.
  3. If the project does not exist, import eba\_e1s1.zip.
  4. If absent, create a package containing all the Maps, Files and Flows.
  5. Build the package in the desired server (to build all maps).
  6. Open the project.
  7. In the Maps tab, open the map e1s17000\_val.
  8. Modify the settings for input card 1 on design canvas to point to the desired test file.
  9. Save, build, and run the map e1s17000\_val.
  10. Right click on the output card on canvas and select View data.
- 

## How to customize the mxconfig.xml file

Rules enforced by the MX validation framework can be customized by editing the mxconfig.xml located under the validation/mx\_extended/data folder.

All the rules enforced by the MX validation map are under element ExtendedValidation.

Any of the sub-elements under ExtendedValidation can be enabled by setting the value to T or disabled by setting to F.

### Example

To disable the BIC lookup validation feature.

### About this task

Change sub-element ExtendedValidation/BICValidation value from T to F.

### Procedure

1. Download the mxconfig.xml file from the Files tab.
2. Open the mxconfig.xml file and edit the sub-element ExtendedValidation/BICValidation value setting from T to F, then save the file.
3. Upload the edited mxconfig.xml file to the project.
4. Run the map.  
Run using an input with BIC lookup validation issue.

Right click on the output card on canvas and select View data. No BIC lookup validation reported.

## EBA E1S1 schema MX validation (using maps) Example

This example demonstrates the EBA EURO1/STEP1 schema validation for EBA E1S1 messages using maps only.

The maps can perform following level of validation:

- Schema validation

- [What the example contains](#)

Files and directories included in this example are as follows:

- [Run the example in Design Studio](#)

- [Run the example in Design Server User Interface](#)

---

## What the example contains

Files and directories included in this example are as follows:

- Maps:

The maps directory contains the following map sources to use when running under Design Studio:

- e1s1\_mx\_schema\_validation\_xsd.mms  
Utility maps called by main map for all EBA E1S1 messages schema validation.
- e1s1\_mx\_schema\_validation\_frmwrk\_map\_xsd.mms  
The Main map used to apply schema validation to EBA E1S1 xml messages.

- Schemas:

The schemas directory contains the following files:

(Based from SWIFT MyStandards Readiness EBA Portal)

- admi.007.001.01.xsd
- camt.003.001.07.xsd
- camt.004.001.08.xsd
- camt.005.001.08.xsd
- camt.006.001.08.xsd
- camt.009.001.07.xsd
- camt.010.001.08.xsd
- camt.011.001.07.xsd
- camt.029.001.09.xsd
- camt.052.001.08.xsd
- camt.053.001.08.xsd
- camt.054.001.08.xsd
- camt.056.001.08.xsd
- camt.998.001.03\_gar.xsd
- camt.998.001.03\_rar.xsd
- camt.998.001.03\_gar.xsd
- sup\_camt.998.001.03\_gar.xsd
- sup\_camt.998.001.03\_rar.xsd
- pacs.002.001.10.xsd
- pacs.004.001.09.xsd
- pacs.008.001.08.xsd
- pacs.009.001.08.xsd
- pacs.010.001.03.xsd
- head.001.001.01.xsd

- Trees:

The trees directory contains the following files:

- swiftroute\_funds.mtt  
Metadata that is used as an internal element placeholder.
- mxvalErrorReport.mtt  
Metadata that represents the xml based structure of the validation report.

- Data:

The data directory contains the following files:

Sample EBA EURO1/STEP1 valid files without header envelope for test purposes:

- admi\_007\_001\_01\_valid.xml

Sample EBA EURO1/STEP1 valid files with header envelope for test purposes:

- bah\_admi\_007\_001\_01\_valid.xml
- bah\_camt\_003\_001\_07\_valid.xml
- bah\_camt\_004\_001\_08\_valid.xml
- bah\_camt\_005\_001\_08\_valid.xml
- bah\_camt\_006\_001\_08\_valid.xml
- bah\_camt\_009\_001\_07\_valid.xml
- bah\_camt\_010\_001\_08\_valid.xml
- bah\_camt\_011\_001\_07\_valid.xml
- bah\_camt\_029\_001\_09\_valid.xml
- bah\_camt\_052\_001\_08\_valid.xml
- bah\_camt\_053\_001\_08\_valid.xml
- bah\_camt\_054\_001\_08\_valid.xml
- bah\_camt\_056\_001\_08\_valid.xml
- bah\_camt\_998\_001\_03\_gar\_valid.xml
- bah\_camt\_998\_001\_03\_rar\_valid.xml
- bah\_pacs\_002\_001\_10\_valid.xml

- bah\_pacs\_004\_001\_09\_valid.xml
  - bah\_pacs\_008\_001\_08\_valid.xml
  - bah\_pacs\_009\_001\_08\_valid.xml
  - bah\_pacs\_010\_001\_03\_valid.xml
- 

## Run the example in Design Studio

1. Build all the maps in the .mms files listed in What the example contains section.
  2. Replace the input card 1 in router map, e1s17200\_val (under e1s1\_schema\_validation\_frmwrk\_map\_xsd.mms file) with desired input file.
  3. Run the main router map, e1s17200\_val.  
You will see the output file in the data folder called e1s1\_error\_report.xml.
- 

## Run the example in Design Server User Interface

1. Open a browser pointing to the installation of the IBM Sterling Transformation Extender.
  2. Enter the user and password credentials.
  3. If the project does not exist, import eba\_e1s1.zip.
  4. If absent, create a package containing all the Maps, Files and Flows.
  5. Build the package in the desired server (to build all maps).
  6. Open the project.
  7. In the Maps tab, open the map e1s17200\_val.
  8. Modify the settings for input card 1 on design canvas to point to the desired test file.
  9. Save, build, and run the map e1s17200\_val.
  10. Right click on the output card on canvas and select View data.
- 

## NBOR ReGIS Examples:

- [\*\*NBOR ReGIS schema validation \(using Flow Server\) example\*\*](#)  
This example demonstrates the NBOR ReGIS schema validation for NBOR ReGIS messages using flow server.
  - [\*\*NBOR ReGIS schema validation \(using maps\) example\*\*](#)  
This example demonstrates the schema validation for NBOR ReGIS messages using maps only.
- 

## NBOR ReGIS schema validation (using Flow Server) example

This example demonstrates the NBOR ReGIS schema validation for NBOR ReGIS messages using flow server.

The flow can perform following level of validation:

- Schema validation
  - [\*\*What the example contains\*\*](#)  
Files included in this example are as follows:
  - [\*\*How to run the example\*\*](#)  
This NBOR ReGIS schema Validation will use the sample files to demonstrate the generation of validation report as output from a NBOR ReGIS XML message.
- 

## What the example contains

Files included in this example are as follows:

- nbor\_regis.zip
  - Test Data Files:
    - rgis\_pacs\_009\_cov\_bad\_schema.xml
    - rgis\_pacs\_009\_cov.xml
  - Schemas:
    - mxvalErrorReport.mtt  
Metadata that represents the xml based structure of the validation report.
    - swiftroute\_funds.mtt  
Metadata that is used as an internal element placeholder.
    - XML schemas based on SWIFT MyStandards Readiness NBOR ReGIS portal:
      - admi.004.001.02.xsd
      - camt.005.001.08.xsd
      - camt.006.001.08.xsd
      - camt.007.001.08.xsd
      - camt.008.001.08.xsd
      - camt.025.001.05.xsd

- camt.029.001.09.xsd
- camt.046.001.05.xsd
- camt.047.001.06.xsd
- camt.052.001.08.xsd
- camt.053.001.08.xsd
- camt.054.001.08.xsd
- camt.056.001.08.xsd
- camt.060.001.05.xsd
- pacs.002.001.10.xsd
- pacs.004.001.09.xsd
- pacs.008.001.08.xsd
- pacs.008.001.08\_TREZROBU.xsd
- pacs.009.001.08.xsd
- pacs.009.001.08\_cov.xsd
- pacs.029.001.01.xsd
- head.001.001.02\_admi\_004.xsd
- head.001.001.02\_camt\_005.xsd
- head.001.001.02\_camt\_006.xsd
- head.001.001.02\_camt\_007.xsd
- head.001.001.02\_camt\_008.xsd
- head.001.001.02\_camt\_025.xsd
- head.001.001.02\_camt\_029.xsd
- head.001.001.02\_camt\_046.xsd
- head.001.001.02\_camt\_047.xsd
- head.001.001.02\_camt\_052.xsd
- head.001.001.02\_camt\_053.xsd
- head.001.001.02\_camt\_054.xsd
- head.001.001.02\_camt\_056.xsd
- head.001.001.02\_camt\_060.xsd
- head.001.001.02\_pacs\_002.xsd
- head.001.001.02\_pacs\_004.xsd
- head.001.001.02\_pacs\_008.xsd
- head.001.001.02\_pacs\_008\_TREZROBU.xsd
- head.001.001.02\_pacs\_009.xsd
- head.001.001.02\_pacs\_009\_cov.xsd
- head.001.001.02\_pacs\_029.xsd

○ Maps:

- For Schema Validation:
  - rgis1500\_val
  - rgis1501\_admi\_004\_001\_02
  - rgis1502\_camt\_005\_001\_08
  - rgis1503\_camt\_006\_001\_08
  - rgis1504\_camt\_007\_001\_08
  - rgis1505\_camt\_008\_001\_08
  - rgis1506\_camt\_025\_001\_05
  - rgis1507\_camt\_029\_001\_09
  - rgis1508\_camt\_046\_001\_05
  - rgis1509\_camt\_047\_001\_06
  - rgis1510\_camt\_052\_001\_08
  - rgis1511\_camt\_053\_001\_08
  - rgis1512\_camt\_054\_001\_08
  - rgis1513\_camt\_056\_001\_08
  - rgis1514\_camt\_060\_001\_05
  - rgis1515\_pacs\_002\_001\_10
  - rgis1516\_pacs\_004\_001\_09
  - rgis1517\_pacs\_008\_001\_08\_trz
  - rgis1518\_pacs\_008\_001\_08
  - rgis1519\_pacs\_009\_001\_08\_cov
  - rgis1520\_pacs\_009\_001\_08
  - rgis1521\_pacs\_029\_001\_01
  - rgis1551\_head112\_admi004
  - rgis1552\_head112\_camt005
  - rgis1553\_head112\_camt006
  - rgis1554\_head112\_camt007
  - rgis1555\_head112\_camt008
  - rgis1556\_head112\_camt025
  - rgis1557\_head112\_camt029
  - rgis1558\_head112\_camt046
  - rgis1559\_head112\_camt047
  - rgis1560\_head112\_camt052
  - rgis1561\_head112\_camt053
  - rgis1562\_head112\_camt054
  - rgis1563\_head112\_camt056
  - rgis1564\_head112\_camt060
  - rgis1565\_head112\_pacs002
  - rgis1566\_head112\_pacs004
  - rgis1567\_head112\_pacs008\_trz
  - rgis1568\_head112\_pacs008
  - rgis1569\_head112\_pacs009\_cov
  - rgis1570\_head112\_pacs009

- rgis1571\_head112\_pacs029
- Common:
  - mxut1009\_bizsvc\_rgis
- Flows:
  - nbor\_rgis\_validation\_flow

## How to run the example

This NBOR ReGIS schema Validation will use the sample files to demonstrate the generation of validation report as output from a NBOR ReGIS XML message.

The stopValidation.json file will report the validation failure due to the pre-conversion checks:

**If input file is other than any of the supported NBOR ReGIS messages as per schema list indicated above.**

1. Import the nbor\_rgis.zip project into the Design Server.
2. Open the nbor\_rgis project in Design Server and view the flow nbor\_rgis\_validation\_flow.

a. The Flow Description points to all the maps to be executed during the flow process, which will be called from within some of the map nodes in the flow. This list has to be added to the description and ensure all lines start in column 1. Following is the list of maps with the description format:

- @packagemap=nbor\_rgis/validation/schema\_only/maps/nbor\_rgis\_mx\_schema\_validation\_xsd/rgis1501\_admi\_004\_001\_02
- @packagemap=nbor\_rgis/validation/schema\_only/maps/nbor\_rgis\_mx\_schema\_validation\_xsd/rgis1502\_camt\_005\_001\_08
- @packagemap=nbor\_rgis/validation/schema\_only/maps/nbor\_rgis\_mx\_schema\_validation\_xsd/rgis1503\_camt\_006\_001\_08
- @packagemap=nbor\_rgis/validation/schema\_only/maps/nbor\_rgis\_mx\_schema\_validation\_xsd/rgis1504\_camt\_007\_001\_08
- @packagemap=nbor\_rgis/validation/schema\_only/maps/nbor\_rgis\_mx\_schema\_validation\_xsd/rgis1505\_camt\_008\_001\_08
- @packagemap=nbor\_rgis/validation/schema\_only/maps/nbor\_rgis\_mx\_schema\_validation\_xsd/rgis1506\_camt\_025\_001\_05
- @packagemap=nbor\_rgis/validation/schema\_only/maps/nbor\_rgis\_mx\_schema\_validation\_xsd/rgis1507\_camt\_029\_001\_09
- @packagemap=nbor\_rgis/validation/schema\_only/maps/nbor\_rgis\_mx\_schema\_validation\_xsd/rgis1508\_camt\_046\_001\_05
- @packagemap=nbor\_rgis/validation/schema\_only/maps/nbor\_rgis\_mx\_schema\_validation\_xsd/rgis1509\_camt\_047\_001\_06
- @packagemap=nbor\_rgis/validation/schema\_only/maps/nbor\_rgis\_mx\_schema\_validation\_xsd/rgis1510\_camt\_052\_001\_08
- @packagemap=nbor\_rgis/validation/schema\_only/maps/nbor\_rgis\_mx\_schema\_validation\_xsd/rgis1511\_camt\_053\_001\_08
- @packagemap=nbor\_rgis/validation/schema\_only/maps/nbor\_rgis\_mx\_schema\_validation\_xsd/rgis1512\_camt\_054\_001\_08
- @packagemap=nbor\_rgis/validation/schema\_only/maps/nbor\_rgis\_mx\_schema\_validation\_xsd/rgis1513\_camt\_056\_001\_08
- @packagemap=nbor\_rgis/validation/schema\_only/maps/nbor\_rgis\_mx\_schema\_validation\_xsd/rgis1514\_camt\_060\_001\_05
- @packagemap=nbor\_rgis/validation/schema\_only/maps/nbor\_rgis\_mx\_schema\_validation\_xsd/rgis1515\_pacs\_002\_001\_10
- @packagemap=nbor\_rgis/validation/schema\_only/maps/nbor\_rgis\_mx\_schema\_validation\_xsd/rgis1516\_pacs\_004\_001\_09
- @packagemap=nbor\_rgis/validation/schema\_only/maps/nbor\_rgis\_mx\_schema\_validation\_xsd/rgis1517\_pacs\_008\_001\_08\_trz
- @packagemap=nbor\_rgis/validation/schema\_only/maps/nbor\_rgis\_mx\_schema\_validation\_xsd/rgis1518\_pacs\_008\_001\_08
- @packagemap=nbor\_rgis/validation/schema\_only/maps/nbor\_rgis\_mx\_schema\_validation\_xsd/rgis1519\_pacs\_009\_001\_08\_cov
- @packagemap=nbor\_rgis/validation/schema\_only/maps/nbor\_rgis\_mx\_schema\_validation\_xsd/rgis1520\_pacs\_009\_001\_08
- @packagemap=nbor\_rgis/validation/schema\_only/maps/nbor\_rgis\_mx\_schema\_validation\_xsd/rgis1521\_pacs\_029\_001\_01
- @packagemap=nbor\_rgis/validation/schema\_only/maps/nbor\_rgis\_mx\_schema\_validation\_xsd/rgis1551\_head112\_admi004
- @packagemap=nbor\_rgis/validation/schema\_only/maps/nbor\_rgis\_mx\_schema\_validation\_xsd/rgis1552\_head112\_camt005
- @packagemap=nbor\_rgis/validation/schema\_only/maps/nbor\_rgis\_mx\_schema\_validation\_xsd/rgis1553\_head112\_camt006
- @packagemap=nbor\_rgis/validation/schema\_only/maps/nbor\_rgis\_mx\_schema\_validation\_xsd/rgis1554\_head112\_camt007
- @packagemap=nbor\_rgis/validation/schema\_only/maps/nbor\_rgis\_mx\_schema\_validation\_xsd/rgis1555\_head112\_camt008
- @packagemap=nbor\_rgis/validation/schema\_only/maps/nbor\_rgis\_mx\_schema\_validation\_xsd/rgis1556\_head112\_camt025
- @packagemap=nbor\_rgis/validation/schema\_only/maps/nbor\_rgis\_mx\_schema\_validation\_xsd/rgis1557\_head112\_camt029
- @packagemap=nbor\_rgis/validation/schema\_only/maps/nbor\_rgis\_mx\_schema\_validation\_xsd/rgis1558\_head112\_camt046
- @packagemap=nbor\_rgis/validation/schema\_only/maps/nbor\_rgis\_mx\_schema\_validation\_xsd/rgis1559\_head112\_camt047
- @packagemap=nbor\_rgis/validation/schema\_only/maps/nbor\_rgis\_mx\_schema\_validation\_xsd/rgis1560\_head112\_camt052
- @packagemap=nbor\_rgis/validation/schema\_only/maps/nbor\_rgis\_mx\_schema\_validation\_xsd/rgis1561\_head112\_camt053
- @packagemap=nbor\_rgis/validation/schema\_only/maps/nbor\_rgis\_mx\_schema\_validation\_xsd/rgis1562\_head112\_camt054
- @packagemap=nbor\_rgis/validation/schema\_only/maps/nbor\_rgis\_mx\_schema\_validation\_xsd/rgis1563\_head112\_camt056
- @packagemap=nbor\_rgis/validation/schema\_only/maps/nbor\_rgis\_mx\_schema\_validation\_xsd/rgis1564\_head112\_camt060
- @packagemap=nbor\_rgis/validation/schema\_only/maps/nbor\_rgis\_mx\_schema\_validation\_xsd/rgis1565\_head112\_pacs002
- @packagemap=nbor\_rgis/validation/schema\_only/maps/nbor\_rgis\_mx\_schema\_validation\_xsd/rgis1566\_head112\_pacs004
- @packagemap=nbor\_rgis/validation/schema\_only/maps/nbor\_rgis\_mx\_schema\_validation\_xsd/rgis1567\_head112\_pacs008\_trz
- @packagemap=nbor\_rgis/validation/schema\_only/maps/nbor\_rgis\_mx\_schema\_validation\_xsd/rgis1568\_head112\_pacs008
- @packagemap=nbor\_rgis/validation/schema\_only/maps/nbor\_rgis\_mx\_schema\_validation\_xsd/rgis1569\_head112\_pacs009\_cov
- @packagemap=nbor\_rgis/validation/schema\_only/maps/nbor\_rgis\_mx\_schema\_validation\_xsd/rgis1570\_head112\_pacs009
- @packagemap=nbor\_rgis/validation/schema\_only/maps/nbor\_rgis\_mx\_schema\_validation\_xsd/rgis1571\_head112\_pacs029

b. It utilizes the following nodes:

- i. Source Nodes:

- mx\_input

This node identifies the input data to be validated in the flow. It uses the variable INPUT\_FILE to set the location of the data.

- ii. Map Nodes:

- MX pre-check

Runs map mxut1009\_bizsvc\_rgis which sets flow variables, checks pre-validation conditions, and creates infoset.json for validation.

- XSD\_VAL

Runs map rgis1500\_val which calls the appropriate map to perform schema validation of a NBOR ReGIS message.

Note: The map rgis1500\_val internally calls all the other maps identified in the above Flow Description step.

- iii. Decision Nodes:

- pre-valid chk

This node checks the flow variable stopTranslation to decide if further validation/processing is not required.

- XSD\_RESULT\_FORMAT

This node checks the value of the flow variable REPORT\_FORMAT to determine if the resulting report for schema-only validation should be generated in XML (default) or JSON.

iv. Log Nodes:

- FAILURE  
In case of no validation due to precondition check failures, this node creates a log file specified by the flow variable FAILURE\_LOG.

v. Fail Node:

- STOP  
It generates error message setup in the node in case of no validation performed.

vi. Passthrough Nodes:

- XSD\_XML\_CONVERT  
This node receives a schema-only validation report in XML format and does not modify the content, thus passing it as is to the appropriate target node.

vii. Format Converter Nodes:

- XSD\_JSON\_CONVERT  
This node receives a schema-only validation report in XML format and converts it to JSON before passing it to the appropriate target node.

viii. Target Nodes:

- xsd\_xml  
This node contains the resulting schema-only validation report in XML format and creates the output as defined by the variable OUTPUT\_RESULT\_XML.

- xsd\_json  
This node contains the resulting schema-only validation report in JSON format and creates the output as defined by the variable OUTPUT\_RESULT\_JSON.

c. It utilizes the following variables:

i. Flow variables:

- VALIDATION\_TYPE  
Default value is schema. The value extended is not supported at this time. This variable is currently not being used in any node.
- REPORT\_FORMAT  
Default value is xml. It can be changed to json. This is used in Decision node XSD\_RESULT\_FORMAT.
- INPUT\_FILE  
Default value is ../tools/mx\_service/data/rgis\_pacs\_009\_cov.xml. This is the data file to be used for validation and can be customized. It is used in the Source node mx\_input.
- OUTPUT\_RESULT\_XML  
Default value is validation\_result.xml. This is the location of the validation report file in xml format. It is used in Target node xsd\_xml. It can be customized.
- OUTPUT\_RESULT\_JSON  
Default value is validation\_result.json. This is the location of the validation report file in json format. It is used in Target node xsd\_json. It can be customized.
- FAILURE\_LOG  
Default value is stopMXValidation.json. This is the location of the failure log. It is used in Log node FAILURE. It can be customized.
- bizSvc  
For internal use in the Map node MX pre-check to determine the type of data being read in the input file.
- stopValidation  
For internal use in the Map node MX pre-check and is checked in Decision node pre-valid chk to cause a Failure log in case a data file was not recognized as a valid NBOR ReGIS message.

3. Open the main flow nbor\_regis\_validation\_flow in the Design Server. It utilizes above one Source node, two Map nodes, two Decision nodes, one Log node, one Fail node, one Passthrough node, one Format Converter node, and two Target nodes.

4. In Design Server, create a package nbor\_regis\_mx\_validation that contains the input files and one flow nbor\_regis\_validation\_flow. The maps will automatically be included during deployment of the package onto the runtime server.

Note: Not required for running flows on Design Server user interface directly.

• [Run the example using three different methods](#)

You can run the example using three different methods:

---

## Run the example using three different methods

You can run the example using three different methods:

1. Running the example from the Design Server user interface.
2. Deploying the example to tx-rest and running it using the Swagger interface.
3. Deploying locally or to ftp server and using the flow command server process (flowcmdserver).

• [Run the example from the Design Server user interface](#)

Use the following steps to run the example from the Design Server user interface:

• [Run the example using the Swagger interface](#)

To run the example, deploy to tx-rest and run it using the Swagger interface.

- [Run the example using the flow command server process \(flowcmdserver\)](#)

To run the example, deploy locally or to ftp server and use the flow command server process (flowcmdserver).

## Run the example from the Design Server user interface

Use the following steps to run the example from the Design Server user interface:

1. In the Design Server, open nbor\_regs\_validation\_flow flow and click Run button.
2. Set the toggle switch Run On Design Server and select flow input file. Example is rgis\_pacs\_009\_cov.xml.  
Note: If a default path is assigned to the variable INPUT\_FILE, the value defined for the variable will be used if an input file is not selected.  
Flow will run and report with the green check box. The report lists execution of all nodes.
3. Right click and select View Link Data on each link to examine the data.
4. View target node by clicking on the node, doing right click on the Input icon and selecting View Data.

## Run the example using the Swagger interface

To run the example, deploy to tx-rest and run it using the Swagger interface.

1. Create a Web type server definition where the runtime tx-rest is installed if you do not have a server definition to deploy in Design Server.
2. If not running, start tx-rest on the server:  
`<DTX_HOME>/restapi/tomcat/dtxtomcat start tx-rest`

3. Deploy the created package to the server definition in Design Server.
4. Bring up the tx-rest Swagger UI:  
`<DTX_HOME>/restapi/tomcat/dtxtomcat showdocs tx-rest`

5. In the Swagger Explore window, enter /tx-rest/v2/docs and click on the Explore button to view the deployed package.  
You can see a Miscellaneous section having two actions:

- **PUT** /v2/run/nbor\_regs\_validation\_flow
- **POST** /v2/run/nbor\_regs\_validation\_flow

6. In the **PUT** action:
  - a. Expand the **PUT** action and select the Try it out button.
  - b. Leave the input and output sections empty.
  - c. Under flow\_vars section, delete the entire content and replace with the commands to run the flow, for example:

```
{
 "INPUT_FILE": "C:/mydir/data/rgis_pacs_009_cov.xml",
 "OUTPUT_RESULT_JSON": "C:/mydir/data/validation_result.json",
 "REPORT_FORMAT": "json"
}
```

or

```
{
 "INPUT_FILE": "C:/mydir/data/rgis_pacs_009_cov.xml",
 "OUTPUT_RESULT_XML": "C:/mydir/data/validation_result.xml",
 "REPORT_FORMAT": "xml"
}
```

- d. Click Execute blue bar for running the flow.

When flow completes execution, 200 response code is shown in the Server Response section. This run should generate the output same as above.

7. To delete the deployed package, refresh the browser to get the Swagger and explore back to /tx-rest/openapi.json.

There will be a DELETE /v2/packages/{name} action.

8. Expand the DELETE /v2/packages/{name} action and select Try it out.

9. In the Name field, enter the name you gave to the created package.

10. Select the true option from the stop query drop down, then select the blue Execute bar.

You can see a response of 204 with a timestamp, if it is successful.

## Run the example using the flow command server process (flowcmdserver)

To run the example, deploy locally or to ftp server and use the flow command server process (flowcmdserver).

1. Create a server definition where the runtime is installed, if you do not have a server local definition or ftp execution definition to deploy in Design Server.
2. In Design Server, deploy the created package to the server definition. For example, deploy to c:\ftpserver\deployment directory.
3. Execute the flow using the flow command server. The below command is for NBOR ReGIS rgis\_pacs\_009\_cov.xml message.

```
flowcmdserver.exe --dir c:\ftpserver\deployment\flows --flow nbor_regs_validation_flow.json --var
"INPUT_FILE=C:\ftpserver\deployment\tools\mx_service\data\rgis_pacs_009_cov.xml" --audit
"C:\ftpserver\deployment\tools\mx_service\data\rgis_pacs_009_cov_adt.json" -ad
```

The results will be in C:\ftpserver\deployment\flows as validation\_results.xml or validation\_results.json depending on the value of the REPORT\_FORMAT variable.

This command will execute with the variable VALIDATION\_TYPE value defined in the Flow settings in Design Server (schema).

To change, the following command can be used:

```
flowcmdserver.exe --dir c:\ftpserver\deployment\flows --flow nb0r_regis_validation_flow.json --var
"INPUT_FILE=C:\ftpserver\deployment\tools\mx_service\data\rgis_pacs_009_cov.xml" --var "VALIDATION_TYPE=schema" --var
"REPORT_FORMAT=xml" --audit "C:\ftpserver\deployment\tools\mx_service\data\rgis_pacs_009_cov_adt.json" -ad
```

4. The flow will report status similar to:

```
***Starting flow command server

Flow UUID: 85eabe2c-2302-4543-b347-9e18d65c6816
The flow audit file is: "C:\\\\ftpserver\\\\deployment\\\\tools\\\\mx_service\\\\data\\\\rgis_pacs_009_cov_adt.json"
Flow completed successfully
Elapsed time: 1048ms
```

5. Examine the flow output result files validation\_results.xml and the audit file rgis\_pacs\_009\_cov\_adt.json.

---

## NBOR ReGIS schema validation (using maps) example

This example demonstrates the schema validation for NBOR ReGIS messages using maps only.

The maps can perform following level of validation:

- Schema validation
- [What the example contains](#)  
Files and directories included in this example are as follows:
- [Run the example in Design Studio](#)
- [Run the example in Design Server User Interface](#)

---

## What the example contains

Files and directories included in this example are as follows:

- Maps:  
The maps directory contains the following map sources:
  - nb0r\_regis\_mx\_schema\_validation\_xsd.mms  
Contains maps for schema validation of any NBOR ReGIS message.
  - nb0r\_regis\_schema\_validation\_frmwrk\_map\_xsd.mms  
The main map used to apply schema validation to NBOR ReGIS xml messages.
- Schemas:  
The schemas directory contains the following files:
  - XML schemas based on SWIFT MyStandards Readiness NBOR ReGIS Portal:
    - admi.004.001.02.xsd
    - camt.005.001.08.xsd
    - camt.006.001.08.xsd
    - camt.007.001.08.xsd
    - camt.008.001.08.xsd
    - camt.025.001.05.xsd
    - camt.029.001.09.xsd
    - camt.046.001.05.xsd
    - camt.047.001.06.xsd
    - camt.052.001.08.xsd
    - camt.053.001.08.xsd
    - camt.054.001.08.xsd
    - camt.056.001.08.xsd
    - camt.060.001.05.xsd
    - pacs.002.001.10.xsd
    - pacs.004.001.09.xsd
    - pacs.008.001.08.xsd
    - pacs.008.001.08\_TREZROBU.xsd
    - pacs.009.001.08.xsd
    - pacs.009.001.08\_cov.xsd
    - pacs.029.001.01.xsd
    - head.001.001.02\_admi\_004.xsd
    - head.001.001.02\_camt\_005.xsd
    - head.001.001.02\_camt\_006.xsd
    - head.001.001.02\_camt\_007.xsd
    - head.001.001.02\_camt\_008.xsd
    - head.001.001.02\_camt\_025.xsd
    - head.001.001.02\_camt\_029.xsd
    - head.001.001.02\_camt\_046.xsd
    - head.001.001.02\_camt\_047.xsd
    - head.001.001.02\_camt\_052.xsd
    - head.001.001.02\_camt\_053.xsd
    - head.001.001.02\_camt\_054.xsd

- head.001.001.02\_camt\_056.xsd
- head.001.001.02\_camt\_060.xsd
- head.001.001.02\_pacs\_002.xsd
- head.001.001.02\_pacs\_004.xsd
- head.001.001.02\_pacs\_008.xsd
- head.001.001.02\_pacs\_008\_TREZROBU.xsd
- head.001.001.02\_pacs\_009.xsd
- head.001.001.02\_pacs\_009\_cov.xsd
- head.001.001.02\_pacs\_029.xsd
- Trees:  
The trees directory contains the following files:
  - mxvalErrorReport.mtt  
Metadata that represents the xml based structure of the validation report.
  - swiftroute\_funds.mtt  
Metadata that is used as an internal element placeholder.
- Data:  
The data directory contains the following file:
  - Sample NBOR ReGIS valid files for test purpose with header envelope:
    - bah\_admi\_004\_001\_02\_valid.xml
    - bah\_camt\_005\_001\_08\_valid.xml
    - bah\_camt\_006\_001\_08\_valid.xml
    - bah\_camt\_007\_001\_08\_valid.xml
    - bah\_camt\_008\_001\_08\_valid.xml
    - bah\_camt\_025\_001\_05\_valid.xml
    - bah\_camt\_029\_001\_09\_valid.xml
    - bah\_camt\_046\_001\_05\_valid.xml
    - bah\_camt\_047\_001\_06\_valid.xml
    - bah\_camt\_052\_001\_08\_valid.xml
    - bah\_camt\_053\_001\_08\_valid.xml
    - bah\_camt\_054\_001\_08\_valid.xml
    - bah\_camt\_056\_001\_08\_valid.xml
    - bah\_camt\_060\_001\_05\_valid.xml
    - bah\_pacs\_002\_001\_10\_valid.xml
    - bah\_pacs\_004\_001\_09\_valid.xml
    - bah\_pacs\_008\_001\_08\_TREZROBU\_valid.xml
    - bah\_pacs\_008\_001\_08\_valid.xml
    - bah\_pacs\_009\_001\_08\_cov\_valid.xml
    - bah\_pacs\_009\_001\_08\_valid.xml
    - bah\_pacs\_029\_001\_01\_valid.xml

---

## Run the example in Design Studio

1. Build all the maps in the .mms files listed in What the example contains section.
2. Replace the input card 1 in router map, rgis1500\_val (under nbor\_regis\_schema\_validation\_frmwrk\_map\_xsd.mms file) with desired input file.
3. Run the main router map, rgis1500\_val.  
You will see the output file in the data folder called rgis\_error\_report.xml.

---

## Run the example in Design Server User Interface

1. Open a browser pointing to the installation of the IBM Sterling Transformation Extender.
2. Enter user and password credentials.
3. If the project does not exist, import nbor\_rgis.zip.
4. If absent, create a package containing all the Maps, Files, and Flows.
5. Build package in desired server (to build all maps).
6. Open the project.
7. In the Maps tab, open the map rgis1500\_val.
8. Modify the settings for input card 1 on design canvas to point to the desired test file.
9. Save, build, and run the map rgis1500\_val.
10. Right click on the output card on canvas and select View data.

---

## TARGET2 CLM/RTGS/CoCo/Securities Examples:

- **TARGET2 CLM Examples:**  
TARGET2 CLM examples are as follow:
- **TARGET2 RTGS Examples:**  
TARGET2 RTGS examples are as follow:
- **TARGET2 CoCo Examples:**  
TARGET2 CoCo examples are as follow:

- **TARGET2 Securities Examples:**

TARGET2 Securities examples are as follow:

---

## TARGET2 CLM Examples:

TARGET2 CLM examples are as follow:

- **T2 CLM enhanced MX validation (using Flow Server) example**

This example demonstrates the TARGET2 CLM enhanced MX validation for TARGET2 CLM messages using the flow server.

- **T2 CLM enhanced MX validation (using maps) example**

This example demonstrates the enhanced MX validation for TARGET2 CLM messages using maps only.

- **T2 CLM schema validation (using maps) example**

This example demonstrates the schema validation for TARGET2 CLM messages using maps only.

---

## T2 CLM enhanced MX validation (using Flow Server) example

This example demonstrates the TARGET2 CLM enhanced MX validation for TARGET2 CLM messages using the flow server.

The flow can perform following level of validation:

- MX extended validation includes following checks:

- BIC lookup
- Country code lookup
- Currency code lookup
- IBAN format
- Allowed maximum fractional digits per currency
- Usage guideline rules, see mxconfig.xml for list of rules

- Schema validation

- **What the example contains**

Files included in this example are as follows:

- **How to run the example**

This TARGET2 CLM MX validation will use the sample files to demonstrate the generation of validation report as output from a TARGET2 CLM XML message.

- **How to customize the mxconfig.xml file**

Rules enforced by the MX validation framework can be customized by editing the mxconfig.xml located under the validation/mx\_extended/data folder.

---

## What the example contains

Files included in this example are as follows:

- **t2\_clm.zip**

- Test Data Files:
  - t2clm\_pacs\_009.xml
  - t2clm\_pacs\_009\_bad\_rule.xml
  - t2clm\_pacs\_009\_bad\_schema.xml
- Validation Files:
  - mxconfig.xml
  - bic.xml
  - currencycodedecimals.xml
- Schemas:
  - bic.xsd  
Metadata that represents the bic.xml repository file structure.
  - ccy.xsd  
Metadata that represents the currencycodedecimals.xml repository file structure.
  - mxconfig.xsd  
Metadata that represents the mxconfig.xml configuration file structure.
  - mxvalErrorReport.mtt  
Metadata that represents the xml based structure of the validation report.
  - swiftroute\_funds.mtt  
Metadata that is used as an internal element placeholder.
- XML schemas based on SWIFT MyStandards Readiness TARGET2 CLM Portal:
  - admi.004.001.02.xsd
  - admi.005.001.01.xsd
  - admi.007.001.01.xsd
  - camt.003.001.07.xsd
  - camt.004.001.08.xsd
  - camt.005.001.08.xsd
  - camt.006.001.08.xsd
  - camt.018.001.05.xsd

- camt.019.001.07.xsd
  - camt.025.001.05.xsd
  - camt.029.001.09.xsd
  - camt.046.001.05.xsd
  - camt.047.001.06.xsd
  - camt.048.001.05.xsd
  - camt.049.001.05.xsd
  - camt.050.001.05.xsd
  - camt.053.001.08\_btcs.xsd
  - camt.053.001.08\_tsgl.xsd
  - camt.054.001.08.xsd
  - camt.056.001.08.xsd
  - camt.998.001.03\_apmr.xsd
  - camt.998.001.03\_gpmr.xsd
  - camt.998.001.03\_ibmr.xsd
  - camt.998.001.03\_ivrmmr.xsd
  - camt.998.001.03\_mcl.xsd
  - camt.998.001.03\_rpimr.xsd
  - camt.998.001.03\_rpmr.xsd
  - pacs.002.001.10.xsd
  - pacs.009.001.08.xsd
  - pacs.010.001.03.xsd
  - head.001.001.01.xsd
  - XML schema based on W3C XML Signature Syntax and Processing version 1.2:
  - xmldsig-core-schema.xsd
- Note: The schema xmldsig-core-schema.xsd is imported inside of schema head.001.001.01.xsd to handle signatures. It is imported three times as a FILE to the project. Once in folder location t2\_clm/validation/mx\_extended/schemas, second copy in folder location t2\_clm/validation/schema\_only/schemas, and third copy without any logical folder.
- Maps:
    - For Extended Validation:
      - t2cl9100\_val
      - t2cl9051\_head111\_camt
      - t2cl9001\_camt\_054\_001\_08
      - t2cl9002\_camt\_029\_001\_09
      - t2cl9003\_camt\_056\_001\_08
      - t2cl9004\_camt\_053\_001\_08\_btcs
      - t2cl9005\_camt\_053\_001\_08\_tsgl
      - t2cl9006\_camt\_998\_001\_03\_apmr
      - t2cl9007\_camt\_998\_001\_03\_gpmr
      - t2cl9008\_camt\_998\_001\_03\_ibmr
      - t2cl9009\_camt\_998\_001\_03\_ivrmmr
      - t2cl9010\_camt\_998\_001\_03\_mcl
      - t2cl9011\_camt\_998\_001\_03\_rpimr
      - t2cl9012\_camt\_998\_001\_03\_rpmr
      - t2cl9013\_camt\_003\_001\_07
      - t2cl9014\_camt\_004\_001\_08
      - t2cl9015\_camt\_005\_001\_08
      - t2cl9016\_camt\_046\_001\_05
      - t2cl9017\_camt\_047\_001\_06
      - t2cl9018\_camt\_048\_001\_05
      - t2cl9019\_camt\_049\_001\_05
      - t2cl9020\_camt\_050\_001\_05
      - t2cl9021\_camt\_006\_001\_08
      - t2cl9022\_camt\_018\_001\_05
      - t2cl9023\_camt\_019\_001\_07
      - t2cl9024\_camt\_025\_001\_05
      - t2cl9151\_head111\_pacs
      - t2cl9101\_pacs\_002\_001\_10
      - t2cl9102\_pacs\_010\_001\_03
      - t2cl9103\_pacs\_009\_001\_08
      - t2cl9351\_head111\_admi
      - t2cl9301\_admi\_004\_001\_02
      - t2cl9302\_admi\_005\_001\_01
      - t2cl9303\_admi\_007\_001\_01
    - For Schema Validation:
      - t2cl9500\_val
      - t2cl9951\_head\_001\_001\_01
      - t2cl9501\_camt\_054\_001\_08
      - t2cl9502\_camt\_029\_001\_09
      - t2cl9503\_camt\_056\_001\_08
      - t2cl9504\_camt\_053\_001\_08\_btcs
      - t2cl9505\_camt\_053\_001\_08\_tsgl
      - t2cl9506\_camt\_998\_001\_03\_apmr
      - t2cl9507\_camt\_998\_001\_03\_gpmr
      - t2cl9508\_camt\_998\_001\_03\_ibmr
      - t2cl9509\_camt\_998\_001\_03\_ivrmmr
      - t2cl9510\_camt\_998\_001\_03\_mcl
      - t2cl9511\_camt\_998\_001\_03\_rpimr
      - t2cl9512\_camt\_998\_001\_03\_rpmr

- t2cl9513\_camt\_003\_001\_07
  - t2cl9514\_camt\_004\_001\_08
  - t2cl9515\_camt\_005\_001\_08
  - t2cl9516\_camt\_046\_001\_05
  - t2cl9517\_camt\_047\_001\_06
  - t2cl9518\_camt\_048\_001\_05
  - t2cl9519\_camt\_049\_001\_05
  - t2cl9520\_camt\_050\_001\_05
  - t2cl9521\_camt\_006\_001\_08
  - t2cl9522\_camt\_018\_001\_05
  - t2cl9523\_camt\_019\_001\_07
  - t2cl9524\_camt\_025\_001\_05
  - t2cl9601\_pacs\_002\_001\_10
  - t2cl9602\_pacs\_010\_001\_03
  - t2cl9603\_pacs\_009\_001\_08
  - t2cl9801\_admi\_004\_001\_02
  - t2cl9802\_admi\_005\_001\_01
  - t2cl9803\_admi\_007\_001\_01
- Common:
    - mxut1004\_bizsvc\_t2clm
  - Flows:
    - t2\_clm\_validation\_flow

## How to run the example

This TARGET2 CLM MX validation will use the sample files to demonstrate the generation of validation report as output from a TARGET2 CLM XML message.

The stopValidation.json file will report the validation failure due to the pre-conversion checks:

**If input file is other than any of the supported TARGET2 CLM messages as per schema list indicated above.**

1. Import the t2\_clm.zip project into the Design Server.

2. Open the t2\_clm project in Design Server and view the flow t2\_clm\_validation\_flow.

a. The Flow Description points to all the maps to be executed during the flow process, which will be called from within some of the map nodes in the flow. This list has to be added to the description and ensure all lines start in column 1. Following is the list of maps with the description format:

- @packagemap=t2\_clm/validation/mx\_extended/maps/t2\_clm\_mx\_camt\_validation\_enh/t2cl9051\_head111\_camt
- @packagemap=t2\_clm/validation/mx\_extended/maps/t2\_clm\_mx\_camt\_validation\_enh/t2cl9001\_camt\_054\_001\_08
- @packagemap=t2\_clm/validation/mx\_extended/maps/t2\_clm\_mx\_camt\_validation\_enh/t2cl9002\_camt\_029\_001\_09
- @packagemap=t2\_clm/validation/mx\_extended/maps/t2\_clm\_mx\_camt\_validation\_enh/t2cl9003\_camt\_056\_001\_08
- @packagemap=t2\_clm/validation/mx\_extended/maps/t2\_clm\_mx\_camt\_validation\_enh/t2cl9004\_camt\_053\_001\_08\_btcs
- @packagemap=t2\_clm/validation/mx\_extended/maps/t2\_clm\_mx\_camt\_validation\_enh/t2cl9005\_camt\_053\_001\_08\_tsdl
- @packagemap=t2\_clm/validation/mx\_extended/maps/t2\_clm\_mx\_camt\_validation\_enh/t2cl9006\_camt\_998\_001\_03\_apmr
- @packagemap=t2\_clm/validation/mx\_extended/maps/t2\_clm\_mx\_camt\_validation\_enh/t2cl9007\_camt\_998\_001\_03\_gpmr
- @packagemap=t2\_clm/validation/mx\_extended/maps/t2\_clm\_mx\_camt\_validation\_enh/t2cl9008\_camt\_998\_001\_03\_ibmr
- @packagemap=t2\_clm/validation/mx\_extended/maps/t2\_clm\_mx\_camt\_validation\_enh/t2cl9009\_camt\_998\_001\_03\_ivrnr
- @packagemap=t2\_clm/validation/mx\_extended/maps/t2\_clm\_mx\_camt\_validation\_enh/t2cl9010\_camt\_998\_001\_03\_mcl
- @packagemap=t2\_clm/validation/mx\_extended/maps/t2\_clm\_mx\_camt\_validation\_enh/t2cl9011\_camt\_998\_001\_03\_rpimr
- @packagemap=t2\_clm/validation/mx\_extended/maps/t2\_clm\_mx\_camt\_validation\_enh/t2cl9012\_camt\_998\_001\_03\_rpmr
- @packagemap=t2\_clm/validation/mx\_extended/maps/t2\_clm\_mx\_camt\_validation\_enh/t2cl9013\_camt\_003\_001\_07
- @packagemap=t2\_clm/validation/mx\_extended/maps/t2\_clm\_mx\_camt\_validation\_enh/t2cl9014\_camt\_004\_001\_08
- @packagemap=t2\_clm/validation/mx\_extended/maps/t2\_clm\_mx\_camt\_validation\_enh/t2cl9015\_camt\_005\_001\_08
- @packagemap=t2\_clm/validation/mx\_extended/maps/t2\_clm\_mx\_camt\_validation\_enh/t2cl9016\_camt\_046\_001\_05
- @packagemap=t2\_clm/validation/mx\_extended/maps/t2\_clm\_mx\_camt\_validation\_enh/t2cl9017\_camt\_047\_001\_06
- @packagemap=t2\_clm/validation/mx\_extended/maps/t2\_clm\_mx\_camt\_validation\_enh/t2cl9018\_camt\_048\_001\_05
- @packagemap=t2\_clm/validation/mx\_extended/maps/t2\_clm\_mx\_camt\_validation\_enh/t2cl9019\_camt\_049\_001\_05
- @packagemap=t2\_clm/validation/mx\_extended/maps/t2\_clm\_mx\_camt\_validation\_enh/t2cl9020\_camt\_050\_001\_05
- @packagemap=t2\_clm/validation/mx\_extended/maps/t2\_clm\_mx\_camt\_validation\_enh/t2cl9021\_camt\_006\_001\_08
- @packagemap=t2\_clm/validation/mx\_extended/maps/t2\_clm\_mx\_camt\_validation\_enh/t2cl9022\_camt\_018\_001\_05
- @packagemap=t2\_clm/validation/mx\_extended/maps/t2\_clm\_mx\_camt\_validation\_enh/t2cl9023\_camt\_019\_001\_07
- @packagemap=t2\_clm/validation/mx\_extended/maps/t2\_clm\_mx\_camt\_validation\_enh/t2cl9024\_camt\_025\_001\_05
- @packagemap=t2\_clm/validation/mx\_extended/maps/t2\_clm\_mx\_pacs\_validation\_enh/t2cl9151\_head111\_pacs
- @packagemap=t2\_clm/validation/mx\_extended/maps/t2\_clm\_mx\_pacs\_validation\_enh/t2cl9101\_pacs\_002\_001\_10
- @packagemap=t2\_clm/validation/mx\_extended/maps/t2\_clm\_mx\_pacs\_validation\_enh/t2cl9102\_pacs\_010\_001\_03
- @packagemap=t2\_clm/validation/mx\_extended/maps/t2\_clm\_mx\_pacs\_validation\_enh/t2cl9103\_pacs\_009\_001\_08
- @packagemap=t2\_clm/validation/mx\_extended/maps/t2\_clm\_mx\_admi\_validation\_enh/t2cl9351\_head111\_admi
- @packagemap=t2\_clm/validation/mx\_extended/maps/t2\_clm\_mx\_admi\_validation\_enh/t2cl9301\_admi\_004\_001\_02
- @packagemap=t2\_clm/validation/mx\_extended/maps/t2\_clm\_mx\_admi\_validation\_enh/t2cl9302\_admi\_005\_001\_01
- @packagemap=t2\_clm/validation/mx\_extended/maps/t2\_clm\_mx\_admi\_validation\_enh/t2cl9303\_admi\_007\_001\_01
- @packagemap=t2\_clm/validation/schema\_only/maps/t2\_clm\_mx\_schema\_validation\_xsd/t2cl9951\_head\_001\_001\_01
- @packagemap=t2\_clm/validation/schema\_only/maps/t2\_clm\_mx\_schema\_validation\_xsd/t2cl9501\_camt\_054\_001\_08
- @packagemap=t2\_clm/validation/schema\_only/maps/t2\_clm\_mx\_schema\_validation\_xsd/t2cl9502\_camt\_029\_001\_09
- @packagemap=t2\_clm/validation/schema\_only/maps/t2\_clm\_mx\_schema\_validation\_xsd/t2cl9503\_camt\_056\_001\_08
- @packagemap=t2\_clm/validation/schema\_only/maps/t2\_clm\_mx\_schema\_validation\_xsd/t2cl9504\_camt\_053\_001\_08\_btcs
- @packagemap=t2\_clm/validation/schema\_only/maps/t2\_clm\_mx\_schema\_validation\_xsd/t2cl9505\_camt\_053\_001\_08\_tsdl
- @packagemap=t2\_clm/validation/schema\_only/maps/t2\_clm\_mx\_schema\_validation\_xsd/t2cl9506\_camt\_998\_001\_03\_apmr
- @packagemap=t2\_clm/validation/schema\_only/maps/t2\_clm\_mx\_schema\_validation\_xsd/t2cl9507\_camt\_998\_001\_03\_gpmr
- @packagemap=t2\_clm/validation/schema\_only/maps/t2\_clm\_mx\_schema\_validation\_xsd/t2cl9508\_camt\_998\_001\_03\_ibmr
- @packagemap=t2\_clm/validation/schema\_only/maps/t2\_clm\_mx\_schema\_validation\_xsd/t2cl9509\_camt\_998\_001\_03\_ivrnr

- @packagemap=t2\_clm/validation/schema\_only/maps/t2\_clm\_mx\_schema\_validation\_xsd/t2cl9510\_camt\_998\_001\_03\_mcl
- @packagemap=t2\_clm/validation/schema\_only/maps/t2\_clm\_mx\_schema\_validation\_xsd/t2cl9511\_camt\_998\_001\_03\_rpimr
- @packagemap=t2\_clm/validation/schema\_only/maps/t2\_clm\_mx\_schema\_validation\_xsd/t2cl9512\_camt\_998\_001\_03\_rpmr
- @packagemap=t2\_clm/validation/schema\_only/maps/t2\_clm\_mx\_schema\_validation\_xsd/t2cl9513\_camt\_003\_001\_07
- @packagemap=t2\_clm/validation/schema\_only/maps/t2\_clm\_mx\_schema\_validation\_xsd/t2cl9514\_camt\_004\_001\_08
- @packagemap=t2\_clm/validation/schema\_only/maps/t2\_clm\_mx\_schema\_validation\_xsd/t2cl9515\_camt\_005\_001\_08
- @packagemap=t2\_clm/validation/schema\_only/maps/t2\_clm\_mx\_schema\_validation\_xsd/t2cl9516\_camt\_046\_001\_05
- @packagemap=t2\_clm/validation/schema\_only/maps/t2\_clm\_mx\_schema\_validation\_xsd/t2cl9517\_camt\_047\_001\_06
- @packagemap=t2\_clm/validation/schema\_only/maps/t2\_clm\_mx\_schema\_validation\_xsd/t2cl9518\_camt\_048\_001\_05
- @packagemap=t2\_clm/validation/schema\_only/maps/t2\_clm\_mx\_schema\_validation\_xsd/t2cl9519\_camt\_049\_001\_05
- @packagemap=t2\_clm/validation/schema\_only/maps/t2\_clm\_mx\_schema\_validation\_xsd/t2cl9520\_camt\_050\_001\_05
- @packagemap=t2\_clm/validation/schema\_only/maps/t2\_clm\_mx\_schema\_validation\_xsd/t2cl9521\_camt\_006\_001\_08
- @packagemap=t2\_clm/validation/schema\_only/maps/t2\_clm\_mx\_schema\_validation\_xsd/t2cl9522\_camt\_018\_001\_05
- @packagemap=t2\_clm/validation/schema\_only/maps/t2\_clm\_mx\_schema\_validation\_xsd/t2cl9523\_camt\_019\_001\_07
- @packagemap=t2\_clm/validation/schema\_only/maps/t2\_clm\_mx\_schema\_validation\_xsd/t2cl9524\_camt\_025\_001\_05
- @packagemap=t2\_clm/validation/schema\_only/maps/t2\_clm\_mx\_schema\_validation\_xsd/t2cl9601\_pacs\_002\_001\_10
- @packagemap=t2\_clm/validation/schema\_only/maps/t2\_clm\_mx\_schema\_validation\_xsd/t2cl9602\_pacs\_010\_001\_03
- @packagemap=t2\_clm/validation/schema\_only/maps/t2\_clm\_mx\_schema\_validation\_xsd/t2cl9603\_pacs\_009\_001\_08
- @packagemap=t2\_clm/validation/schema\_only/maps/t2\_clm\_mx\_schema\_validation\_xsd/t2cl9801\_admi\_004\_001\_02
- @packagemap=t2\_clm/validation/schema\_only/maps/t2\_clm\_mx\_schema\_validation\_xsd/t2cl9802\_admi\_005\_001\_01
- @packagemap=t2\_clm/validation/schema\_only/maps/t2\_clm\_mx\_schema\_validation\_xsd/t2cl9803\_admi\_007\_001\_01

b. It utilizes the following nodes:

i. Source Nodes:

- mx\_input

This node identifies the input data to be validated in the flow. It uses the variable INPUT\_FILE to set the location of the data.

ii. Map Nodes:

- MX pre-check

Runs map mxut1004\_bizsvc\_t2clm which sets flow variables, checks pre-validation conditions, and creates infoset.json for validation.

- EXT\_VAL

Runs map t2cl9100\_val which calls the appropriate map to perform extended validation of a TARGET2 CLM message.

- XSD\_VAL

Runs map t2cl9500\_val which calls the appropriate map to perform schema validation of a TARGET2 CLM message.

Note: The maps t2cl9100\_val and t2cl9500\_val internally call all the other maps identified in the above Flow Description step.

iii. Decision Nodes:

- pre-valid chk

This node checks the flow variable stopTranslation to decide if further validation/processing is not required.

- EXT\_RESULT\_FORMAT

This node checks the value of the flow variable REPORT\_FORMAT to determine if the resulting report for extended validation should be generated in XML (default) or JSON.

- XSD\_RESULT\_FORMAT

This node checks the value of the flow variable REPORT\_FORMAT to determine if the resulting report for schema-only validation should be generated in XML (default) or JSON.

iv. Route Nodes:

- VAL\_TYPE

This node checks the variable VALIDATION\_TYPE to determine if the process will do extended validation or schema-only validation on the data.

v. Log Nodes:

- FAILURE

In case of no validation due to precondition check failures, this node creates a log file specified by the flow variable FAILURE\_LOG.

vi. Fail Node:

- STOP

It generates error message setup in the node in case of no validation performed.

vii. Passthrough Nodes:

- EXT\_XML\_CONVERT

This node receives an extended validation report in XML format and does not modify the content, thus passing it as is to the appropriate target node.

- XSD\_XML\_CONVERT

This node receives a schema-only validation report in XML format and does not modify the content, thus passing it as is to the appropriate target node.

viii. Format Converter Nodes:

- EXT\_JSON\_CONVERT

This node receives an extended validation report in XML format and converts it to JSON before passing it to the appropriate target node.

- XSD\_JSON\_CONVERT

This node receives a schema-only validation report in XML format and converts it to JSON before passing it to the appropriate target node.

ix. Target Nodes:

- ext\_xml

This node contains the resulting extended validation report in XML format and creates the output as defined by the variable OUTPUT\_RESULT\_XML.

- ext\_json

This node contains the resulting extended validation report in JSON format and creates the output as defined by the variable OUTPUT\_RESULT\_JSON.

- xsd\_xml  
This node contains the resulting schema-only validation report in XML format and creates the output as defined by the variable OUTPUT\_RESULT\_XML.
- xsd\_json  
This node contains the resulting schema-only validation report in JSON format and creates the output as defined by the variable OUTPUT\_RESULT\_JSON.

c. It utilizes the following variables:

i. Flow variables:

- VALIDATION\_TYPE  
Default value is extended. For schema-only validation, it can be changed to schema. It is used in Route node VAL\_TYPE.
- REPORT\_FORMAT  
Default value is xml. It can be changed to json. It is used in Decision nodes EXT\_RESULT\_FORMAT and XSD\_RESULT\_FORMAT.
- INPUT\_FILE  
Default value is ../tools/mx\_service/data/t2clm\_pacs\_009.xml. This is the data file to be used for validation and can be customized. It is used in the Source node mx\_input.
- BIC\_FILE  
Default value is ../data/bic.xml. This is the location of the bic cross-reference file that is used in all the maps executed by the map in the node EXT\_VAL. It can be customized.
- CCY\_FILE  
Default value is ../data/currencycodedecimals.xml. This is the location of the country code cross-reference file that is used in all the maps executed by the map in the node EXT\_VAL. It can be customized.
- MXCONFIG\_FILE  
Default value is ../data/mxconfig.xml. This is the location of the file containing the rule validation settings that is used in all the maps executed by the map in the node EXT\_VAL. It can be customized.
- OUTPUT\_RESULT\_XML  
Default value is validation\_result.xml. This is the location of the validation report file in xml format. It is used in Target nodes ext\_xml and xsd\_xml. It can be customized.
- OUTPUT\_RESULT\_JSON  
Default value is validation\_result.json. This is the location of the validation report file in json format. It is used in Target nodes ext\_json and xsd\_json. It can be customized.
- FAILURE\_LOG  
Default value is stopMXValidation.json. This is the location of the failure log. It is used in Log node FAILURE. It can be customized.
- bizSvc  
For internal use in the Map node MX pre-check to determine the type of data being read in the input file.
- stopValidation  
For internal use in the Map node MX pre-check and is checked in Decision node pre-valid chk to cause a Failure log in case a data file was not recognized as a valid TARGET2 CLM message.

3. Open the main flow t2\_clm\_validation\_flow in the Design Server. It utilizes the above one Source node, three Map nodes, one Route node, three Decision nodes, one Log node, one Fail node, two Passthrough nodes, two Format Converter nodes, and four Target nodes.

4. In Design Server, create a package t2\_clm\_mx\_validation that contains the input files and one flow t2\_clm\_validation\_flow. The maps will automatically be included during deployment of the package onto the runtime server.

Note: Not required for running flows on Design Server user interface directly.

- [\*\*Run the example using three different methods\*\*](#)

You can run the example using three different methods:

---

## Run the example using three different methods

You can run the example using three different methods:

1. Running the example from the Design Server user interface.
2. Deploying the example to tx-rest and running it using the Swagger interface.
3. Deploying locally or to ftp server and using the flow command server process (flowcmdserver).

- [\*\*Run the example from the Design Server user interface\*\*](#)

Use the following steps to run the example from the Design Server user interface:

- [\*\*Run the example using the Swagger interface\*\*](#)

To run the example, deploy to tx-rest and run it using the Swagger interface.

- [\*\*Run the example using the flow command server process \(flowcmdserver\)\*\*](#)

To run the example, deploy locally or to ftp server and use the flow command server process (flowcmdserver).

---

## Run the example from the Design Server user interface

Use the following steps to run the example from the Design Server user interface:

1. In the Design Server, open t2\_clm\_validation\_flow flow and click on Run button.
2. Set the toggle switch Run On Design Server and select flow input file. Example is t2clm\_pacs\_009.xml.  
Note: If a default path is assigned to the variable INPUT\_FILE, the value defined for the variable will be used if an input file is not selected.  
Flow will run and report with the green check box. The report lists execution of all nodes.
3. Right click and select View Link Data on each link to examine the data.
4. View target node by clicking on the node, doing right click on the Input icon and selecting View Data.

---

## Run the example using the Swagger interface

To run the example, deploy to tx-rest and run it using the Swagger interface.

1. Create a Web type server definition where the runtime tx-rest is installed if you do not have a server definition to deploy in Design Server.

2. If not running, start tx-rest on the server:  
`<DTX_HOME>/restapi/tomcat/dtxtomcat start tx-rest`

3. Deploy the created package to the server definition in Design Server.

4. Bring up the tx-rest Swagger UI:  
`<DTX_HOME>/restapi/tomcat/dtxtomcat showdocs tx-rest`

5. In the Swagger Explore window, enter /tx-rest/v2/docs and click on the Explore button to view the deployed package.

You can see a Miscellaneous section having two actions:

- **PUT** /v2/run/t2\_clm\_validation\_flow
- **POST** /v2/run/t2\_clm\_validation\_flow

6. In the **PUT** action:

- a. Expand the **PUT** action and select the Try it out button.
- b. Leave the input and output sections empty.

- c. Under flow\_vars section, delete the entire content and replace with the commands to run the flow, for example:

```
{
 "INPUT_FILE": "C:/mydir/data/t2clm_pacs_009.xml",
 "OUTPUT_RESULT_JSON": "C:/mydir/data/validation_result.json",
 "REPORT_FORMAT": "json",
 "VALIDATION_TYPE": "extended"
}
```

or

```
{
 "INPUT_FILE": "C:/mydir/data/t2clm_pacs_009.xml",
 "OUTPUT_RESULT_XML": "C:/mydir/data/validation_result.xml",
 "REPORT_FORMAT": "xml",
 "VALIDATION_TYPE": "extended"
}
```

- d. Click Execute blue bar for running the flow.

When flow completes execution, 200 response code is shown in the Server Response section. This run should generate the output same as above.

7. To delete the deployed package, refresh the browser to get the Swagger and explore back to /tx-rest/openapi.json.

There will be a DELETE /v2/packages/{name} action.

8. Expand the DELETE /v2/packages/{name} action and select Try it out.

9. In the Name field, enter the name you gave to the created package.

10. Select the true from the stop query drop down, then select the blue Execute bar.

You can see a response of 204 with a timestamp, if it is successful.

---

## Run the example using the flow command server process (flowcmdserver)

To run the example, deploy locally or to ftp server and use the flow command server process (flowcmdserver).

1. Create a server definition where the runtime is installed if you do not have a server local definition or ftp execution definition to deploy in Design Server.
2. In Design Server, deploy the created package to the server definition. For example, deploy to c:\ftpserver\deployment directory.
3. Execute the flow using the flow command server. The below command is for TARGET2 CLM t2clm\_pacs\_009.xml message.

```
flowcmdserver.exe --dir c:\ftpserver\deployment\flows --flow t2_clm_validation_flow.json --var
"INPUT_FILE=C:\ftpserver\deployment\tools\mx_service\data\t2clm_pacs_009.xml" --audit
"C:\ftpserver\deployment\tools\mx_service\data\t2clm_pacs_009_adt.json" -ad
```

The results will be in C:\ftpserver\deployment\flows as validation\_results.xml or validation\_results.json depending on the value of the REPORT\_FORMAT variable.

This command will execute with the variable VALIDATION\_TYPE value defined in the Flow settings in Design Server (extended or schema).

To change, the following command can be used:

```
flowcmdserver.exe --dir c:\ftpserver\deployment\flows --flow t2_clm_validation_flow.json --var
"INPUT_FILE=C:\ftpserver\deployment\tools\mx_service\data\t2clm_pacs_009.xml" --var "VALIDATION_TYPE=schema" --var
"REPORT_FORMAT=json" --audit "C:\ftpserver\deployment\tools\mx_service\data\t2clm_pacs_009_adt.json" -ad
```

4. The flow will report status similar to:

```
***Starting flow command server
```

```
Flow UUID: 85eabe2c-2302-4543-b347-9e18d65c6816
The flow audit file is: "C:\\\\ftpserver\\\\deployment\\\\tools\\\\mx_service\\\\data\\\\t2clm_pacs_009_adt.json"
Flow completed successfully
Elapsed time: 3048ms
```

5. Examine the flow output result files validation\_results.xml and the audit file t2clm\_pacs\_009\_adt.json.

## How to customize the mxconfig.xml file

Rules enforced by the MX validation framework can be customized by editing the mxconfig.xml located under the validation/mx\_extended/data folder.

All the rules enforced by the MX validation map are under element ExtendedValidation.

Any of the sub-elements under ExtendedValidation can be enabled by setting the value to T or disabled by setting to F.

### Example

To disable the BIC lookup validation feature.

### About this task

Change sub-element ExtendedValidation/BICValidation value from T to F.

### Procedure

1. Download the mxconfig.xml file from the Files tab.
2. Open the mxconfig.xml file and edit the sub-element ExtendedValidation/BICValidation value setting from T to F, then save the file.
3. Upload the edited mxconfig.xml file to the project.
4. Run the flow.

Run using an input with BIC lookup validation issue.

Right click on the output card on canvas and select View data. No BIC lookup validation reported.

## T2 CLM enhanced MX validation (using maps) example

This example demonstrates the enhanced MX validation for TARGET2 CLM messages using maps only.

The maps can perform following level of validation:

- MX extended validation includes following checks:
  - BIC lookup
  - Country code lookup
  - Currency code lookup
  - IBAN format
  - Allowed maximum fractional digits per currency
  - Usage guideline rules, see mxconfig.xml for list of rules
- Schema validation
- [What the example contains](#)  
Files and directories included in this example are as follows:
- [Run the example in Design Studio](#)
- [Run the example in Design Server User Interface](#)
- [How to customize the mxconfig.xml file](#)

Rules enforced by the MX validation framework can be customized by editing the mxconfig.xml located under the validation/mx\_extended/data folder.

## What the example contains

Files and directories included in this example are as follows:

- Maps:  
The maps directory contains the following map sources to use when running under Design Studio:
  - t2\_clm\_mx\_admi\_validation\_enh.mms  
Utility maps called by main map for all the admi TARGET2 CLM xml message MX validation.
  - t2\_clm\_mx\_camt\_validation\_enh.mms  
Utility maps called by main map for all the camt TARGET2 CLM xml message MX validation.
  - t2\_clm\_mx\_pacs\_validation\_enh.mms  
Utility maps called by main map for all the pacs TARGET2 CLM xml message MX validation.
  - t2\_clm\_mx\_validation\_frmwrk\_map\_enh.mms  
The main map used to apply MX validation to TARGET2 CLM xml messages.

Note: When running under Design Server, the main framework map is t2\_clm\_mx\_validation\_frmwrk\_map\_flw\_enh.mms.

- Schemas:

The schemas directory contains the following files:

- bic.xsd  
Metadata that represents the bic.xml repository file structure.
- ccy.xsd  
Metadata that represents the currencycodedecimals.xml repository file structure.
- mxconfig.xsd  
Metadata that represents the mxconfig.xml configuration file structure.
- XML schemas based on SWIFT MyStandards Readiness TARGET2 CLM Portal:
  - admi.004.001.02.xsd
  - admi.005.001.01.xsd
  - admi.007.001.01.xsd
  - camt.003.001.07.xsd
  - camt.004.001.08.xsd
  - camt.005.001.08.xsd
  - camt.006.001.08.xsd
  - camt.018.001.05.xsd
  - camt.019.001.07.xsd
  - camt.025.001.05.xsd
  - camt.029.001.09.xsd
  - camt.046.001.05.xsd
  - camt.047.001.06.xsd
  - camt.048.001.05.xsd
  - camt.049.001.05.xsd
  - camt.050.001.05.xsd
  - camt.053.001.08\_btcs.xsd
  - camt.053.001.08\_tsgl.xsd
  - camt.054.001.08.xsd
  - camt.056.001.08.xsd
  - camt.998.001.03\_apmr.xsd
  - camt.998.001.03\_gpmr.xsd
  - camt.998.001.03\_ibmr.xsd
  - camt.998.001.03\_ivrmr.xsd
  - camt.998.001.03\_mcl.xsd
  - camt.998.001.03\_rpimr.xsd
  - camt.998.001.03\_rpmr.xsd
  - pacs.002.001.10.xsd
  - pacs.009.001.08.xsd
  - pacs.010.001.03.xsd
  - head.001.001.01.xsd
- XML schema based on W3C XML Signature Syntax and Processing version 1.2:
  - xmldsig-core-schema.xsd

- Trees:

The trees directory contains the following files:

- mxvalErrorReport.mtt  
Metadata that represents the xml based structure of the validation report.
- swiftroute\_funds.mtt  
Metadata that is used as an internal element placeholder.

- Data:

The data directory contains the following file:

- bic.xml  
Repository file listing all BICs which are used during validation.
- currencycodedecimals.xml  
Repository file list country codes, currency codes and corresponding maximum fractionally digits, used as reference for validation.
- mxconfig.xml  
Holds the MX configuration information on how to process the message.
- Sample TARGET2 CLM valid files without header envelope for test purposes:
  - admi\_007\_001\_01\_valid.xml
- Sample TARGET2 CLM valid files with header envelope for test purposes:
  - bah\_admi\_004\_001\_02\_valid.xml
  - bah\_admi\_005\_001\_01\_valid.xml
  - bah\_admi\_007\_001\_01\_valid.xml
  - bah\_camt\_003\_001\_07\_valid.xml
  - bah\_camt\_004\_001\_08\_valid.xml
  - bah\_camt\_005\_001\_08\_valid.xml
  - bah\_camt\_006\_001\_08\_valid.xml
  - bah\_camt\_018\_001\_05\_valid.xml
  - bah\_camt\_019\_001\_07\_valid.xml
  - bah\_camt\_025\_001\_05\_valid.xml
  - bah\_camt\_029\_001\_09\_valid.xml

- bah\_camt\_046\_001\_05\_valid.xml
- bah\_camt\_047\_001\_06\_valid.xml
- bah\_camt\_048\_001\_05\_valid.xml
- bah\_camt\_049\_001\_05\_valid.xml
- bah\_camt\_050\_001\_05\_valid.xml
- bah\_camt\_053\_001\_08\_btcs\_valid.xml
- bah\_camt\_053\_001\_08\_tsgl\_valid.xml
- bah\_camt\_054\_001\_08\_valid.xml
- bah\_camt\_056\_001\_08\_valid.xml
- bah\_camt\_998\_001\_03\_apmr\_valid.xml
- bah\_camt\_998\_001\_03\_gpmr\_valid.xml
- bah\_camt\_998\_001\_03\_ibmr\_valid.xml
- bah\_camt\_998\_001\_03\_ivrnr\_valid.xml
- bah\_camt\_998\_001\_03\_mcl\_valid.xml
- bah\_camt\_998\_001\_03\_rpimr\_valid.xml
- bah\_camt\_998\_001\_03\_rpmr\_valid.xml
- bah\_pacs\_002\_001\_10\_valid.xml
- bah\_pacs\_009\_001\_08\_valid.xml
- bah\_pacs\_010\_001\_03\_valid.xml

## Run the example in Design Studio

1. Build all the maps in the .mms files listed in What the example contains section.
  2. Replace the input card 1 in router map, t2cl9000\_val (under t2\_clm\_mx\_validation\_frmwrk\_map\_enh.mms or t2\_clm\_mx\_validation\_frmwrk\_map\_flw\_enh.mms file if running under Design Server) with desired input file.
  3. Run the main router map, t2cl9000\_val.
- You will see the output file in the data folder called t2clm\_error\_report.xml.

## Run the example in Design Server User Interface

1. Open a browser pointing to the installation of the IBM Sterling Transformation Extender.
2. Enter user and password credentials.
3. If the project does not exist, import t2\_clm.zip.
4. If absent, create a package containing all the Maps, Files, and Flows.
5. Build package in desired server (to build all maps).
6. Open the project.
7. In the Maps tab, open the map t2cl9000\_val.
8. Modify the settings for input card 1 on design canvas to point to the desired test file.
9. Save, build, and run the map t2cl9000\_val.
10. Right click on the output card on canvas and select View data.

## How to customize the mxconfig.xml file

Rules enforced by the MX validation framework can be customized by editing the mxconfig.xml located under the validation/mx\_extended/data folder.

All the rules enforced by the MX validation map are under element ExtendedValidation.

Any of the sub-elements under ExtendedValidation can be enabled by setting the value to T or disabled by setting to F.

### Example

To disable the BIC lookup validation feature.

### About this task

Change sub-element ExtendedValidation/BICValidation value from T to F.

### Procedure

1. Download the mxconfig.xml file from the Files tab.
2. Open the mxconfig.xml file and edit the sub-element ExtendedValidation/BICValidation value setting from T to F, then save the file.
3. Upload the edited mxconfig.xml file to the project.
4. Run the map.

Run using an input with BIC lookup validation issue.

Right click on the output card on canvas and select View data. No BIC lookup validation reported.

## T2 CLM schema validation (using maps) example

This example demonstrates the schema validation for TARGET2 CLM messages using maps only.

The maps can perform following level of validation:

- Schema validation
- [What the example contains](#)  
Files and directories included in this example are as follows:
- [Run the example in Design Studio](#)
- [Run the example in Design Server User Interface](#)

---

## What the example contains

Files and directories included in this example are as follows:

- Maps:  
The maps directory contains the following map sources to use when running under Design Studio:
  - t2\_clm\_mx\_schema\_validation\_xsd.mms  
Utility maps called by main map for all TARGET2 CLM xml messages schema validation.
  - t2\_clm\_schema\_validation\_frmwrk\_map\_xsd.mms  
The main map used to apply schema only validation to TARGET2 CLM xml messages.
- Schemas:  
The schemas directory contains the following files:
  - XML schemas based on SWIFT MyStandards Readiness TARGET2 CLM Portal:
    - admi.004.001.02.xsd
    - admi.005.001.01.xsd
    - admi.007.001.01.xsd
    - camt.003.001.07.xsd
    - camt.004.001.08.xsd
    - camt.005.001.08.xsd
    - camt.006.001.08.xsd
    - camt.018.001.05.xsd
    - camt.019.001.07.xsd
    - camt.025.001.05.xsd
    - camt.029.001.09.xsd
    - camt.046.001.05.xsd
    - camt.047.001.06.xsd
    - camt.048.001.05.xsd
    - camt.049.001.05.xsd
    - camt.050.001.05.xsd
    - camt.053.001.08\_btcs.xsd
    - camt.053.001.08\_tsgl.xsd
    - camt.054.001.08.xsd
    - camt.056.001.08.xsd
    - camt.998.001.03\_apmr.xsd
    - camt.998.001.03\_gpmr.xsd
    - camt.998.001.03\_ibmr.xsd
    - camt.998.001.03\_ivrnr.xsd
    - camt.998.001.03\_mcl.xsd
    - camt.998.001.03\_rpimr.xsd
    - camt.998.001.03\_rpmr.xsd
    - pacs.002.001.10.xsd
    - pacs.009.001.08.xsd
    - pacs.010.001.03.xsd
    - head.001.001.01.xsd
  - XML schemas based on W3C XML Signature Syntax and Processing version 1.2:
    - xmldsig-core-schema.xsd
- Trees:  
The trees directory contains the following files:
  - mxvalErrorReport.mtt  
Metadata that represents the xml based structure of the validation report.
  - swiftroute\_funds.mtt  
Metadata that is used as an internal element placeholder.
- Data:  
The data directory contains the following file:
  - Sample TARGET2 CLM valid files without header envelope for test purposes:
    - admi\_007\_001\_01\_valid.xml
  - Sample TARGET2 CLM valid files with header envelope for test purposes:
    - bah\_admi\_004\_001\_02\_valid.xml
    - bah\_admi\_005\_001\_01\_valid.xml
    - bah\_admi\_007\_001\_01\_valid.xml
    - bah\_camt\_003\_001\_07\_valid.xml

- bah\_camt\_004\_001\_08\_valid.xml
- bah\_camt\_005\_001\_08\_valid.xml
- bah\_camt\_006\_001\_08\_valid.xml
- bah\_camt\_018\_001\_05\_valid.xml
- bah\_camt\_019\_001\_07\_valid.xml
- bah\_camt\_025\_001\_05\_valid.xml
- bah\_camt\_029\_001\_09\_valid.xml
- bah\_camt\_046\_001\_05\_valid.xml
- bah\_camt\_047\_001\_06\_valid.xml
- bah\_camt\_048\_001\_05\_valid.xml
- bah\_camt\_049\_001\_05\_valid.xml
- bah\_camt\_050\_001\_05\_valid.xml
- bah\_camt\_053\_001\_08\_btcs\_valid.xml
- bah\_camt\_053\_001\_08\_tsgl\_valid.xml
- bah\_camt\_054\_001\_08\_valid.xml
- bah\_camt\_056\_001\_08\_valid.xml
- bah\_camt\_998\_001\_03\_apmr\_valid.xml
- bah\_camt\_998\_001\_03\_gpmr\_valid.xml
- bah\_camt\_998\_001\_03\_ibmr\_valid.xml
- bah\_camt\_998\_001\_03\_ivrnr\_valid.xml
- bah\_camt\_998\_001\_03\_mcl\_valid.xml
- bah\_camt\_998\_001\_03\_rpimr\_valid.xml
- bah\_camt\_998\_001\_03\_rpmr\_valid.xml
- bah\_pacs\_002\_001\_10\_valid.xml
- bah\_pacs\_009\_001\_08\_valid.xml
- bah\_pacs\_010\_001\_03\_valid.xml

## Run the example in Design Studio

1. Build all the maps in the .mms files listed in What the example contains section.
2. Replace the input card 1 in router map, t2cl9500\_val (under t2\_clm\_schema\_validation\_frmwrk\_map\_xsd.mms file) with desired input file.
3. Run the main router map, t2cl9500\_val.  
You will see the output file in the data folder called t2clm\_error\_report.xml.

## Run the example in Design Server User Interface

1. Open a browser pointing to the installation of the IBM Sterling Transformation Extender.
2. Enter user and password credentials.
3. If the project does not exist, import t2\_clm.zip.
4. If absent, create a package containing all the Maps, Files, and Flows.
5. Build package in desired server (to build all maps).
6. Open the project.
7. In the Maps tab, open the map t2cl9500\_val.
8. Modify the settings for input card 1 on design canvas to point to the desired test file.
9. Save, build, and run the map t2cl9500\_val.
10. Right click on the output card on canvas and select View data.

## TARGET2 RTGS Examples:

TARGET2 RTGS examples are as follow:

- [\*\*T2 RTGS enhanced MX validation \(using Flow Server\) example\*\*](#)  
This example demonstrates the TARGET2 RTGS enhanced MX validation for TARGET2 RTGS messages using flow server.
- [\*\*T2 RTGS enhanced MX validation \(using maps\) example\*\*](#)  
This example demonstrates the enhanced MX validation for TARGET2 RTGS messages using maps only.
- [\*\*T2 RTGS schema validation \(using maps\) example\*\*](#)  
This example demonstrates the schema validation for TARGET2 RTGS messages using maps only.

## T2 RTGS enhanced MX validation (using Flow Server) example

This example demonstrates the TARGET2 RTGS enhanced MX validation for TARGET2 RTGS messages using flow server.

The flow can perform following level of validation:

- MX extended validation includes following checks:
  - BIC lookup
  - Country code lookup
  - Currency code lookup
  - IBAN format
  - Allowed maximum fractional digits per currency

- Usage guideline rules, see mxconfig.xml for list of rules
  - Schema validation
  - **[What the example contains](#)**  
Files included in this example are as follows:
  - **[How to run the example](#)**  
This TARGET2 RTGS MX Validation will use the sample files to demonstrate the generation of validation report as output from a TARGET2 RTGS XML message.
  - **[How to customize the mxconfig.xml file](#)**  
Rules enforced by the MX validation framework can be customized by editing the mxconfig.xml located under the validation/mx\_extended/data folder.
- 

## What the example contains

Files included in this example are as follows:

- t2\_rtgs\_flow.zip
  - Test Data Files:
    - t2rtgs\_pacs\_009.xml
    - t2rtgs\_pacs\_009\_bad\_rule.xml
    - t2rtgs\_pacs\_009\_bad\_schema.xml
  - Validation Files:
    - mxconfig.xml
    - bic.xml
    - currencycodedecimals.xml
  - Schemas:
    - bic.xsd  
Metadata that represents the bic.xml repository file structure.
    - ccy.xsd  
Metadata that represents the currencycodedecimals.xml repository file structure.
    - mxconfig.xsd  
Metadata that represents the mxconfig.xml configuration file structure.
    - mxvalErrorReport.mtt  
Metadata that represents the xml based structure of the validation report.
    - swiftroute\_funds.mtt  
Metadata that is used as an internal element placeholder.
  - XML schemas based on SWIFT MyStandards Readiness TARGET2 RTGS Portal:
    - admi.004.001.02.xsd
    - admi.005.001.01.xsd
    - admi.007.001.01.xsd
    - camt.003.001.07.xsd
    - camt.004.001.08.xsd
    - camt.005.001.08.xsd
    - camt.006.001.08.xsd
    - camt.007.001.08.xsd
    - camt.009.001.07.xsd
    - camt.010.001.08.xsd
    - camt.011.001.07.xsd
    - camt.012.001.07.xsd
    - camt.018.001.05.xsd
    - camt.019.001.07.xsd
    - camt.021.001.06.xsd
    - camt.025.001.05.xsd
    - camt.029.001.09.xsd
    - camt.046.001.05.xsd
    - camt.047.001.06.xsd
    - camt.048.001.05.xsd
    - camt.049.001.05.xsd
    - camt.050.001.05.xsd
    - camt.053.001.08.xsd
    - camt.054.001.08.xsd
    - camt.056.001.08.xsd
    - pacs.002.001.10.xsd
    - pacs.004.001.09.xsd
    - pacs.008.001.08.xsd
    - pacs.009.001.08.xsd
    - pacs.010.001.03.xsd
    - pain.998.001.01\_ais.xsd
    - pain.998.001.01\_ati.xsd
    - pain.998.001.01\_atn.xsd
    - head.001.001.01.xsd
  - XML schema based on W3C XML Signature Syntax and Processing version 1.2:
    - xmldsig-core-schema.xsd  
Note: The schema xmldsig-core-schema.xsd is imported inside of schema head.001.001.01.xsd to handle signatures. It is imported three times as a FILE to the project. Once in folder location t2\_rtgs/validation/mx\_extended/schemas, second copy in folder location t2\_rtgs/validation/schema\_only/schemas, and third copy without any logical folder.

- Maps:
  - For Extended Validation:
    - t2rt9100\_val
    - t2rt9051\_head111\_camt
    - t2rt9001\_camt\_054\_001\_08
    - t2rt9002\_camt\_029\_001\_09
    - t2rt9003\_camt\_056\_001\_08
    - t2rt9004\_camt\_053\_001\_08
    - t2rt9005\_camt\_003\_001\_07
    - t2rt9006\_camt\_004\_001\_08
    - t2rt9007\_camt\_005\_001\_08
    - t2rt9008\_camt\_046\_001\_05
    - t2rt9009\_camt\_047\_001\_06
    - t2rt9010\_camt\_048\_001\_05
    - t2rt9011\_camt\_049\_001\_05
    - t2rt9012\_camt\_050\_001\_05
    - t2rt9013\_camt\_006\_001\_08
    - t2rt9014\_camt\_018\_001\_05
    - t2rt9015\_camt\_019\_001\_07
    - t2rt9016\_camt\_025\_001\_05
    - t2rt9017\_camt\_007\_001\_08
    - t2rt9018\_camt\_009\_001\_07
    - t2rt9019\_camt\_010\_001\_08
    - t2rt9020\_camt\_011\_001\_07
    - t2rt9021\_camt\_012\_001\_07
    - t2rt9022\_camt\_021\_001\_06
    - t2rt9151\_head111\_pacs
    - t2rt9101\_pacs\_002\_001\_10
    - t2rt9102\_pacs\_008\_001\_08
    - t2rt9103\_pacs\_010\_001\_03
    - t2rt9104\_pacs\_004\_001\_09
    - t2rt9105\_pacs\_009\_001\_08
    - t2rt9251\_head111\_pain
    - t2rt9201\_pain\_998\_001\_01\_atn
    - t2rt9202\_pain\_998\_001\_01\_ais
    - t2rt9203\_pain\_998\_001\_01\_ati
    - t2rt9351\_head111\_admi
    - t2rt9301\_admi\_004\_001\_02
    - t2rt9302\_admi\_005\_001\_01
    - t2rt9303\_admi\_007\_001\_01
  - For Schema Validation:
    - t2rt9500\_val
    - t2rt9951\_head\_001\_001\_01
    - t2rt9501\_camt\_054\_001\_08
    - t2rt9502\_camt\_029\_001\_09
    - t2rt9503\_camt\_056\_001\_08
    - t2rt9504\_camt\_053\_001\_08
    - t2rt9505\_camt\_003\_001\_07
    - t2rt9506\_camt\_004\_001\_08
    - t2rt9507\_camt\_005\_001\_08
    - t2rt9508\_camt\_046\_001\_05
    - t2rt9509\_camt\_047\_001\_06
    - t2rt9510\_camt\_048\_001\_05
    - t2rt9511\_camt\_049\_001\_05
    - t2rt9512\_camt\_050\_001\_05
    - t2rt9513\_camt\_006\_001\_08
    - t2rt9514\_camt\_018\_001\_05
    - t2rt9515\_camt\_019\_001\_07
    - t2rt9516\_camt\_025\_001\_05
    - t2rt9517\_camt\_007\_001\_08
    - t2rt9518\_camt\_009\_001\_07
    - t2rt9519\_camt\_010\_001\_08
    - t2rt9520\_camt\_011\_001\_07
    - t2rt9521\_camt\_012\_001\_07
    - t2rt9522\_camt\_021\_001\_06
    - t2rt9601\_pacs\_002\_001\_10
    - t2rt9602\_pacs\_008\_001\_08
    - t2rt9603\_pacs\_010\_001\_03
    - t2rt9604\_pacs\_004\_001\_09
    - t2rt9605\_pacs\_009\_001\_08
    - t2rt9701\_pain\_998\_001\_01\_atn
    - t2rt9702\_pain\_998\_001\_01\_ais
    - t2rt9703\_pain\_998\_001\_01\_ati
    - t2rt9801\_admi\_004\_001\_02
    - t2rt9802\_admi\_005\_001\_01
    - t2rt9803\_admi\_007\_001\_01
  - Common:
    - mxut1005\_bizsvc\_t2rtgs
- Flows:
  - t2\_rtgs\_validation\_flow

## How to run the example

This TARGET2 RTGS MX Validation will use the sample files to demonstrate the generation of validation report as output from a TARGET2 RTGS XML message.

The stopValidation.json file will report the validation failure due to the pre-conversion checks:

If input file is other than any of the supported TARGET2 RTGS messages as per schema list indicated above.

1. Import the t2\_rtgs.zip project into the Design Server.
2. Open the t2\_rtgs project in Design Server and view the flow t2\_rtgs\_validation\_flow.

a. The Flow Description points to all the maps to be executed during the flow process, which will be called from within some of the map nodes in the flow. This list has to be added to the description and ensure all lines start in column 1. Following is the list of maps with the description format:

- @packagemap=t2\_rtgs/validation/mx\_extended/maps/t2\_rtgs\_mx\_camt\_validation\_enh/t2rt9051\_head111\_camt
- @packagemap=t2\_rtgs/validation/mx\_extended/maps/t2\_rtgs\_mx\_camt\_validation\_enh/t2rt9001\_camt\_054\_001\_08
- @packagemap=t2\_rtgs/validation/mx\_extended/maps/t2\_rtgs\_mx\_camt\_validation\_enh/t2rt9002\_camt\_029\_001\_09
- @packagemap=t2\_rtgs/validation/mx\_extended/maps/t2\_rtgs\_mx\_camt\_validation\_enh/t2rt9003\_camt\_056\_001\_08
- @packagemap=t2\_rtgs/validation/mx\_extended/maps/t2\_rtgs\_mx\_camt\_validation\_enh/t2rt9004\_camt\_053\_001\_08
- @packagemap=t2\_rtgs/validation/mx\_extended/maps/t2\_rtgs\_mx\_camt\_validation\_enh/t2rt9005\_camt\_003\_001\_07
- @packagemap=t2\_rtgs/validation/mx\_extended/maps/t2\_rtgs\_mx\_camt\_validation\_enh/t2rt9006\_camt\_004\_001\_08
- @packagemap=t2\_rtgs/validation/mx\_extended/maps/t2\_rtgs\_mx\_camt\_validation\_enh/t2rt9007\_camt\_005\_001\_08
- @packagemap=t2\_rtgs/validation/mx\_extended/maps/t2\_rtgs\_mx\_camt\_validation\_enh/t2rt9008\_camt\_046\_001\_05
- @packagemap=t2\_rtgs/validation/mx\_extended/maps/t2\_rtgs\_mx\_camt\_validation\_enh/t2rt9009\_camt\_047\_001\_06
- @packagemap=t2\_rtgs/validation/mx\_extended/maps/t2\_rtgs\_mx\_camt\_validation\_enh/t2rt9010\_camt\_048\_001\_05
- @packagemap=t2\_rtgs/validation/mx\_extended/maps/t2\_rtgs\_mx\_camt\_validation\_enh/t2rt9011\_camt\_049\_001\_05
- @packagemap=t2\_rtgs/validation/mx\_extended/maps/t2\_rtgs\_mx\_camt\_validation\_enh/t2rt9012\_camt\_050\_001\_05
- @packagemap=t2\_rtgs/validation/mx\_extended/maps/t2\_rtgs\_mx\_camt\_validation\_enh/t2rt9013\_camt\_006\_001\_08
- @packagemap=t2\_rtgs/validation/mx\_extended/maps/t2\_rtgs\_mx\_camt\_validation\_enh/t2rt9014\_camt\_018\_001\_05
- @packagemap=t2\_rtgs/validation/mx\_extended/maps/t2\_rtgs\_mx\_camt\_validation\_enh/t2rt9015\_camt\_019\_001\_07
- @packagemap=t2\_rtgs/validation/mx\_extended/maps/t2\_rtgs\_mx\_camt\_validation\_enh/t2rt9016\_camt\_025\_001\_05
- @packagemap=t2\_rtgs/validation/mx\_extended/maps/t2\_rtgs\_mx\_camt\_validation\_enh/t2rt9017\_camt\_007\_001\_08
- @packagemap=t2\_rtgs/validation/mx\_extended/maps/t2\_rtgs\_mx\_camt\_validation\_enh/t2rt9018\_camt\_009\_001\_07
- @packagemap=t2\_rtgs/validation/mx\_extended/maps/t2\_rtgs\_mx\_camt\_validation\_enh/t2rt9019\_camt\_010\_001\_08
- @packagemap=t2\_rtgs/validation/mx\_extended/maps/t2\_rtgs\_mx\_camt\_validation\_enh/t2rt9020\_camt\_011\_001\_07
- @packagemap=t2\_rtgs/validation/mx\_extended/maps/t2\_rtgs\_mx\_camt\_validation\_enh/t2rt9021\_camt\_012\_001\_07
- @packagemap=t2\_rtgs/validation/mx\_extended/maps/t2\_rtgs\_mx\_camt\_validation\_enh/t2rt9022\_camt\_021\_001\_06
- @packagemap=t2\_rtgs/validation/mx\_extended/maps/t2\_rtgs\_mx\_pacs\_validation\_enh/t2rt9151\_head111\_pacs
- @packagemap=t2\_rtgs/validation/mx\_extended/maps/t2\_rtgs\_mx\_pacs\_validation\_enh/t2rt9101\_pacs\_002\_001\_10
- @packagemap=t2\_rtgs/validation/mx\_extended/maps/t2\_rtgs\_mx\_pacs\_validation\_enh/t2rt9102\_pacs\_008\_001\_08
- @packagemap=t2\_rtgs/validation/mx\_extended/maps/t2\_rtgs\_mx\_pacs\_validation\_enh/t2rt9103\_pacs\_010\_001\_03
- @packagemap=t2\_rtgs/validation/mx\_extended/maps/t2\_rtgs\_mx\_pacs\_validation\_enh/t2rt9104\_pacs\_004\_001\_09
- @packagemap=t2\_rtgs/validation/mx\_extended/maps/t2\_rtgs\_mx\_pacs\_validation\_enh/t2rt9105\_pacs\_009\_001\_08
- @packagemap=t2\_rtgs/validation/mx\_extended/maps/t2\_rtgs\_mx\_pain\_validation\_enh/t2rt9251\_head111\_pain
- @packagemap=t2\_rtgs/validation/mx\_extended/maps/t2\_rtgs\_mx\_pain\_validation\_enh/t2rt9201\_pain\_998\_001\_01\_atn
- @packagemap=t2\_rtgs/validation/mx\_extended/maps/t2\_rtgs\_mx\_pain\_validation\_enh/t2rt9202\_pain\_998\_001\_01\_ais
- @packagemap=t2\_rtgs/validation/mx\_extended/maps/t2\_rtgs\_mx\_pain\_validation\_enh/t2rt9203\_pain\_998\_001\_01\_ati
- @packagemap=t2\_rtgs/validation/mx\_extended/maps/t2\_rtgs\_mx\_admi\_validation\_enh/t2rt9351\_head111\_admi
- @packagemap=t2\_rtgs/validation/mx\_extended/maps/t2\_rtgs\_mx\_admi\_validation\_enh/t2rt9301\_admi\_004\_001\_02
- @packagemap=t2\_rtgs/validation/mx\_extended/maps/t2\_rtgs\_mx\_admi\_validation\_enh/t2rt9302\_admi\_005\_001\_01
- @packagemap=t2\_rtgs/validation/mx\_extended/maps/t2\_rtgs\_mx\_admi\_validation\_enh/t2rt9303\_admi\_007\_001\_01
- @packagemap=t2\_rtgs/validation/schema\_only/maps/t2\_rtgs\_mx\_schema\_validation\_xsd/t2rt9951\_head\_001\_001\_01
- @packagemap=t2\_rtgs/validation/schema\_only/maps/t2\_rtgs\_mx\_schema\_validation\_xsd/t2rt9501\_camt\_054\_001\_08
- @packagemap=t2\_rtgs/validation/schema\_only/maps/t2\_rtgs\_mx\_schema\_validation\_xsd/t2rt9502\_camt\_029\_001\_09
- @packagemap=t2\_rtgs/validation/schema\_only/maps/t2\_rtgs\_mx\_schema\_validation\_xsd/t2rt9503\_camt\_056\_001\_08
- @packagemap=t2\_rtgs/validation/schema\_only/maps/t2\_rtgs\_mx\_schema\_validation\_xsd/t2rt9504\_camt\_053\_001\_08
- @packagemap=t2\_rtgs/validation/schema\_only/maps/t2\_rtgs\_mx\_schema\_validation\_xsd/t2rt9505\_camt\_003\_001\_07
- @packagemap=t2\_rtgs/validation/schema\_only/maps/t2\_rtgs\_mx\_schema\_validation\_xsd/t2rt9506\_camt\_004\_001\_08
- @packagemap=t2\_rtgs/validation/schema\_only/maps/t2\_rtgs\_mx\_schema\_validation\_xsd/t2rt9507\_camt\_005\_001\_08
- @packagemap=t2\_rtgs/validation/schema\_only/maps/t2\_rtgs\_mx\_schema\_validation\_xsd/t2rt9508\_camt\_046\_001\_05
- @packagemap=t2\_rtgs/validation/schema\_only/maps/t2\_rtgs\_mx\_schema\_validation\_xsd/t2rt9509\_camt\_047\_001\_06
- @packagemap=t2\_rtgs/validation/schema\_only/maps/t2\_rtgs\_mx\_schema\_validation\_xsd/t2rt9510\_camt\_048\_001\_05
- @packagemap=t2\_rtgs/validation/schema\_only/maps/t2\_rtgs\_mx\_schema\_validation\_xsd/t2rt9511\_camt\_049\_001\_05
- @packagemap=t2\_rtgs/validation/schema\_only/maps/t2\_rtgs\_mx\_schema\_validation\_xsd/t2rt9512\_camt\_050\_001\_05
- @packagemap=t2\_rtgs/validation/schema\_only/maps/t2\_rtgs\_mx\_schema\_validation\_xsd/t2rt9513\_camt\_006\_001\_08
- @packagemap=t2\_rtgs/validation/schema\_only/maps/t2\_rtgs\_mx\_schema\_validation\_xsd/t2rt9514\_camt\_018\_001\_05
- @packagemap=t2\_rtgs/validation/schema\_only/maps/t2\_rtgs\_mx\_schema\_validation\_xsd/t2rt9515\_camt\_019\_001\_07
- @packagemap=t2\_rtgs/validation/schema\_only/maps/t2\_rtgs\_mx\_schema\_validation\_xsd/t2rt9516\_camt\_025\_001\_05
- @packagemap=t2\_rtgs/validation/schema\_only/maps/t2\_rtgs\_mx\_schema\_validation\_xsd/t2rt9517\_camt\_007\_001\_08
- @packagemap=t2\_rtgs/validation/schema\_only/maps/t2\_rtgs\_mx\_schema\_validation\_xsd/t2rt9518\_camt\_009\_001\_07
- @packagemap=t2\_rtgs/validation/schema\_only/maps/t2\_rtgs\_mx\_schema\_validation\_xsd/t2rt9519\_camt\_010\_001\_08
- @packagemap=t2\_rtgs/validation/schema\_only/maps/t2\_rtgs\_mx\_schema\_validation\_xsd/t2rt9520\_camt\_011\_001\_07
- @packagemap=t2\_rtgs/validation/schema\_only/maps/t2\_rtgs\_mx\_schema\_validation\_xsd/t2rt9521\_camt\_012\_001\_07
- @packagemap=t2\_rtgs/validation/schema\_only/maps/t2\_rtgs\_mx\_schema\_validation\_xsd/t2rt9522\_camt\_021\_001\_06
- @packagemap=t2\_rtgs/validation/schema\_only/maps/t2\_rtgs\_mx\_schema\_validation\_xsd/t2rt9601\_pacs\_002\_001\_10
- @packagemap=t2\_rtgs/validation/schema\_only/maps/t2\_rtgs\_mx\_schema\_validation\_xsd/t2rt9602\_pacs\_008\_001\_08
- @packagemap=t2\_rtgs/validation/schema\_only/maps/t2\_rtgs\_mx\_schema\_validation\_xsd/t2rt9603\_pacs\_010\_001\_03
- @packagemap=t2\_rtgs/validation/schema\_only/maps/t2\_rtgs\_mx\_schema\_validation\_xsd/t2rt9604\_pacs\_004\_001\_09
- @packagemap=t2\_rtgs/validation/schema\_only/maps/t2\_rtgs\_mx\_schema\_validation\_xsd/t2rt9605\_pacs\_009\_001\_08
- @packagemap=t2\_rtgs/validation/schema\_only/maps/t2\_rtgs\_mx\_schema\_validation\_xsd/t2rt9701\_pain\_998\_001\_01\_atn

- @packagemap=t2\_rtgs/validation/schema\_only/maps/t2\_rtgs\_mx\_schema\_validation\_xsd/t2rt9702\_pain\_998\_001\_01\_ais
- @packagemap=t2\_rtgs/validation/schema\_only/maps/t2\_rtgs\_mx\_schema\_validation\_xsd/t2rt9703\_pain\_998\_001\_01\_ati
- @packagemap=t2\_rtgs/validation/schema\_only/maps/t2\_rtgs\_mx\_schema\_validation\_xsd/t2rt9801\_admi\_004\_001\_02
- @packagemap=t2\_rtgs/validation/schema\_only/maps/t2\_rtgs\_mx\_schema\_validation\_xsd/t2rt9802\_admi\_005\_001\_01
- @packagemap=t2\_rtgs/validation/schema\_only/maps/t2\_rtgs\_mx\_schema\_validation\_xsd/t2rt9803\_admi\_007\_001\_01

b. It utilizes the following nodes:

i. Source Nodes:

- mx\_input

This node identifies the input data to be validated in the flow. It uses the variable INPUT\_FILE to set the location of the data.

ii. Map Nodes:

- MX pre-check

Runs map mxut1005\_bizsvc\_t2rtgs which sets flow variables, checks pre-validation conditions, and creates infoset.json for validation.

- EXT\_VAL

Runs map t2rt9100\_val which calls the appropriate map to perform extended validation of a TARGET2 RTGS message.

- XSD\_VAL

Runs map t2rt9100\_val which calls the appropriate map to perform schema validation of a TARGET2 RTGS message.

Note: The maps t2rt9100\_val and t2rt9500\_val internally call all the other maps identified in the above Flow Description step.

iii. Decision Nodes:

- pre-valid chk

This node checks the flow variable stopValidation to decide if further validation/processing is not required.

- EXT\_RESULT\_FORMAT

This node checks the value of the flow variable REPORT\_FORMAT to determine if the resulting report for extended validation should be generated in XML (default) or JSON.

- XSD\_RESULT\_FORMAT

This node checks the value of the flow variable REPORT\_FORMAT to determine if the resulting report for schema-only validation should be generated in XML (default) or JSON.

iv. Route Nodes:

- VAL\_TYPE

This node checks the variable VALIDATION\_TYPE to determine if the process will do extended validation or schema-only validation on the data.

v. Log Nodes:

- FAILURE

In case of no validation due to precondition check failures, this node creates a log file specified by the flow variable FAILURE\_LOG.

vi. Fail Node:

- STOP

It generates error message setup in the node in case of no validation performed.

vii. Passthrough Nodes:

- EXT\_XML\_CONVERT

This node receives an extended validation report in XML format and does not modify the content, thus passing it as is to the appropriate target node.

- XSD\_XML\_CONVERT

This node receives a schema-only validation report in XML format and does not modify the content, thus passing it as is to the appropriate target node.

viii. Format Converter Nodes:

- EXT\_JSON\_CONVERT

This node receives an extended validation report in XML format and converts it to JSON before passing it to the appropriate target node.

- XSD\_JSON\_CONVERT

This node receives a schema-only validation report in XML format and converts it to JSON before passing it to the appropriate target node.

ix. Target Nodes:

- ext\_xml

This node contains the resulting extended validation report in XML format and creates the output as defined by the variable OUTPUT\_RESULT\_XML.

- ext\_json

This node contains the resulting extended validation report in JSON format and creates the output as defined by the variable OUTPUT\_RESULT\_JSON.

- xsd\_xml

This node contains the resulting schema-only validation report in XML format and creates the output as defined by the variable OUTPUT\_RESULT\_XML.

- xsd\_json

This node contains the resulting schema-only validation report in JSON format and creates the output as defined by the variable OUTPUT\_RESULT\_JSON.

c. It utilizes the following variables:

i. Flow variables:

- VALIDATION\_TYPE

Default value is extended. For schema-only validation, it can be changed to schema. It is used in Route node VAL\_TYPE.

- REPORT\_FORMAT

Default value is xml. It can be changed to json. It is used in Decision nodes EXT\_RESULT\_FORMAT and XSD\_RESULT\_FORMAT.

- INPUT\_FILE  
Default value is ../tools/mx\_service/data/t2rtgs\_pacs\_009.xml. This is the data file to be used for validation and can be customized. It is used in the Source node mx\_input.
- BIC\_FILE  
Default value is ../data/bic.xml. This is the location of the bic cross-reference file that is used in all the maps executed by the map in node EXT\_VAL. It can be customized.
- CCY\_FILE  
Default value is ../data/currencycodedecimals.xml. This is the location of the country code cross-reference file that is used in all the maps executed by the map in node EXT\_VAL. It can be customized.
- MXCONFIG\_FILE  
Default value is ../data/mxconfig.xml. This is the location of the file containing the rule validation settings that is used in all the maps executed by the map in node EXT\_VAL. It can be customized.
- OUTPUT\_RESULT\_XML  
Default value is validation\_result.xml. This is the location of the validation report file in xml format. It is used in Target nodes ext\_xml and xsd\_xml. It can be customized.
- OUTPUT\_RESULT\_JSON  
Default value is validation\_result.json. This is the location of the validation report file in json format. It is used in Target nodes ext\_json and xsd\_json. It can be customized.
- FAILURE\_LOG  
Default value is stopMXValidation.json. This is the location of the failure log. It is used in Log node FAILURE. It can be customized.
- bizSvc  
For internal use in the Map node MX pre-check to determine the type of data being read in the input file.
- stopValidation  
For internal use in the Map node MX pre-check and is checked in Decision node pre-valid chk to cause a Failure log in case a data file was not recognized as a valid TARGET2 RTGS message.

3. Open the main flow t2\_rtgs\_validation\_flow in the Design Server. It utilizes the above one Source node, three Map nodes, one Route node, three Decision nodes, one Log node, one Fail node, two Passthrough nodes, two Format Converter nodes, and four Target nodes.
4. In Design Server, create a package t2\_rtgs\_mx\_validation that contains the input files and one flow t2\_rtgs\_validation\_flow. The maps will automatically be included during deployment of the package onto the runtime server.  
Note: Not required for running flows on Design Server user interface directly.

- **[Run the example using three different methods](#)**

You can run the example using three different methods:

---

## Run the example using three different methods

You can run the example using three different methods:

1. Running the example from the Design Server user interface.
2. Deploying the example to tx-rest and running it using the Swagger interface.
3. Deploying locally or to ftp server and using the flow command server process (flowcmdserver).

- **[Run the example from the Design Server user interface](#)**

Use the following steps to run the example from the Design Server user interface:

- **[Run the example using the Swagger interface](#)**  
To run the example, deploy to tx-rest and run it using the Swagger interface.
  - **[Run the example using the flow command server process \(flowcmdserver\)](#)**  
To run the example, deploy locally or to ftp server and use the flow command server process (flowcmdserver).
- 
- ## Run the example from the Design Server user interface
- Use the following steps to run the example from the Design Server user interface:
1. In the Design Server, open t2\_rtgs\_validation\_flow flow and click on Run button.
  2. Set the toggle switch Run On Design Server and select flow input file. Example is t2rtgs\_pacs\_009.xml.  
Note: If a default path is assigned to the variable INPUT\_FILE, the value defined for the variable will be used if an input file is not selected.  
Flow will run and report with the green check box. The report lists execution of all nodes.
  3. Right click and select View Link Data on each link to examine the data.
  4. View target node by clicking on the node, doing right click on the Input icon and selecting View Data.

---

## Run the example using the Swagger interface

To run the example, deploy to tx-rest and run it using the Swagger interface.

1. Create a Web type server definition where the runtime tx-rest is installed if you do not have a server definition to deploy in Design Server.
2. If not running, start tx-rest on the server:  

```
<DTX_HOME>/restapi/tomcat/dtxtomcat start tx-rest
```
3. Deploy the created package to the server definition in Design Server.
4. Bring up the tx-rest Swagger UI:  

```
<DTX_HOME>/restapi/tomcat/dtxtomcat showdocs tx-rest
```
5. In the Swagger Explore window, enter /tx-rest/v2/docs and click on the Explore button to view the deployed package.  
 You can see a Miscellaneous section having two actions:

- **PUT** /v2/run/t2\_rtgs\_validation\_flow
- **POST** /v2/run/t2\_rtgs\_validation\_flow

6. In the **PUT** action:
  - a. Expand the **PUT** action and select the Try it out button.
  - b. Leave the input and output sections empty.
  - c. Under flow\_vars section, delete the entire content and replace with the commands to run the flow, for example:

```
{
 "INPUT_FILE": "C:/mydir/data/t2rtgs_pacs_009.xml",
 "OUTPUT_RESULT_JSON": "C:/mydir/data/validation_result.json",
 "REPORT_FORMAT": "json",
 "VALIDATION_TYPE": "extended"
}
```

or

```
{
 "INPUT_FILE": "C:/mydir/data/t2rtgs_pacs_009.xml",
 "OUTPUT_RESULT_XML": "C:/mydir/data/validation_result.xml",
 "REPORT_FORMAT": "xml",
 "VALIDATION_TYPE": "extended"
}
```

- d. Click Execute blue bar for running the flow.

When flow completes execution, 200 response code is shown in the Server Response section. This run should generate the output same as above.

7. To delete the deployed package, refresh the browser to get the Swagger and explore back to /tx-rest/openapi.json.

There will be a DELETE /v2/packages/{name} action.

8. Expand the DELETE /v2/packages/{name} action and select Try it out.

9. In the Name field, enter the name you gave to the created package.

10. Select the true option from the stop query drop down, then select the blue Execute bar.

You can see a response of 204 with a timestamp, if it is successful.

## Run the example using the flow command server process (flowcmdserver)

To run the example, deploy locally or to ftp server and use the flow command server process (flowcmdserver).

1. Create a server definition where the runtime is installed if you do not have a server local definition or ftp execution definition to deploy in Design Server.
2. In Design Server, deploy the created package to the server definition. For example, deploy to c:\ftpserver\deployment directory.
3. Execute the flow using the flow command server. The below command is for TARGET2 RTGS t2rtgs\_pacs\_009.xml message.

```
flowcmdserver.exe --dir c:\ftpserver\deployment\flows --flow t2_rtgs_validation_flow.json --var
"INPUT_FILE=C:\ftpserver\deployment\tools\mx_service\data\t2rtgs_pacs_009.xml" --audit
"C:\ftpserver\deployment\tools\mx_service\data\t2rtgs_pacs_009_adt.json" -ad
```

The results will be in C:\ftpserver\deployment\flows as validation\_results.xml or validation\_results.json depending on the value of the REPORT\_FORMAT variable.

This command will execute with the variable VALIDATION\_TYPE value defined in the Flow settings in Design Server (extended or schema).

To change, the following command can be used:

```
flowcmdserver.exe --dir c:\ftpserver\deployment\flows --flow t2_rtgs_validation_flow.json --var
"INPUT_FILE=C:\ftpserver\deployment\tools\mx_service\data\t2rtgs_pacs_009.xml" --var "VALIDATION_TYPE=schema" --var
"REPORT_FORMAT=json" --audit "C:\ftpserver\deployment\tools\mx_service\data\t2rtgs_pacs_009_adt.json" -ad
```

4. The flow will report status similar to:

```
***Starting flow command server
Flow UUID: 85eabe2c-2302-4543-b347-9e18d65c6816
The flow audit file is: "C:\ftpserver\deployment\tools\mx_service\data\t2rtgs_pacs_009_adt.json"
Flow completed successfully
Elapsed time: 3048ms
```

5. Examine the flow output result files validation\_results.xml and the audit file t2rtgs\_pacs\_009\_adt.json.

## How to customize the mxconfig.xml file

Rules enforced by the MX validation framework can be customized by editing the mxconfig.xml located under the validation/mx\_extended/data folder.

All the rules enforced by the MX validation map are under element ExtendedValidation.

Any of the sub-elements under ExtendedValidation can be enabled by setting the value to T or disabled by setting to F.

## Example

---

To disable the BIC lookup validation feature.

## About this task

---

Change sub-element ExtendedValidation/BICValidation value from T to F.

## Procedure

---

1. Download the mxconfig.xml file from the Files tab.
2. Open the mxconfig.xml file and edit the sub-element ExtendedValidation/BICValidation value setting from T to F, then save the file.
3. Upload the edited mxconfig.xml file to the project.
4. Run the flow.  
Run using an input with BIC lookup validation issue.

Right click on the output card on canvas and select View data. No BIC lookup validation reported.

---

## T2 RTGS enhanced MX validation (using maps) example

This example demonstrates the enhanced MX validation for TARGET2 RTGS messages using maps only.

The maps can perform following level of validation:

- MX extended validation includes following checks:

- BIC lookup
- Country code lookup
- Currency code lookup
- IBAN format
- Allowed maximum fractional digits per currency
- Usage guideline rules, see mxconfig.xml for list of rules

- Schema validation

- **What the example contains**

Files and directories included in this example are as follows:

- **Run the example in Design Studio**

- **Run the example in Design Server User Interface**

- **How to customize the mxconfig.xml file**

Rules enforced by the MX validation framework can be customized by editing the mxconfig.xml located under the validation/mx\_extended/data folder.

---

## What the example contains

Files and directories included in this example are as follows:

- Maps:

The maps directory contains the following map sources to use when running under Design Studio.

- t2\_rtgs\_mx\_admi\_validation\_enh.mms  
Utility maps called by main map for all the admi TARGET2 RTGS xml message MX validation.
- t2\_rtgs\_mx\_camt\_validation\_enh.mms  
Utility maps called by main map for all the camt TARGET2 RTGS xml message MX validation.
- t2\_rtgs\_mx\_pacs\_validation\_enh.mms  
Utility maps called by main map for all the pacs TARGET2 RTGS xml message MX validation.
- t2\_rtgs\_mx\_pain\_validation\_enh.mms  
Utility maps called by main map for all the pain TARGET2 RTGS xml message MX validation.
- t2\_rtgs\_mx\_validation\_frmwrk\_map\_enh.mms  
The main map used to apply MX validation to TARGET2 RTGS xml messages.

Note: When running under Design Server, the main framework map is t2\_rtgs\_mx\_validation\_frmwrk\_map\_flw\_enh.mms.

- Schemas:

The schemas directory contains the following files:

- bic.xsd  
Metadata that represents the bic.xml repository file structure.
- ccy.xsd  
Metadata that represents the currencycodedecimals.xml repository file structure.
- mxconfig.xsd  
Metadata that represents the mxconfig.xml configuration file structure.
- XML schemas based on SWIFT MyStandards Readiness TARGET2 RTGS Portal:

- admi.004.001.02.xsd
- admi.005.001.01.xsd
- admi.007.001.01.xsd
- camt.003.001.07.xsd
- camt.004.001.08.xsd
- camt.005.001.08.xsd
- camt.006.001.08.xsd
- camt.007.001.08.xsd
- camt.009.001.07.xsd
- camt.010.001.08.xsd
- camt.011.001.07.xsd
- camt.012.001.07.xsd
- camt.018.001.05.xsd
- camt.019.001.07.xsd
- camt.021.001.06.xsd
- camt.025.001.05.xsd
- camt.029.001.09.xsd
- camt.046.001.05.xsd
- camt.047.001.06.xsd
- camt.048.001.05.xsd
- camt.049.001.05.xsd
- camt.050.001.05.xsd
- camt.053.001.08.xsd
- camt.054.001.08.xsd
- camt.056.001.08.xsd
- pacs.002.001.10.xsd
- pacs.004.001.09.xsd
- pacs.008.001.08.xsd
- pacs.009.001.08.xsd
- pacs.010.001.03.xsd
- pain.998.001.01\_ais.xsd
- pain.998.001.01\_ati.xsd
- pain.998.001.01\_atn.xsd
- head.001.001.01.xsd
- XML schema based on W3C XML Signature Syntax and Processing version 1.2:
  - xmldsig-core-schema.xsd

- Trees:

The trees directory contains the following files:

- mxvalErrorReport.mtt  
Metadata that represents the xml based structure of the validation report.
- swiftroute\_funds.mtt  
Metadata that is used as an internal element placeholder.

- Data:

The data directory contains the following file:

- bic.xml  
Repository file listing all BICs which are used during validation.
- currencycodedecimals.xml  
Repository file list country codes, currency codes and corresponding maximum fractionally digits, used as reference for validation.
- mxconfig.xml  
Holds the MX configuration information on how to process the message.
- Sample TARGET2 RTGS valid files without header envelope for test purposes:
  - admi\_007\_001\_01\_valid.xml
- Sample TARGET2 RTGS valid files with header envelope for test purposes:
  - bah\_admi\_004\_001\_02\_valid.xml
  - bah\_admi\_005\_001\_01\_valid.xml
  - bah\_admi\_007\_001\_01\_valid.xml
  - bah\_camt\_003\_001\_07\_valid.xml
  - bah\_camt\_004\_001\_08\_valid.xml
  - bah\_camt\_005\_001\_08\_valid.xml
  - bah\_camt\_006\_001\_08\_valid.xml
  - bah\_camt\_007\_001\_08\_valid.xml
  - bah\_camt\_009\_001\_07\_valid.xml
  - bah\_camt\_010\_001\_08\_valid.xml
  - bah\_camt\_011\_001\_07\_valid.xml
  - bah\_camt\_012\_001\_07\_valid.xml
  - bah\_camt\_018\_001\_05\_valid.xml
  - bah\_camt\_019\_001\_07\_valid.xml
  - bah\_camt\_021\_001\_06\_valid.xml
  - bah\_camt\_025\_001\_05\_valid.xml
  - bah\_camt\_029\_001\_09\_valid.xml
  - bah\_camt\_046\_001\_05\_valid.xml
  - bah\_camt\_047\_001\_06\_valid.xml
  - bah\_camt\_048\_001\_05\_valid.xml
  - bah\_camt\_049\_001\_05\_valid.xml

- bah\_camt\_050\_001\_05\_valid.xml
- bah\_camt\_053\_001\_08\_valid.xml
- bah\_camt\_054\_001\_08\_valid.xml
- bah\_camt\_056\_001\_08\_valid.xml
- bah\_pacs\_002\_001\_10\_valid.xml
- bah\_pacs\_004\_001\_09\_valid.xml
- bah\_pacs\_008\_001\_08\_valid.xml
- bah\_pacs\_009\_001\_08\_valid.xml
- bah\_pacs\_010\_001\_03\_valid.xml
- bah\_pain\_998\_001\_01\_ais\_valid.xml
- bah\_pain\_998\_001\_01\_ati\_valid.xml
- bah\_pain\_998\_001\_01\_atn\_valid.xml

## Run the example in Design Studio

1. Build all the maps in the .mms files listed in What the example contains section.
2. Replace the input card 1 in router map, t2rt9000\_val (under t2\_rtgs\_mx\_validation\_frmwrk\_map\_enh.mms file or t2\_rtgs\_mx\_validation\_frmwrk\_map\_flw\_enh.mms file if running under Design Server) with desired input file.
3. Run the main router map, t2rt9000\_val.  
You will see the output file in the data folder called t2rtgs\_error\_report.xml.

## Run the example in Design Server User Interface

1. Open a browser pointing to the installation of the IBM Sterling Transformation Extender.
2. Enter user and password credentials.
3. If the project does not exist, import t2\_rtgs.zip.
4. If absent, create a package containing all the Maps, Files, and Flows.
5. Build package in desired server (to build all maps).
6. Open the project.
7. In the Maps tab, open the map t2rt9000\_val.
8. Modify the settings for input card 1 on design canvas to point to the desired test file.
9. Save, build, and run the map t2rt9000\_val.
10. Right click on the output card on canvas and select View data.

## How to customize the mxconfig.xml file

Rules enforced by the MX validation framework can be customized by editing the mxconfig.xml located under the validation/mx\_extended/data folder.

All the rules enforced by the MX validation map are under element ExtendedValidation.

Any of the sub-elements under ExtendedValidation can be enabled by setting the value to T or disabled by setting to F.

### Example

To disable the BIC lookup validation feature.

### About this task

Change sub-element ExtendedValidation/BICValidation value from T to F.

### Procedure

1. Download the mxconfig.xml file from the Files tab.
2. Open the mxconfig.xml file and edit the sub-element ExtendedValidation/BICValidation value setting from T to F, then save the file.
3. Upload the edited mxconfig.xml file to the project.
4. Run the map.  
Run using an input with BIC lookup validation issue.

Right click on the output card on canvas and select View data. No BIC lookup validation reported.

## T2 RTGS schema validation (using maps) example

This example demonstrates the schema validation for TARGET2 RTGS messages using maps only.

The maps can perform following level of validation:

- Schema validation

- [What the example contains](#)
  - [Run the example in Design Studio](#)
  - [Run the example in Design Server User Interface](#)
- 

## What the example contains

This example contains the following directories and files:

- Maps:

The maps directory contains the following map sources to use when running under Design Studio:

- t2\_rtgs\_mx\_schema\_validation\_xsd.mms  
Utility maps called by main map for all TARGET2 RTGS xml messages schema validation.
- t2\_rtgs\_schema\_validation\_frmwrk\_map\_xsd.mms  
The main map used to apply schema validation to TARGET2 RTGS xml messages.

- Schemas:

The schemas directory contains the following files:

- XML schemas based on SWIFT MyStandards Readiness TARGET2 RTGS Portal:

- admi.004.001.02.xsd
- admi.005.001.01.xsd
- admi.007.001.01.xsd
- camt.003.001.07.xsd
- camt.004.001.08.xsd
- camt.005.001.08.xsd
- camt.006.001.08.xsd
- camt.007.001.08.xsd
- camt.009.001.07.xsd
- camt.010.001.08.xsd
- camt.011.001.07.xsd
- camt.012.001.07.xsd
- camt.018.001.05.xsd
- camt.019.001.07.xsd
- camt.021.001.06.xsd
- camt.025.001.05.xsd
- camt.029.001.09.xsd
- camt.046.001.05.xsd
- camt.047.001.06.xsd
- camt.048.001.05.xsd
- camt.049.001.05.xsd
- camt.050.001.05.xsd
- camt.053.001.08.xsd
- camt.054.001.08.xsd
- camt.056.001.08.xsd
- pacs.002.001.10.xsd
- pacs.004.001.09.xsd
- pacs.008.001.08.xsd
- pacs.009.001.08.xsd
- pacs.010.001.03.xsd
- pain.998.001.01\_ais.xsd
- pain.998.001.01\_ati.xsd
- pain.998.001.01\_atn.xsd
- head.001.001.01.xsd

- XML schemas based on W3C XML Signature Syntax and Processing version 1.2:
  - xmldsig-core-schema.xsd

- Trees:

The trees directory contains the following files:

- mxvalErrorReport.mtt  
Metadata that represents the xml based structure of the validation report.
- swiftroute\_funds.mtt  
Metadata that is used as an internal element placeholder.

- Data:

The data directory contains the following file:

- Sample TARGET2 RTGS valid files without header envelope for test purposes:
  - admi\_007\_001\_01\_valid.xml
- Sample TARGET2 RTGS valid files with header envelope for test purposes:
  - bah\_admi\_004\_001\_02\_valid.xml
  - bah\_admi\_005\_001\_01\_valid.xml
  - bah\_admi\_007\_001\_01\_valid.xml
  - bah\_camt\_003\_001\_07\_valid.xml
  - bah\_camt\_004\_001\_08\_valid.xml
  - bah\_camt\_005\_001\_08\_valid.xml
  - bah\_camt\_006\_001\_08\_valid.xml

- bah\_camt\_007\_001\_08\_valid.xml
- bah\_camt\_009\_001\_07\_valid.xml
- bah\_camt\_010\_001\_08\_valid.xml
- bah\_camt\_011\_001\_07\_valid.xml
- bah\_camt\_012\_001\_07\_valid.xml
- bah\_camt\_018\_001\_05\_valid.xml
- bah\_camt\_019\_001\_07\_valid.xml
- bah\_camt\_021\_001\_06\_valid.xml
- bah\_camt\_025\_001\_05\_valid.xml
- bah\_camt\_029\_001\_09\_valid.xml
- bah\_camt\_046\_001\_05\_valid.xml
- bah\_camt\_047\_001\_06\_valid.xml
- bah\_camt\_048\_001\_05\_valid.xml
- bah\_camt\_049\_001\_05\_valid.xml
- bah\_camt\_050\_001\_05\_valid.xml
- bah\_camt\_053\_001\_08\_valid.xml
- bah\_camt\_054\_001\_08\_valid.xml
- bah\_camt\_056\_001\_08\_valid.xml
- bah\_pacs\_002\_001\_10\_valid.xml
- bah\_pacs\_004\_001\_09\_valid.xml
- bah\_pacs\_008\_001\_08\_valid.xml
- bah\_pacs\_009\_001\_08\_valid.xml
- bah\_pacs\_010\_001\_03\_valid.xml
- bah\_pain\_998\_001\_01\_ais\_valid.xml
- bah\_pain\_998\_001\_01\_ati\_valid.xml
- bah\_pain\_998\_001\_01\_atn\_valid.xml

## Run the example in Design Studio

1. Build all the maps in the .mms files listed in What the example contains section.
2. Replace the input card 1 in router map, t2rt9500\_val (under t2\_rtgs\_schema\_validation\_frmwrk\_map\_xsd.mms file) with desired input file.
3. Run the main router map, t2rt9500\_val.  
You will see the output file in the data folder called t2rtgs\_error\_report.xml.

## Run the example in Design Server User Interface

1. Open a browser pointing to the installation of the IBM Sterling Transformation Extender.
2. Enter user and password credentials.
3. If the project does not exist, import t2\_rtgs.zip.
4. If absent, create a package containing all the Maps, Files, and Flows.
5. Build package in desired server (to build all maps).
6. Open the project.
7. In the Maps tab, open the map t2rt9500\_val.
8. Modify the settings for input card 1 on design canvas to point to the desired test file.
9. Save, build, and run the map t2rt9500\_val.
10. Right click on the output card on canvas and select View data.

## TARGET2 CoCo Examples:

TARGET2 CoCo examples are as follow:

- [\*\*T2 CoCo enhanced MX validation \(using Flow Server\) example\*\*](#)  
This example demonstrates the TARGET2 CoCo enhanced MX validation for TARGET2 CoCo messages using flow server.
- [\*\*T2 CoCo enhanced MX validation \(using maps\) example\*\*](#)  
This example demonstrates the TARGET2 CoCo enhanced MX validation for TARGET2 CoCo messages using maps only.
- [\*\*T2 CoCo schema validation \(using maps\) example\*\*](#)  
This example demonstrates the TARGET2 CoCo schema validation for TARGET2 CoCo messages using maps only.

## T2 CoCo enhanced MX validation (using Flow Server) example

This example demonstrates the TARGET2 CoCo enhanced MX validation for TARGET2 CoCo messages using flow server.

The flow can perform following level of validation:

- MX extended validation includes following checks:
  - BIC lookup
  - Country code lookup
  - Currency code lookup
  - IBAN format
  - Allowed maximum fractional digits per currency

- Usage guideline rules, see mxconfig.xml for list of rules
  - Schema validation
  - **[What the example contains](#)**  
Files included in this example are as follows:
  - **[How to run the example](#)**  
This TARGET2 CoCo MX Validation will use the sample files to demonstrate the generation of validation report as output from a TARGET2 CoCo XML message.
  - **[How to customize the mxconfig.xml file](#)**  
Rules enforced by the MX validation framework can be customized by editing the mxconfig.xml located under the validation/mx\_extended/data folder.
- 

## What the example contains

Files included in this example are as follows:

- t2\_coco.zip
  - Test Data Files:
    - t2coco\_camt\_009.xml
    - t2coco\_camt\_009\_bad\_rule.xml
    - t2coco\_camt\_009\_bad\_schema.xml
  - Validation Files:
    - mxconfig.xml
    - bic.xml
    - currencycodedecimals.xml
  - Schemas:
    - bic.xsd  
Metadata that represents the bic.xml repository file structure.
    - ccy.xsd  
Metadata that represents the currencycodedecimals.xml repository file structure.
    - mxconfig.xsd  
Metadata that represents the mxconfig.xml configuration file structure.
    - mxvalErrorReport.mtt  
Metadata that represents the xml based structure of the validation report.
    - swiftroute\_funds.mtt  
Metadata that is used as an internal element placeholder.
  - XML schemas based on SWIFT MyStandards Readiness TARGET2 CoCo Portal:
    - acmt.007.001.02.xsd
    - acmt.010.001.02.xsd
    - acmt.011.001.02.xsd
    - acmt.015.001.02.xsd
    - acmt.019.001.02.xsd
    - acmt.025.001.02.xsd
    - acmt.026.001.02.xsd
    - admi.007.001.01.xsd
    - camt.009.001.07.xsd
    - camt.010.001.08.xsd
    - camt.011.001.07.xsd
    - camt.012.001.07.xsd
    - camt.018.001.05.xsd
    - camt.019.001.07.xsd
    - camt.024.001.06.xsd
    - camt.025.001.05.xsd
    - camt.048.001.05.xsd
    - camt.069.001.03.xsd
    - camt.070.001.04.xsd
    - camt.071.001.03.xsd
    - camt.076.001.01.xsd
    - camt.077.001.01.xsd
    - camt.099.001.01.xsd
    - camt.100.001.01.xsd
    - reda.014.001.01.xsd
    - reda.015.001.01.xsd
    - reda.016.001.01.xsd
    - reda.017.001.01.xsd
    - reda.022.001.01.xsd
    - reda.031.001.01.xsd
    - reda.039.001.01.xsd
    - reda.040.001.01.xsd
    - reda.042.001.01.xsd
    - reda.043.001.01.xsd
    - reda.064.001.01.xsd
    - reda.065.001.01.xsd
    - head.001.001.01.xsd
  - XML schema based on W3C XML Signature Syntax and Processing version 1.2:

- xmldsig-core-schema.xsd  
Note: The schema xmldsig-core-schema.xsd is imported inside of schema head.001.001.01.xsd to handle signatures. It is imported three times as a FILE to the project. Once in folder location t2\_coco/validation/mx\_extended/schemas, second copy in folder location t2\_coco/validation/schema\_only/schemas, and third copy without any logical folder.
- Maps:
  - For Extended Validation:
    - t2co9100\_val
    - t2co9051\_head111\_camt
    - t2co9001\_camt\_009\_001\_07
    - t2co9002\_camt\_010\_001\_08
    - t2co9003\_camt\_011\_001\_07
    - t2co9004\_camt\_012\_001\_07
    - t2co9005\_camt\_018\_001\_05
    - t2co9006\_camt\_019\_001\_07
    - t2co9007\_camt\_024\_001\_06
    - t2co9008\_camt\_025\_001\_05
    - t2co9009\_camt\_048\_001\_05
    - t2co9010\_camt\_069\_001\_03
    - t2co9011\_camt\_070\_001\_04
    - t2co9012\_camt\_071\_001\_03
    - t2co9013\_camt\_076\_001\_01
    - t2co9014\_camt\_077\_001\_01
    - t2co9015\_camt\_099\_001\_01
    - t2co9016\_camt\_100\_001\_01
    - t2co9151\_head111\_acmt
    - t2co9101\_acmt\_007\_001\_02
    - t2co9102\_acmt\_010\_001\_02
    - t2co9103\_acmt\_011\_001\_02
    - t2co9104\_acmt\_015\_001\_02
    - t2co9105\_acmt\_019\_001\_02
    - t2co9106\_acmt\_025\_001\_02
    - t2co9107\_acmt\_026\_001\_02
    - t2co9251\_head111\_reda
    - t2co9201\_reda\_014\_001\_01
    - t2co9202\_reda\_015\_001\_01
    - t2co9203\_reda\_016\_001\_01
    - t2co9204\_reda\_017\_001\_01
    - t2co9205\_reda\_022\_001\_01
    - t2co9206\_reda\_031\_001\_01
    - t2co9207\_reda\_039\_001\_01
    - t2co9208\_reda\_040\_001\_01
    - t2co9209\_reda\_042\_001\_01
    - t2co9210\_reda\_043\_001\_01
    - t2co9211\_reda\_064\_001\_01
    - t2co9212\_reda\_065\_001\_01
    - t2co9351\_head111\_admi
    - t2co9301\_admi\_007\_001\_01
  - For Schema Validation:
    - t2co9500\_val
    - t2co9951\_head\_001\_001\_01
    - t2co9501\_camt\_009\_001\_07
    - t2co9502\_camt\_010\_001\_08
    - t2co9503\_camt\_011\_001\_07
    - t2co9504\_camt\_012\_001\_07
    - t2co9505\_camt\_018\_001\_05
    - t2co9506\_camt\_019\_001\_07
    - t2co9507\_camt\_024\_001\_06
    - t2co9508\_camt\_025\_001\_05
    - t2co9509\_camt\_048\_001\_05
    - t2co9510\_camt\_069\_001\_03
    - t2co9511\_camt\_070\_001\_04
    - t2co9512\_camt\_071\_001\_03
    - t2co9513\_camt\_076\_001\_01
    - t2co9514\_camt\_077\_001\_01
    - t2co9515\_camt\_099\_001\_01
    - t2co9516\_camt\_100\_001\_01
    - t2co9601\_acmt\_007\_001\_02
    - t2co9602\_acmt\_010\_001\_02
    - t2co9603\_acmt\_011\_001\_02
    - t2co9604\_acmt\_015\_001\_02
    - t2co9605\_acmt\_019\_001\_02
    - t2co9606\_acmt\_025\_001\_02
    - t2co9607\_acmt\_026\_001\_02
    - t2co9701\_reda\_014\_001\_01
    - t2co9702\_reda\_015\_001\_01
    - t2co9703\_reda\_016\_001\_01
    - t2co9704\_reda\_017\_001\_01
    - t2co9705\_reda\_022\_001\_01
    - t2co9706\_reda\_031\_001\_01
    - t2co9707\_reda\_039\_001\_01

- t2co9708\_reda\_040\_001\_01
- t2co9709\_reda\_042\_001\_01
- t2co9710\_reda\_043\_001\_01
- t2co9711\_reda\_064\_001\_01
- t2co9712\_reda\_065\_001\_01
- t2co9801\_admi\_007\_001\_01
- Common:
  - mxut1006\_bizsvc\_t2coco
- Flows:
  - t2\_coco\_validation\_flow

## How to run the example

This TARGET2 CoCo MX Validation will use the sample files to demonstrate the generation of validation report as output from a TARGET2 CoCo XML message.

The stopValidation.json file will report the validation failure due to the pre-conversion checks:

**If input file is other than any of the supported TARGET2 CoCo messages as per schema list indicated above.**

1. Import the t2\_coco.zip project into the Design Server.
2. Open the t2\_coco project in Design Server and view the flow t2\_coco\_validation\_flow.
  - a. The Flow Description points to all the maps to be executed during the flow process, which will be called from within some of the map nodes in the flow. This list has to be added to the description and ensure all lines start in column 1. Following is the list of maps with the description format:
    - @packagemap=t2\_coco/validation/mx\_extended/maps/t2\_coco\_mx\_camt\_validation\_enh/t2co9051\_head111\_camt
    - @packagemap=t2\_coco/validation/mx\_extended/maps/t2\_coco\_mx\_camt\_validation\_enh/t2co9001\_camt\_009\_001\_07
    - @packagemap=t2\_coco/validation/mx\_extended/maps/t2\_coco\_mx\_camt\_validation\_enh/t2co9002\_camt\_010\_001\_08
    - @packagemap=t2\_coco/validation/mx\_extended/maps/t2\_coco\_mx\_camt\_validation\_enh/t2co9003\_camt\_011\_001\_07
    - @packagemap=t2\_coco/validation/mx\_extended/maps/t2\_coco\_mx\_camt\_validation\_enh/t2co9004\_camt\_012\_001\_07
    - @packagemap=t2\_coco/validation/mx\_extended/maps/t2\_coco\_mx\_camt\_validation\_enh/t2co9005\_camt\_018\_001\_05
    - @packagemap=t2\_coco/validation/mx\_extended/maps/t2\_coco\_mx\_camt\_validation\_enh/t2co9006\_camt\_019\_001\_07
    - @packagemap=t2\_coco/validation/mx\_extended/maps/t2\_coco\_mx\_camt\_validation\_enh/t2co9007\_camt\_024\_001\_06
    - @packagemap=t2\_coco/validation/mx\_extended/maps/t2\_coco\_mx\_camt\_validation\_enh/t2co9008\_camt\_025\_001\_05
    - @packagemap=t2\_coco/validation/mx\_extended/maps/t2\_coco\_mx\_camt\_validation\_enh/t2co9009\_camt\_048\_001\_05
    - @packagemap=t2\_coco/validation/mx\_extended/maps/t2\_coco\_mx\_camt\_validation\_enh/t2co9010\_camt\_069\_001\_03
    - @packagemap=t2\_coco/validation/mx\_extended/maps/t2\_coco\_mx\_camt\_validation\_enh/t2co9011\_camt\_070\_001\_04
    - @packagemap=t2\_coco/validation/mx\_extended/maps/t2\_coco\_mx\_camt\_validation\_enh/t2co9012\_camt\_071\_001\_03
    - @packagemap=t2\_coco/validation/mx\_extended/maps/t2\_coco\_mx\_camt\_validation\_enh/t2co9013\_camt\_076\_001\_01
    - @packagemap=t2\_coco/validation/mx\_extended/maps/t2\_coco\_mx\_camt\_validation\_enh/t2co9014\_camt\_077\_001\_01
    - @packagemap=t2\_coco/validation/mx\_extended/maps/t2\_coco\_mx\_camt\_validation\_enh/t2co9015\_camt\_099\_001\_01
    - @packagemap=t2\_coco/validation/mx\_extended/maps/t2\_coco\_mx\_camt\_validation\_enh/t2co9016\_camt\_100\_001\_01
    - @packagemap=t2\_coco/validation/mx\_extended/maps/t2\_coco\_mx\_acmt\_validation\_enh/t2co9151\_head111\_acmt
    - @packagemap=t2\_coco/validation/mx\_extended/maps/t2\_coco\_mx\_acmt\_validation\_enh/t2co9101\_acmt\_007\_001\_02
    - @packagemap=t2\_coco/validation/mx\_extended/maps/t2\_coco\_mx\_acmt\_validation\_enh/t2co9102\_acmt\_010\_001\_02
    - @packagemap=t2\_coco/validation/mx\_extended/maps/t2\_coco\_mx\_acmt\_validation\_enh/t2co9103\_acmt\_011\_001\_02
    - @packagemap=t2\_coco/validation/mx\_extended/maps/t2\_coco\_mx\_acmt\_validation\_enh/t2co9104\_acmt\_015\_001\_02
    - @packagemap=t2\_coco/validation/mx\_extended/maps/t2\_coco\_mx\_acmt\_validation\_enh/t2co9105\_acmt\_019\_001\_02
    - @packagemap=t2\_coco/validation/mx\_extended/maps/t2\_coco\_mx\_acmt\_validation\_enh/t2co9106\_acmt\_025\_001\_02
    - @packagemap=t2\_coco/validation/mx\_extended/maps/t2\_coco\_mx\_acmt\_validation\_enh/t2co9107\_acmt\_026\_001\_02
    - @packagemap=t2\_coco/validation/mx\_extended/maps/t2\_coco\_mx\_reda\_validation\_enh/t2co9251\_head111\_reda
    - @packagemap=t2\_coco/validation/mx\_extended/maps/t2\_coco\_mx\_reda\_validation\_enh/t2co9201\_reda\_014\_001\_01
    - @packagemap=t2\_coco/validation/mx\_extended/maps/t2\_coco\_mx\_reda\_validation\_enh/t2co9202\_reda\_015\_001\_01
    - @packagemap=t2\_coco/validation/mx\_extended/maps/t2\_coco\_mx\_reda\_validation\_enh/t2co9203\_reda\_016\_001\_01
    - @packagemap=t2\_coco/validation/mx\_extended/maps/t2\_coco\_mx\_reda\_validation\_enh/t2co9204\_reda\_017\_001\_01
    - @packagemap=t2\_coco/validation/mx\_extended/maps/t2\_coco\_mx\_reda\_validation\_enh/t2co9205\_reda\_022\_001\_01
    - @packagemap=t2\_coco/validation/mx\_extended/maps/t2\_coco\_mx\_reda\_validation\_enh/t2co9206\_reda\_031\_001\_01
    - @packagemap=t2\_coco/validation/mx\_extended/maps/t2\_coco\_mx\_reda\_validation\_enh/t2co9207\_reda\_039\_001\_01
    - @packagemap=t2\_coco/validation/mx\_extended/maps/t2\_coco\_mx\_reda\_validation\_enh/t2co9208\_reda\_040\_001\_01
    - @packagemap=t2\_coco/validation/mx\_extended/maps/t2\_coco\_mx\_reda\_validation\_enh/t2co9209\_reda\_042\_001\_01
    - @packagemap=t2\_coco/validation/mx\_extended/maps/t2\_coco\_mx\_reda\_validation\_enh/t2co9210\_reda\_043\_001\_01
    - @packagemap=t2\_coco/validation/mx\_extended/maps/t2\_coco\_mx\_reda\_validation\_enh/t2co9211\_reda\_064\_001\_01
    - @packagemap=t2\_coco/validation/mx\_extended/maps/t2\_coco\_mx\_reda\_validation\_enh/t2co9212\_reda\_065\_001\_01
    - @packagemap=t2\_coco/validation/mx\_extended/maps/t2\_coco\_mx\_admi\_validation\_enh/t2co9351\_head111\_admi
    - @packagemap=t2\_coco/validation/mx\_extended/maps/t2\_coco\_mx\_admi\_validation\_enh/t2co9301\_admi\_007\_001\_01
    - @packagemap=t2\_coco/validation/schema\_only/maps/t2\_coco\_mx\_schema\_validation\_xsd/t2co9951\_head\_001\_001\_01
    - @packagemap=t2\_coco/validation/schema\_only/maps/t2\_coco\_mx\_schema\_validation\_xsd/t2co9501\_camt\_009\_001\_07
    - @packagemap=t2\_coco/validation/schema\_only/maps/t2\_coco\_mx\_schema\_validation\_xsd/t2co9502\_camt\_010\_001\_08
    - @packagemap=t2\_coco/validation/schema\_only/maps/t2\_coco\_mx\_schema\_validation\_xsd/t2co9503\_camt\_011\_001\_07
    - @packagemap=t2\_coco/validation/schema\_only/maps/t2\_coco\_mx\_schema\_validation\_xsd/t2co9504\_camt\_012\_001\_07
    - @packagemap=t2\_coco/validation/schema\_only/maps/t2\_coco\_mx\_schema\_validation\_xsd/t2co9505\_camt\_018\_001\_05
    - @packagemap=t2\_coco/validation/schema\_only/maps/t2\_coco\_mx\_schema\_validation\_xsd/t2co9506\_camt\_019\_001\_07
    - @packagemap=t2\_coco/validation/schema\_only/maps/t2\_coco\_mx\_schema\_validation\_xsd/t2co9507\_camt\_024\_001\_06
    - @packagemap=t2\_coco/validation/schema\_only/maps/t2\_coco\_mx\_schema\_validation\_xsd/t2co9508\_camt\_025\_001\_05
    - @packagemap=t2\_coco/validation/schema\_only/maps/t2\_coco\_mx\_schema\_validation\_xsd/t2co9509\_camt\_048\_001\_05
    - @packagemap=t2\_coco/validation/schema\_only/maps/t2\_coco\_mx\_schema\_validation\_xsd/t2co9510\_camt\_069\_001\_03
    - @packagemap=t2\_coco/validation/schema\_only/maps/t2\_coco\_mx\_schema\_validation\_xsd/t2co9511\_camt\_070\_001\_04
    - @packagemap=t2\_coco/validation/schema\_only/maps/t2\_coco\_mx\_schema\_validation\_xsd/t2co9512\_camt\_071\_001\_03
    - @packagemap=t2\_coco/validation/schema\_only/maps/t2\_coco\_mx\_schema\_validation\_xsd/t2co9513\_camt\_076\_001\_01
    - @packagemap=t2\_coco/validation/schema\_only/maps/t2\_coco\_mx\_schema\_validation\_xsd/t2co9514\_camt\_077\_001\_01

- @packagemap=t2\_coco/validation/schema\_only/maps/t2\_coco\_mx\_schema\_validation\_xsd/t2co9515\_camt\_099\_001\_01
- @packagemap=t2\_coco/validation/schema\_only/maps/t2\_coco\_mx\_schema\_validation\_xsd/t2co9516\_camt\_100\_001\_01
- @packagemap=t2\_coco/validation/schema\_only/maps/t2\_coco\_mx\_schema\_validation\_xsd/t2co9601\_acmt\_007\_001\_02
- @packagemap=t2\_coco/validation/schema\_only/maps/t2\_coco\_mx\_schema\_validation\_xsd/t2co9602\_acmt\_010\_001\_02
- @packagemap=t2\_coco/validation/schema\_only/maps/t2\_coco\_mx\_schema\_validation\_xsd/t2co9603\_acmt\_011\_001\_02
- @packagemap=t2\_coco/validation/schema\_only/maps/t2\_coco\_mx\_schema\_validation\_xsd/t2co9604\_acmt\_015\_001\_02
- @packagemap=t2\_coco/validation/schema\_only/maps/t2\_coco\_mx\_schema\_validation\_xsd/t2co9605\_acmt\_019\_001\_02
- @packagemap=t2\_coco/validation/schema\_only/maps/t2\_coco\_mx\_schema\_validation\_xsd/t2co9606\_acmt\_025\_001\_02
- @packagemap=t2\_coco/validation/schema\_only/maps/t2\_coco\_mx\_schema\_validation\_xsd/t2co9607\_acmt\_026\_001\_02
- @packagemap=t2\_coco/validation/schema\_only/maps/t2\_coco\_mx\_schema\_validation\_xsd/t2co9701\_reda\_014\_001\_01
- @packagemap=t2\_coco/validation/schema\_only/maps/t2\_coco\_mx\_schema\_validation\_xsd/t2co9702\_reda\_015\_001\_01
- @packagemap=t2\_coco/validation/schema\_only/maps/t2\_coco\_mx\_schema\_validation\_xsd/t2co9703\_reda\_016\_001\_01
- @packagemap=t2\_coco/validation/schema\_only/maps/t2\_coco\_mx\_schema\_validation\_xsd/t2co9704\_reda\_017\_001\_01
- @packagemap=t2\_coco/validation/schema\_only/maps/t2\_coco\_mx\_schema\_validation\_xsd/t2co9705\_reda\_022\_001\_01
- @packagemap=t2\_coco/validation/schema\_only/maps/t2\_coco\_mx\_schema\_validation\_xsd/t2co9706\_reda\_031\_001\_01
- @packagemap=t2\_coco/validation/schema\_only/maps/t2\_coco\_mx\_schema\_validation\_xsd/t2co9707\_reda\_039\_001\_01
- @packagemap=t2\_coco/validation/schema\_only/maps/t2\_coco\_mx\_schema\_validation\_xsd/t2co9708\_reda\_040\_001\_01
- @packagemap=t2\_coco/validation/schema\_only/maps/t2\_coco\_mx\_schema\_validation\_xsd/t2co9709\_reda\_042\_001\_01
- @packagemap=t2\_coco/validation/schema\_only/maps/t2\_coco\_mx\_schema\_validation\_xsd/t2co9710\_reda\_043\_001\_01
- @packagemap=t2\_coco/validation/schema\_only/maps/t2\_coco\_mx\_schema\_validation\_xsd/t2co9711\_reda\_064\_001\_01
- @packagemap=t2\_coco/validation/schema\_only/maps/t2\_coco\_mx\_schema\_validation\_xsd/t2co9712\_reda\_065\_001\_01
- @packagemap=t2\_coco/validation/schema\_only/maps/t2\_coco\_mx\_schema\_validation\_xsd/t2co9801\_admi\_007\_001\_01

b. It utilizes the following nodes:

i. Source Nodes:

- mx\_input

This node identifies the input data to be validated in the flow. It uses the variable INPUT\_FILE to set the location of the data.

ii. Map Nodes:

- MX pre-check

Runs map mxut1006\_bizsvc\_t2coco which sets flow variables, checks pre-validation conditions, and creates infoset.json for validation.

- EXT\_VAL

Runs map t2co9100\_val which calls the appropriate map to perform extended validation of a TARGET2 CoCo message.

- XSD\_VAL

Runs map t2co9500\_val which calls the appropriate map to perform schema validation of a TARGET2 CoCo message.

Note: The maps t2co9100\_val and t2co9500\_val internally call all the other maps identified in the above Flow Description step.

iii. Decision Nodes:

- pre-valid chk

This node checks the flow variable stopTranslation to decide if further validation/processing is not required.

- EXT\_RESULT\_FORMAT

This node checks the value of the flow variable REPORT\_FORMAT to determine if the resulting report for extended validation should be generated in XML (default) or JSON.

- XSD\_RESULT\_FORMAT

This node checks the value of the flow variable REPORT\_FORMAT to determine if the resulting report for schema-only validation should be generated in XML (default) or JSON.

iv. Route Nodes:

- VAL\_TYPE

This node checks the variable VALIDATION\_TYPE to determine if the process will do extended validation or schema-only validation on the data.

v. Log Nodes:

- FAILURE

In case of no validation due to precondition check failures, this node creates a log file specified by the flow variable FAILURE\_LOG.

vi. Fail Node:

- STOP

It generates error message setup in the node in case of no validation performed.

vii. Passthrough Nodes:

- EXT\_XML\_CONVERT

This node receives an extended validation report in XML format and does not modify the content, thus passing it as is to the appropriate target node.

- XSD\_XML\_CONVERT

This node receives a schema-only validation report in XML format and does not modify the content, thus passing it as is to the appropriate target node.

viii. Format Converter Nodes:

- EXT\_JSON\_CONVERT

This node receives an extended validation report in XML format and converts it to JSON before passing it to the appropriate target node.

- XSD\_JSON\_CONVERT

This node receives a schema-only validation report in XML format and converts it to JSON before passing it to the appropriate target node.

ix. Target Nodes:

- ext\_xml

This node contains the resulting extended validation report in XML format and creates the output as defined by the variable OUTPUT\_RESULT\_XML.

- ext\_json  
This node contains the resulting extended validation report in JSON format and creates the output as defined by the variable OUTPUT\_RESULT\_JSON.
  - xsd\_xml  
This node contains the resulting schema-only validation report in XML format and creates the output as defined by the variable OUTPUT\_RESULT\_XML.
  - xsd\_json  
This node contains the resulting schema-only validation report in JSON format and creates the output as defined by the variable OUTPUT\_RESULT\_JSON.
- c. It utilizes the following variables:
- i. Flow variables:
    - VALIDATION\_TYPE  
Default value is extended. For schema-only validation, it can be changed to schema. It is used in Route node VAL\_TYPE.
    - REPORT\_FORMAT  
Default value is xml. It can be changed to json. It is used in Decision nodes EXT\_RESULT\_FORMAT and XSD\_RESULT\_FORMAT.
    - INPUT\_FILE  
Default value is ../tools/mx\_service/data/t2coco\_camt\_009.xml. This is the data file to be used for validation and can be customized. It is used in the Source node mx\_input.
    - BIC\_FILE  
Default value is ../data/bic.xml. This is the location of the bic cross-reference file that is used in all the maps executed by the map in the node EXT\_VAL. It can be customized.
    - CCY\_FILE  
Default value is ../data/currencycodedecimals.xml. This is the location of the country code cross-reference file that is used in all the maps executed by the map in the node EXT\_VAL. It can be customized.
    - MXCONFIG\_FILE  
Default value is ../data/mxconfig.xml. This is the location of the file containing the rule validation settings that is used in all the maps executed by the map in the node EXT\_VAL. It can be customized.
    - OUTPUT\_RESULT\_XML  
Default value is validation\_result.xml. This is the location of the validation report file in xml format. It is used in Target nodes ext\_xml and xsd\_xml. It can be customized.
    - OUTPUT\_RESULT\_JSON  
Default value is validation\_result.json. This is the location of the validation report file in json format. It is used in Target nodes ext\_json and xsd\_json. It can be customized.
    - FAILURE\_LOG  
Default value is stopMXValidation.json. This is the location of the failure log. It is used in Log node FAILURE. It can be customized.
    - bizSvc  
For internal use in the Map node MX pre-check to determine the type of data being read in the input file.
    - stopValidation  
For internal use in the Map node MX pre-check and is checked in Decision node pre-valid chk to cause a Failure log in case a data file was not recognized as a valid TARGET2 Coco message.

3. Open the main flow t2\_coco\_validation\_flow in the Design Server. It utilizes the above one Source node, three Map nodes, one Route node, three Decision nodes, one Log node, one Fail node, two Passthrough nodes, two Format Converter nodes, and four Target nodes.

4. In Design Server, create a package t2\_coco\_mx\_validation that contains the input files and one flow t2\_coco\_validation\_flow. The maps will automatically be included during deployment of the package onto the runtime server.

Note: Not required for running flows on Design Server user interface directly.

- [Run the example using three different methods](#)

You can run the example using three different methods:

## Run the example using three different methods

You can run the example using three different methods:

1. Running the example from the Design Server user interface.
2. Deploying the example to tx-rest and running it using the Swagger interface.
3. Deploying locally or to ftp server and using the flow command server process (flowcmdserver).

- [Run the example from the Design Server user interface](#)

Use the following steps to run the example from the Design Server user interface:

- [Run the example using the Swagger interface](#)

To run the example, deploy to tx-rest and run it using the Swagger interface.

- [Run the example using the flow command server process \(flowcmdserver\)](#)

To run the example, deploy locally or to ftp server and use the flow command server process (flowcmdserver).

## Run the example from the Design Server user interface

Use the following steps to run the example from the Design Server user interface:

1. In the Design Server, open t2\_coco\_validation\_flow flow and click on Run button.
2. Set the toggle switch Run On Design Server and select flow input file. Example is t2coco\_camt\_009.xml.  
Note: If a default path is assigned to the variable INPUT\_FILE, the value defined for the variable will be used if an input file is not selected.  
Flow will run and report with the green check box. The report lists execution of all nodes.
3. Right click and select View Link Data on each link to examine the data.
4. View target node by clicking on the node, doing right click on the Input icon and selecting View Data.

---

## Run the example using the Swagger interface

To run the example, deploy to tx-rest and run it using the Swagger interface.

1. Create a Web type server definition where the runtime tx-rest is installed if you do not have a server definition to deploy in Design Server.

2. If not running, start tx-rest on the server:

```
<DTX_HOME>/restapi/tomcat/dtxtomcat start tx-rest
```

3. Deploy the created package to the server definition in Design Server.

4. Bring up the tx-rest Swagger UI:

```
<DTX_HOME>/restapi/tomcat/dtxtomcat showdocs tx-rest
```

5. In the Swagger Explore window, enter /tx-rest/v2/docs and click on the Explore button to view the deployed package.

You can see a Miscellaneous section having two actions:

- **PUT** /v2/run/t2\_coco\_validation\_flow
- **POST** /v2/run/t2\_coco\_validation\_flow

6. In the **PUT** action:

- a. Expand the **PUT** action and select the Try it out button.

- b. Leave the input and output sections empty.

- c. Under flow\_vars section, delete the entire content and replace with the commands to run the flow, for example:

```
{
 "INPUT_FILE": "C:/mydir/data/t2coco_camt_009.xml",
 "OUTPUT_RESULT_JSON": "C:/mydir/data/validation_result.json",
 "REPORT_FORMAT": "json",
 "VALIDATION_TYPE": "extended"
}
```

or

```
{
 "INPUT_FILE": "C:/mydir/data/t2coco_camt_009.xml",
 "OUTPUT_RESULT_XML": "C:/mydir/data/validation_result.xml",
 "REPORT_FORMAT": "xml",
 "VALIDATION_TYPE": "extended"
}
```

- d. Click Execute blue bar for running the flow.

When flow completes execution, 200 response code is shown in the Server Response section. This run should generate the output same as above.

7. To delete the deployed package, refresh the browser to get the Swagger and explore back to /tx-rest/openapi.json.

There will be a DELETE /v2/packages/{name} action.

8. Expand the DELETE /v2/packages/{name} action and select Try it out.

9. In the Name field, enter the name you gave to the created package.

10. Select the true from the stop query drop down, then select the blue Execute bar.

You can see a response of 204 with a timestamp, if it is successful.

---

## Run the example using the flow command server process (flowcmdserver)

To run the example, deploy locally or to ftp server and use the flow command server process (flowcmdserver).

1. Create a server definition where the runtime is installed if you do not have a server local definition or ftp execution definition to deploy in Design Server.
2. In Design Server, deploy the created package to the server definition. For example, deploy to c:\ftpserver\deployment directory.
3. Execute the flow using the flow command server. The below command is for TARGET2 CoCo t2coco\_camt\_009.xml message.

```
flowcmdserver.exe --dir c:\ftpserver\deployment\flows --flow t2_coco_validation_flow.json --var
"INPUT_FILE=C:\ftpserver\deployment\tools\mx_service\data\t2coco_camt_009.xml" --audit
"C:\ftpserver\deployment\tools\mx_service\data\t2coco_camt_009_adt.json" -ad
```

The results will be in C:\ftpserver\deployment\flows as validation\_results.xml or validation\_results.json depending on the value of the REPORT\_FORMAT variable.

This command will execute with the variable VALIDATION\_TYPE value defined in the Flow settings in Design Server (extended or schema).

To change, the following command can be used:

```
flowcmdserver.exe --dir c:\ftpserver\deployment\flows --flow t2_coco_validation_flow.json --var
"INPUT_FILE=C:\ftpserver\deployment\tools\mx_service\data\t2coco_camt_009.xml" --var "VALIDATION_TYPE=schema" --var
"REPORT_FORMAT=json" --audit "C:\ftpserver\deployment\tools\mx_service\data\t2coco_camt_009_adt.json" -ad
```

4. The flow will report status similar to:

```
***Starting flow command server
```

```
Flow UUID: 85eabe2c-2302-4543-b347-9e18d65c6816
The flow audit file is: "C:\\\\ftpserver\\\\deployment\\\\tools\\\\mx_service\\\\data\\\\t2coco_camt_009_adt.json"
Flow completed successfully
Elapsed time: 3048ms
```

5. Examine the flow output result files bah\_validation\_results.xml, doc\_validation\_results.xml, and the audit file t2coco\_camt\_009\_adt.json.

## How to customize the mxconfig.xml file

Rules enforced by the MX validation framework can be customized by editing the mxconfig.xml located under the validation/mx\_extended/data folder.

All the rules enforced by the MX validation map are under element ExtendedValidation.

Any of the sub-elements under ExtendedValidation can be enabled by setting the value to T or disabled by setting to F.

### Example

To disable the BIC lookup validation feature.

### About this task

Change sub-element ExtendedValidation/BICValidation value from T to F.

### Procedure

1. Download the mxconfig.xml file from the Files tab.
2. Open the mxconfig.xml file and edit the sub-element ExtendedValidation/BICValidation value setting from T to F, then save the file.
3. Upload the edited mxconfig.xml file to the project.
4. Run the flow.

Run using an input with BIC lookup validation issue.

Right click on the output card on canvas and select View data. No BIC lookup validation reported.

## T2 CoCo enhanced MX validation (using maps) example

This example demonstrates the TARGET2 CoCo enhanced MX validation for TARGET2 CoCo messages using maps only.

The maps can perform following level of validation:

- MX extended validation includes following checks:
  - BIC lookup
  - Country code lookup
  - Currency code lookup
  - IBAN format
  - Allowed maximum fractional digits per currency
  - Usage guideline rules, see mxconfig.xml for list of rules
- Schema validation
  - Enforces validation based on XSD
- [What the example contains](#)
- [Run the example in Design Studio](#)
- [Run the example in Design Server User Interface](#)
- [How to customize the mxconfig.xml file](#)

Rules enforced by the MX validation framework can be customized by editing the mxconfig.xml located under the validation/mx\_extended/data folder.

## What the example contains

This example contains the following directories and files:

- Maps:  
The maps directory contains the following map sources to use when running under Design Studio:
  - t2\_coco\_mx\_acmt\_validation\_enh.mrms  
Utility maps called by main map for all the acmt TARGET2 CoCo xml message MX validation.
  - t2\_coco\_mx\_admi\_validation\_enh.mrms  
Utility maps called by main map for all the admi TARGET2 CoCo xml message MX validation.
  - t2\_coco\_mx\_camt\_validation\_enh.mrms  
Utility maps called by main map for all the camt TARGET2 CoCo xml message MX validation.
  - t2\_coco\_mx\_reda\_validation\_enh.mrms  
Utility maps called by main map for all the reda TARGET2 CoCo xml message MX validation.

- t2\_coco\_mx\_validation\_frmwrk\_map\_enh.mms  
The main map used to apply MX validation to TARGET2 CoCo xml messages.

Note: When running under Design Server, the main framework map is t2\_coco\_mx\_validation\_frmwrk\_map\_flw\_enh.mms.

- Schemas:  
The schemas directory contains the following files:

- bic.xsd  
Metadata that represents the bic.xml repository file structure.
- ccy.xsd  
Metadata that represents the currencycodedecimals.xml repository file structure.
- mxconfig.xsd  
Metadata that represents the mxconfig.xml configuration file structure.
- XML schemas based on SWIFT MyStandards Readiness TARGET2 CoCo Portal:
  - acmt.007.001.02.xsd
  - acmt.010.001.02.xsd
  - acmt.011.001.02.xsd
  - acmt.015.001.02.xsd
  - acmt.019.001.02.xsd
  - acmt.025.001.02.xsd
  - acmt.026.001.02.xsd
  - adm1.007.001.01.xsd
  - camt.009.001.07.xsd
  - camt.010.001.08.xsd
  - camt.011.001.07.xsd
  - camt.012.001.07.xsd
  - camt.018.001.05.xsd
  - camt.019.001.07.xsd
  - camt.024.001.06.xsd
  - camt.025.001.05.xsd
  - camt.048.001.05.xsd
  - camt.069.001.03.xsd
  - camt.070.001.04.xsd
  - camt.071.001.03.xsd
  - camt.076.001.01.xsd
  - camt.077.001.01.xsd
  - camt.099.001.01.xsd
  - camt.100.001.01.xsd
  - reda.014.001.01.xsd
  - reda.015.001.01.xsd
  - reda.016.001.01.xsd
  - reda.017.001.01.xsd
  - reda.022.001.01.xsd
  - reda.031.001.01.xsd
  - reda.039.001.01.xsd
  - reda.040.001.01.xsd
  - reda.042.001.01.xsd
  - reda.043.001.01.xsd
  - reda.064.001.01.xsd
  - reda.065.001.01.xsd
  - head.001.001.01.xsd
- XML schema based on W3C XML Signature Syntax and Processing version 1.2:
  - xmldsig-core-schema.xsd

- Trees:

The trees directory contains the following files:

- mxvalErrorReport.mtt  
Metadata that represents the xml based structure of the validation report.
- swiftroute\_funds.mtt  
Metadata that is used as an internal element placeholder.

- Data:

The data directory contains the following file:

- bic.xml  
Repository file listing all BICs which are used during validation.
- currencycodedecimals.xml  
Repository file list country codes, currency codes and corresponding maximum fractionally digits, used as reference for validation.
- mxconfig.xml  
Holds the MX configuration information on how to process the message.
- Sample TARGET2 CoCo valid files without header envelope for test purposes:
  - adm1\_007\_001\_01\_valid.xml
- Sample TARGET2 CoCo valid files with header envelope for test purposes:
  - bah\_acmt\_007\_001\_02\_valid.xml
  - bah\_acmt\_010\_001\_02\_valid.xml

- bah\_acmt\_011\_001\_02\_valid.xml
- bah\_acmt\_015\_001\_02\_valid.xml
- bah\_acmt\_019\_001\_02\_valid.xml
- bah\_acmt\_025\_001\_02\_valid.xml
- bah\_acmt\_026\_001\_02\_valid.xml
- bah\_admi\_007\_001\_01\_valid.xml
- bah\_camt\_009\_001\_07\_valid.xml
- bah\_camt\_010\_001\_08\_valid.xml
- bah\_camt\_011\_001\_07\_valid.xml
- bah\_camt\_012\_001\_07\_valid.xml
- bah\_camt\_018\_001\_05\_valid.xml
- bah\_camt\_019\_001\_07\_valid.xml
- bah\_camt\_024\_001\_06\_valid.xml
- bah\_camt\_025\_001\_05\_valid.xml
- bah\_camt\_048\_001\_05\_valid.xml
- bah\_camt\_069\_001\_03\_valid.xml
- bah\_camt\_070\_001\_04\_valid.xml
- bah\_camt\_071\_001\_03\_valid.xml
- bah\_camt\_076\_001\_01\_valid.xml
- bah\_camt\_077\_001\_01\_valid.xml
- bah\_camt\_099\_001\_01\_valid.xml
- bah\_camt\_100\_001\_01\_valid.xml
- bah\_reda\_014\_001\_01\_valid.xml
- bah\_reda\_015\_001\_01\_valid.xml
- bah\_reda\_016\_001\_01\_valid.xml
- bah\_reda\_017\_001\_01\_valid.xml
- bah\_reda\_022\_001\_01\_valid.xml
- bah\_reda\_031\_001\_01\_valid.xml
- bah\_reda\_039\_001\_01\_valid.xml
- bah\_reda\_040\_001\_01\_valid.xml
- bah\_reda\_042\_001\_01\_valid.xml
- bah\_reda\_043\_001\_01\_valid.xml
- bah\_reda\_064\_001\_01\_valid.xml
- bah\_reda\_065\_001\_01\_valid.xml

## Run the example in Design Studio

1. Build all the maps in the .mms files listed in What the example contains section.
2. Replace the input card 1 in router map, t2co9000\_val (under t2\_coco\_mx\_validation\_frmwrk\_map\_enh.mms file or t2\_coco\_mx\_validation\_frmwrk\_map\_flw\_enh.mms file if running under Design Server) with desired input file.
3. Run the main router map, t2co9000\_val.  
You will see the output file in the data folder called t2coco\_error\_report.xml.

## Run the example in Design Server User Interface

1. Open a browser pointing to the installation of the IBM Sterling Transformation Extender.
2. Enter user and password credentials.
3. If the project does not exist, import t2\_coco.zip.
4. If absent, create a package containing all the Maps, Files, and Flows.
5. Build package in desired server (to build all maps).
6. Open the project.
7. In the Maps tab, open the map t2co9000\_val.
8. Modify the settings for input card 1 on design canvas to point to the desired test file.
9. Save, build, and run the map t2co9000\_val.
10. Right click on the output card on canvas and select View data.

## How to customize the mxconfig.xml file

Rules enforced by the MX validation framework can be customized by editing the mxconfig.xml located under the validation/mx\_extended/data folder.

All the rules enforced by the MX validation map are under element ExtendedValidation.

Any of the sub-elements under ExtendedValidation can be enabled by setting the value to T or disabled by setting to F.

## Example

To disable the BIC lookup validation feature.

## About this task

Change sub-element ExtendedValidation/BICValidation value from T to F.

## Procedure

---

1. Download the mxconfig.xml file from the Files tab.
2. Open the mxconfig.xml file and edit the sub-element ExtendedValidation/BICValidation value setting from T to F, then save the file.
3. Upload the edited mxconfig.xml file to the project.
4. Run the map.  
Run using an input with BIC lookup validation issue.

Right click on the output card on canvas and select View data. No BIC lookup validation reported.

---

## T2 CoCo schema validation (using maps) example

This example demonstrates the TARGET2 CoCo schema validation for TARGET2 CoCo messages using maps only.

The maps can perform following level of validation:

- Schema validation
  - [What the example contains](#)
  - [Run the example in Design Studio](#)
  - [Run the example in Design Server User Interface](#)
- 

## What the example contains

This example contains the following directories and files:

- Maps:  
The maps directory contains the following map sources to use when running under Design Studio:
  - t2\_coco\_mx\_schema\_validation\_xsd.mms  
Utility maps called by main map for all TARGET2 CoCo messages schema validation.
  - t2\_coco\_schema\_validation\_frmwrk\_map\_xsd.mms  
The main map used to apply schema validation to TARGET2 CoCo xml messages.
- Schemas:  
The schemas directory contains the following files:
  - XML schemas based on SWIFT MyStandards Readiness TARGET2 CoCo Portal:
    - acmt.007.001.02.xsd
    - acmt.010.001.02.xsd
    - acmt.011.001.02.xsd
    - acmt.015.001.02.xsd
    - acmt.019.001.02.xsd
    - acmt.025.001.02.xsd
    - acmt.026.001.02.xsd
    - admi.007.001.01.xsd
    - camt.009.001.07.xsd
    - camt.010.001.08.xsd
    - camt.011.001.07.xsd
    - camt.012.001.07.xsd
    - camt.018.001.05.xsd
    - camt.019.001.07.xsd
    - camt.024.001.06.xsd
    - camt.025.001.05.xsd
    - camt.048.001.05.xsd
    - camt.069.001.03.xsd
    - camt.070.001.04.xsd
    - camt.071.001.03.xsd
    - camt.076.001.01.xsd
    - camt.077.001.01.xsd
    - camt.099.001.01.xsd
    - camt.100.001.01.xsd
    - reda.014.001.01.xsd
    - reda.015.001.01.xsd
    - reda.016.001.01.xsd
    - reda.017.001.01.xsd
    - reda.022.001.01.xsd
    - reda.031.001.01.xsd
    - reda.039.001.01.xsd
    - reda.040.001.01.xsd
    - reda.042.001.01.xsd
    - reda.043.001.01.xsd
    - reda.064.001.01.xsd
    - reda.065.001.01.xsd
    - head.001.001.01.xsd

- XML schemas based on W3C XML Signature Syntax and Processing version 1.2:
    - xmldsig-core-schema.xsd
- Trees:  
The trees directory contains the following files:
  - mxvalErrorReport.mtt  
Metadata that represents the xml based structure of the validation report.
  - swiftroute\_funds.mtt  
Metadata that is used as an internal element placeholder.
- Data:  
The data directory contains the following file:
  - Sample TARGET2 CoCo valid files without header envelope for test purposes:
    - admi\_007\_001\_01\_valid.xml
  - Sample TARGET2 CoCo valid files with header envelope for test purposes:
    - bah\_acmt\_007\_001\_02\_valid.xml
    - bah\_acmt\_010\_001\_02\_valid.xml
    - bah\_acmt\_011\_001\_02\_valid.xml
    - bah\_acmt\_015\_001\_02\_valid.xml
    - bah\_acmt\_019\_001\_02\_valid.xml
    - bah\_acmt\_025\_001\_02\_valid.xml
    - bah\_acmt\_026\_001\_02\_valid.xml
    - bah\_admi\_007\_001\_01\_valid.xml
    - bah\_camt\_009\_001\_07\_valid.xml
    - bah\_camt\_010\_001\_08\_valid.xml
    - bah\_camt\_011\_001\_07\_valid.xml
    - bah\_camt\_012\_001\_07\_valid.xml
    - bah\_camt\_018\_001\_05\_valid.xml
    - bah\_camt\_019\_001\_07\_valid.xml
    - bah\_camt\_024\_001\_06\_valid.xml
    - bah\_camt\_025\_001\_05\_valid.xml
    - bah\_camt\_048\_001\_05\_valid.xml
    - bah\_camt\_069\_001\_03\_valid.xml
    - bah\_camt\_070\_001\_04\_valid.xml
    - bah\_camt\_071\_001\_03\_valid.xml
    - bah\_camt\_076\_001\_01\_valid.xml
    - bah\_camt\_077\_001\_01\_valid.xml
    - bah\_camt\_099\_001\_01\_valid.xml
    - bah\_camt\_100\_001\_01\_valid.xml
    - bah\_reda\_014\_001\_01\_valid.xml
    - bah\_reda\_015\_001\_01\_valid.xml
    - bah\_reda\_016\_001\_01\_valid.xml
    - bah\_reda\_017\_001\_01\_valid.xml
    - bah\_reda\_022\_001\_01\_valid.xml
    - bah\_reda\_031\_001\_01\_valid.xml
    - bah\_reda\_039\_001\_01\_valid.xml
    - bah\_reda\_040\_001\_01\_valid.xml
    - bah\_reda\_042\_001\_01\_valid.xml
    - bah\_reda\_043\_001\_01\_valid.xml
    - bah\_reda\_064\_001\_01\_valid.xml
    - bah\_reda\_065\_001\_01\_valid.xml

## Run the example in Design Studio

1. Build all the maps in the .mms files listed in What the example contains section.
2. Replace the input card 1 in router map, t2co9500\_val (under t2\_coco\_schema\_validation\_frmwrk\_map\_xsd.mms file) with desired input file.
3. Run the main router map, t2co9500\_val.  
You will see the output file in the data folder called t2coco\_error\_report.xml.

## Run the example in Design Server User Interface

1. Open a browser pointing to the installation of the IBM Sterling Transformation Extender.
2. Enter user and password credentials.
3. If the project does not exist, import t2\_coco.zip.
4. If absent, create a package containing all the Maps, Files, and Flows.
5. Build package in desired server (to build all maps).
6. Open the project.
7. In the Maps tab, open the map t2co9500\_val.
8. Modify the settings for input card 1 on design canvas to point to the desired test file.
9. Save, build, and run the map t2co9500\_val.
10. Right click on the output card on canvas and select View data.

---

## TARGET2 Securities Examples:

TARGET2 Securities examples are as follow:

- [\*\*T2 Securities \(T2S\) schema validation \(using Flow Server\) example\*\*](#)  
This example demonstrates the TARGET2 Securities (T2S) schema validation for T2S messages using flow server.
  - [\*\*T2 Securities \(T2S\) schema validation \(using maps\) example\*\*](#)  
This example demonstrates the TARGET2 Securities (T2S) schema validation for T2S messages using maps only.
- 

## T2 Securities (T2S) schema validation (using Flow Server) example

This example demonstrates the TARGET2 Securities (T2S) schema validation for T2S messages using flow server.

The flow can perform following level of validation:

- Schema validation
  - [\*\*What the example contains\*\*](#)  
Files included in this example are as follows:
  - [\*\*How to run the example \(using t2\\_sec\\_validation\\_flow\)\*\*](#)  
This TARGET2 Security (T2S) schema validation will use the sample files to demonstrate the generation of validation reports as output from a T2S XML message.
  - [\*\*How to run the example \(using t2\\_sec\\_head002\\_validation\\_flow\)\*\*](#)  
This TARGET2 Security (T2S) schema validation will allow validation of files using a different header allowing for multiple payloads. Use a sample file to demonstrate the generation of validation reports as output from a T2S XML message using head\_002.001.01.
- 

## What the example contains

Files included in this example are as follows:

- t2\_sec.zip
  - Test Data Files:
    - t2se\_admi\_005.xml
    - t2se\_admi\_005\_schema.xml
  - Schemas:
    - mxvalErrorReport.mtt  
Metadata that represents the xml based structure of the validation report.
    - swiftroute\_funds.mtt  
Metadata that is used as an internal element placeholder.
    - XML schemas based on SWIFT MyStandards Readiness Target 2 Securities (T2S) Portal:
      - admi.005.001.01.xsd
      - admi.006.001.01.xsd
      - admi.007.001.01.xsd
      - camt.003.001.07.xsd
      - camt.004.001.08.xsd
      - camt.005.001.08.xsd
      - camt.006.001.08.xsd
      - camt.009.001.07.xsd
      - camt.010.001.08.xsd
      - camt.019.001.07.xsd
      - camt.025.001.05.xsd
      - camt.050.001.05.xsd
      - camt.051.001.05.xsd
      - camt.052.001.08.xsd
      - camt.053.001.08.xsd
      - camt.054.001.08.xsd
      - camt.064.001.01.xsd
      - camt.065.001.01.xsd
      - camt.066.001.01.xsd
      - camt.067.001.01.xsd
      - camt.068.001.01.xsd
      - camt.069.001.03.xsd
      - camt.070.001.04.xsd
      - camt.072.001.01.xsd
      - camt.073.001.01.xsd
      - camt.074.001.01.xsd
      - camt.075.001.01.xsd
      - camt.078.001.01.xsd
      - camt.079.001.01.xsd
      - camt.080.001.01.xsd
      - camt.081.001.01.xsd
      - camt.082.001.01.xsd

- camt.083.001.01.xsd
- camt.084.001.01.xsd
- camt.085.001.01.xsd
- colr.001.001.01.xsd
- colr.002.001.01.xsd
- head.001.001.01.xsd
- head.002.001.01.xsd
- semt.002.001.10.xsd
- semt.013.001.04.xsd
- semt.014.001.06.xsd
- semt.015.001.07.xsd
- semt.016.001.07.xsd
- semt.017.001.09.xsd
- semt.018.001.10.xsd
- semt.019.001.08.xsd
- semt.020.001.05.xsd
- semt.022.001.04.xsd
- semt.025.001.01.xsd
- semt.026.001.01.xsd
- semt.027.001.01.xsd
- semt.028.001.01.xsd
- semt.029.001.01.xsd
- semt.030.001.01.xsd
- semt.031.001.01.xsd
- semt.032.001.01.xsd
- semt.033.001.01.xsd
- semt.034.001.01.xsd
- semt.040.001.01.xsd
- semt.044.001.01.xsd
- sese.020.001.06.xsd
- sese.021.001.05.xsd
- sese.022.001.05.xsd
- sese.023.001.09.xsd
- sese.024.001.10.xsd
- sese.025.001.09.xsd
- sese.027.001.05.xsd
- sese.028.001.08.xsd
- sese.029.001.04.xsd
- sese.030.001.08.xsd
- sese.031.001.08.xsd
- sese.032.001.09.xsd
- supl.021.001.01.xsd

- XML schema based on W3C XML Signature Syntax and Processing version 1.2:

- xmldsig-core-schema.xsd

Note: The schema xmldsig-core-schema.xsd is imported inside of schema head.001.001.01.xsd to handle signatures. It is imported two times as a FILE to the project. Once in the folder location t2\_sec/validation/schema\_only/schemas and a second copy without any logical folder.

Note: The schema supl.021.001.01.xsd is imported inside the following schemas:

- camt.067.001.01.xsd
- camt.068.001.01.xsd
- semt.014.001.06.xsd
- semt.015.001.07.xsd
- sese.024.001.10.xsd
- sese.025.001.09.xsd
- sese.032.001.09.xsd

It is imported into the project two times as a SCHEMA. Once in the folder t2\_sec/validation/schema\_only/schemas and a second copy without any logical folder.

- Maps:

- For Schema Validation:
  - t2se9500\_val
  - t2se9501\_admi\_005\_001\_01
  - t2se9502\_admi\_006\_001\_01
  - t2se9503\_admi\_007\_001\_01
  - t2se9504\_camt\_003\_001\_07
  - t2se9505\_camt\_004\_001\_08
  - t2se9506\_camt\_005\_001\_08
  - t2se9507\_camt\_006\_001\_08
  - t2se9508\_camt\_009\_001\_07
  - t2se9509\_camt\_010\_001\_08
  - t2se9510\_camt\_019\_001\_07
  - t2se9511\_camt\_025\_001\_05
  - t2se9512\_camt\_050\_001\_05
  - t2se9513\_camt\_051\_001\_05
  - t2se9514\_camt\_052\_001\_08
  - t2se9515\_camt\_053\_001\_08
  - t2se9516\_camt\_054\_001\_08
  - t2se9517\_camt\_064\_001\_01
  - t2se9518\_camt\_065\_001\_01
  - t2se9519\_camt\_066\_001\_01

- t2se9520\_camt\_067\_001\_01
  - t2se9521\_camt\_068\_001\_01
  - t2se9522\_camt\_069\_001\_03
  - t2se9523\_camt\_070\_001\_04
  - t2se9524\_camt\_072\_001\_01
  - t2se9525\_camt\_073\_001\_01
  - t2se9526\_camt\_074\_001\_01
  - t2se9527\_camt\_075\_001\_01
  - t2se9528\_camt\_078\_001\_01
  - t2se9529\_camt\_079\_001\_01
  - t2se9530\_camt\_080\_001\_01
  - t2se9531\_camt\_081\_001\_01
  - t2se9532\_camt\_082\_001\_01
  - t2se9533\_camt\_083\_001\_01
  - t2se9534\_camt\_084\_001\_01
  - t2se9535\_camt\_085\_001\_01
  - t2se9536\_colr\_001\_001\_01
  - t2se9537\_colr\_002\_001\_01
  - t2se9538\_semt\_002\_001\_10
  - t2se9539\_semt\_013\_001\_04
  - t2se9540\_semt\_014\_001\_06
  - t2se9541\_semt\_015\_001\_07
  - t2se9542\_semt\_016\_001\_07
  - t2se9543\_semt\_017\_001\_09
  - t2se9544\_semt\_018\_001\_10
  - t2se9545\_semt\_019\_001\_08
  - t2se9546\_semt\_020\_001\_05
  - t2se9547\_semt\_022\_001\_04
  - t2se9548\_semt\_025\_001\_01
  - t2se9549\_semt\_026\_001\_01
  - t2se9550\_semt\_027\_001\_01
  - t2se9551\_semt\_028\_001\_01
  - t2se9552\_semt\_029\_001\_01
  - t2se9553\_semt\_030\_001\_01
  - t2se9554\_semt\_031\_001\_01
  - t2se9555\_semt\_032\_001\_01
  - t2se9556\_semt\_033\_001\_01
  - t2se9557\_semt\_034\_001\_01
  - t2se9558\_semt\_040\_001\_01
  - t2se9559\_semt\_044\_001\_01
  - t2se9560\_sese\_020\_001\_06
  - t2se9561\_sese\_021\_001\_05
  - t2se9562\_sese\_022\_001\_05
  - t2se9563\_sese\_023\_001\_09
  - t2se9564\_sese\_024\_001\_10
  - t2se9565\_sese\_025\_001\_09
  - t2se9566\_sese\_027\_001\_05
  - t2se9567\_sese\_028\_001\_08
  - t2se9568\_sese\_029\_001\_04
  - t2se9569\_sese\_030\_001\_08
  - t2se9570\_sese\_031\_001\_08
  - t2se9571\_sese\_032\_001\_09
  - t2se9572\_supl\_021\_001\_01
  - t2se9600\_val
  - t2se9951\_head\_001\_001\_01
- Common:
    - mxut1003\_bizsvc\_t2sec
  - Flows:
    - t2\_sec\_validation\_flow
    - t2\_sec\_head002\_validation\_flow

## How to run the example (using t2\_sec\_validation\_flow)

This TARGET2 Security (T2S) schema validation will use the sample files to demonstrate the generation of validation reports as output from a T2S XML message.

The stopValidation.json file will report the validation failure due to the pre-conversion checks:

**If input file is other than any of the supported T2S messages as per schema list indicated above.**

1. Import the t2\_sec.zip project into the Design Server.
2. Open the t2\_sec project in Design Server and view the flow t2\_sec\_validation\_flow.
  - a. The Flow Description points to all the maps to be executed during the flow process, which will be called from within some of the map nodes in the flow. This list has to be added to the description and ensure all lines start in column 1. Following is the list of maps with the description format:
    - @packagemap=t2\_sec/validation/schema\_only/maps/t2\_sec\_mx\_schema\_validation\_xsd/t2se9501\_admi\_005\_001\_01
    - @packagemap=t2\_sec/validation/schema\_only/maps/t2\_sec\_mx\_schema\_validation\_xsd/t2se9502\_admi\_006\_001\_01
    - @packagemap=t2\_sec/validation/schema\_only/maps/t2\_sec\_mx\_schema\_validation\_xsd/t2se9503\_admi\_007\_001\_01
    - @packagemap=t2\_sec/validation/schema\_only/maps/t2\_sec\_mx\_schema\_validation\_xsd/t2se9504\_camt\_003\_001\_07
    - @packagemap=t2\_sec/validation/schema\_only/maps/t2\_sec\_mx\_schema\_validation\_xsd/t2se9505\_camt\_004\_001\_08

- @packagenode=tz\_sec

b. It utilizes the following nodes:

#### i. Source Nodes:

- mx input

This node identifies the input data to be validated in the flow. It uses the variable INPUT\_FILE to set the location of the data.

## ii) Map Nodes:

- MX pre-check

Runs map mxut1003 bizsvc t2sec which sets flow variables, checks pre-validation conditions, and creates `infoset.json` for validation.

- XSD VAL

`Runs map t2se9500_val which calls the appropriate map to perform schema validation of a T2S Plus message.`

Note: The map t2se9500\_val internally calls all the other maps identified in the above Flow Description step.

iii. Decision Nodes:

- pre-valid chk  
This node checks the flow variable stopTranslation to decide if further validation/processing is not required.
- XSD\_RESULT\_FORMAT  
This node checks the value of the flow variable REPORT\_FORMAT to determine if the resulting report for schema-only validation should be generated in XML (default) or JSON.

iv. Log Nodes:

- FAILURE  
This node creates a log file specified by the flow variable FAILURE\_LOG in case of no validation due to precondition check failures.

v. Fail Node:

- STOP  
It generates error message setup in the node in case of no validation performed.

vi. Passthrough Nodes:

- XSD\_XML\_CONVERT  
This node receives a schema-only validation report in XML format and does not modify the content, thus passing it as is to the appropriate target node.

vii. Format Converter Nodes:

- XSD\_JSON\_CONVERT  
This node receives a schema-only validation report in XML format and converts it to JSON before passing it to the appropriate target node.

viii. Target Nodes:

- xsd\_xml  
This node contains the resulting schema-only validation report in XML format and creates the output as defined by the variable OUTPUT\_RESULT\_XML.

- xsd\_json  
This node contains the resulting schema-only validation report in JSON format and creates the output as defined by the variable OUTPUT\_RESULT\_JSON.

c. It utilizes the following variables:

i. Flow variables:

- VALIDATION\_TYPE  
Default value is schema. The value extended is not supported at this time. This variable is currently not being used in any node.
- REPORT\_FORMAT  
Default value is xml. It can be changed to json. This is used in Decision node XSD\_RESULT\_FORMAT.
- INPUT\_FILE  
Default value is ../tools/mx\_service/data/t2sec\_admi\_005.xml. This is the data file to be used for validation and can be customized. It is used in the Source node mx\_input.
- BIC\_FILE  
Default value is ../data/bic.xml. This is the location of the bic cross-reference file that is used in validation maps. Since this flow is only doing schema validation, this variable is not used.
- CCY\_FILE  
Default value is ../data/currencycodedecimals.xml. This is the location of the country code cross-reference file that is used in validation maps. Since this flow is only doing schema validation, this variable is not used.
- MXCONFIG\_FILE  
Default value is ../data/mxconfig.xml. This is the location of the file containing the rule validation settings that is used in validation maps. Since this flow is only doing schema validation, this variable is not used.
- OUTPUT\_RESULT\_XML  
Default value is validation\_result.xml. This is the location of the validation report file in xml format. It is used in Target node xsd\_xml. It can be customized.
- OUTPUT\_RESULT\_JSON  
Default value is validation\_result.json. This is the location of the validation report file in json format. It is used in Target node xsd\_json. It can be customized.
- FAILURE\_LOG  
Default value is stopMXValidation.json. This is the location of the failure log. It is used in Log node FAILURE. It can be customized.
- bizSvc  
For internal use in the Map node MX pre-check to determine the type of data being read in the input file.
- stopValidation  
For internal use in the Map node MX pre-check and is checked in Decision node pre-valid chk to cause a Failure log in case a data file was not recognized as a valid T2S message.

3. Open the main flow t2\_sec\_validation\_flow in the Design Server. It utilizes above one Source node, two Map nodes, two Decision nodes, one Log node, one Fail node, one Passthrough nodes, one Format Converter nodes, and two Target nodes.

4. In Design Server, create a package t2\_sec\_validation\_flow that contains the input files and the flow t2\_sec\_validation\_flow. The maps will automatically be included during deployment of the package onto the runtime server.

Note: Not required for running flows on Design Server user interface directly.

• [Run the example using three different methods](#)

You can run the example using three different methods:

## Run the example using three different methods

You can run the example using three different methods:

1. Running the example from the Design Server user interface.
2. Deploying the example to tx-rest and running it using the Swagger interface.
3. Deploying locally or to ftp server and using the flow command server process (flowcmdserver).

- **[Run the example from the Design Server user interface](#)**

Use the following steps to run the example from the Design Server user interface:

- **[Run the example using the Swagger interface](#)**

To run the example, deploy to tx-rest and run it using the Swagger interface.

- **[Run the example using the flow command server process \(flowcmdserver\)](#)**

To run the example, deploy locally or to ftp server and use the flow command server process (flowcmdserver).

## Run the example from the Design Server user interface

Use the following steps to run the example from the Design Server user interface:

1. In the Design Server, open t2\_sec\_validation\_flow flow and click on Run button.
2. Set the toggle switch Run On Design Server and select flow input file. Example is t2\_sec\_admi\_005.xml.  
Note: If a default path is assigned to the variable INPUT\_FILE, the value defined for the variable will be used if an input file is not selected.  
Flow will run and report with the green check box. The report lists execution of all nodes.
3. Right click and select View Link Data on each link to examine the data.
4. View target node by clicking on the node, doing right click on the Input icon and selecting View Data.

## Run the example using the Swagger interface

To run the example, deploy to tx-rest and run it using the Swagger interface.

1. Create a Web type server definition where the runtime tx-rest is installed if you do not have a server definition to deploy in Design Server.

2. If not running, start tx-rest on the server:  
`<DTX_HOME>/restapi/tomcat/dtxtomcat start tx-rest`

3. Deploy the created package to the server definition in Design Server.

4. Bring up the tx-rest Swagger UI:  
`<DTX_HOME>/restapi/tomcat/dtxtomcat showdocs tx-rest`

5. In the Swagger Explore window, enter /tx-rest/v2/docs and click on the Explore button to view the deployed package.

You can see a Miscellaneous section having two actions:

- **PUT** /v2/run/t2\_sec\_validation\_flow
- **POST** /v2/run/t2\_sec\_validation\_flow

6. In the **PUT** action:

a. Expand the **PUT** action and select the Try it out button.

b. Leave the input and output sections empty.

c. Under flow\_vars section, delete the entire content and replace with the commands to run the flow, for example:

```
{
 "INPUT_FILE": "C:/mydir/data/t2_sec_admi_005.xml",
 "OUTPUT_RESULT_JSON": "C:/mydir/data/validation_result.json",
 "REPORT_FORMAT": "json"
}
```

or

```
{
 "INPUT_FILE": "C:/mydir/data/t2_sec_admi_005.xml",
 "OUTPUT_RESULT_XML": "C:/mydir/data/validation_result.xml",
 "REPORT_FORMAT": "xml"
}
```

d. Click Execute blue bar for running the flow.

When flow completes execution, 200 response code is shown in the Server Response section. This run should generate the output same as above.

7. To delete the deployed package, refresh the browser to get the Swagger and explore back to /tx-rest/openapi.json.

There will be a DELETE /v2/packages/{name} action.

8. Expand the DELETE /v2/packages/{name} action and select Try it out.

9. In the Name field, enter the name you gave to the created package.

10. Select the true option from the stop query drop down, then select the blue Execute bar.

You can see a response of 204 with a timestamp, if it is successful.

## Run the example using the flow command server process (flowcmdserver)

To run the example, deploy locally or to ftp server and use the flow command server process (flowcmdserver).

1. Create a server definition where the runtime is installed if you do not have a server local definition or ftp execution definition to deploy in Design Server.
2. In Design Server, deploy the created package to the server definition. For example, deploy to c:\ftpserver\deployment directory.
3. Execute the flow using the flow command server. The below command is for TARGET2 Securities t2\_sec\_pacs\_009.xml message.

```
flowcmdserver.exe --dir c:\ftpserver\deployment\flows --flow t2_sec_validation_flow.json --var
"INPUT_FILE=C:\ftpserver\deployment\tools\mx_service\data\t2_sec_admi_005.xml" --audit
"C:\ftpserver\deployment\tools\mx_service\data\t2_sec_admi_005.adt.json" -ad
```

The results will be in C:\ftpserver\deployment\flows as validation\_results.xml or validation\_results.json depending on the value of the REPORT\_FORMAT variable.

This command will execute with the variable VALIDATION\_TYPE value defined in the Flow settings in Design Server (only schema is supported).

To change, the following command can be used:

```
flowcmdserver.exe --dir c:\ftpserver\deployment\flows --flow t2_sec_validation_flow.json --var
"INPUT_FILE=C:\ftpserver\deployment\tools\mx_service\data\t2_sec_admi_005.xml" --var "REPORT_FORMAT=json" --audit
"C:\ftpserver\deployment\tools\mx_service\data\t2_sec_admi_005.adt.json" -ad
```

4. The flow will report status similar to:

```
***Starting flow command server

Flow UUID: 85eabe2c-2302-4543-b347-9e18d65c6816
The flow audit file is: "C:\\\\ftpserver\\\\deployment\\\\tools\\\\mx_service\\\\data\\\\t2_sec_admi_005.adt.json"
Flow completed successfully
Elapsed time: 3048ms
```

5. Examine the flow output result files validation\_results.xml and the audit file t2\_sec\_admi\_005.adt.json.

## How to run the example (using t2\_sec\_head002\_validation\_flow)

This TARGET2 Security (T2S) schema validation will allow validation of files using a different header allowing for multiple payloads. Use a sample file to demonstrate the generation of validation reports as output from a T2S XML message using head\_002.001.01.

1. Import the t2\_sec.zip project into the Design Server.
2. Open the t2\_sec project in Design Server and view the flow t2\_sec\_head002\_validation\_flow.
  - a. The Flow Description points to all the maps to be executed during the flow process, which will be called from within some of the map nodes in the flow. This list has to be added to the description and ensure all lines start in column 1. Following is the list of maps with the description format:
    - @packagemap=t2\_sec/validation/schema\_only/maps/t2\_sec\_schema\_validation\_frmwrk\_map\_xsd/t2se9500\_val
    - @packagemap=t2\_sec/validation/schema\_only/maps/t2\_sec\_mx\_schema\_validation\_xsd/t2se9501\_admi\_001\_01
    - @packagemap=t2\_sec/validation/schema\_only/maps/t2\_sec\_mx\_schema\_validation\_xsd/t2se9502\_admi\_006\_001\_01
    - @packagemap=t2\_sec/validation/schema\_only/maps/t2\_sec\_mx\_schema\_validation\_xsd/t2se9503\_admi\_007\_001\_01
    - @packagemap=t2\_sec/validation/schema\_only/maps/t2\_sec\_mx\_schema\_validation\_xsd/t2se9504\_camt\_003\_001\_07
    - @packagemap=t2\_sec/validation/schema\_only/maps/t2\_sec\_mx\_schema\_validation\_xsd/t2se9505\_camt\_004\_001\_08
    - @packagemap=t2\_sec/validation/schema\_only/maps/t2\_sec\_mx\_schema\_validation\_xsd/t2se9506\_camt\_005\_001\_08
    - @packagemap=t2\_sec/validation/schema\_only/maps/t2\_sec\_mx\_schema\_validation\_xsd/t2se9507\_camt\_006\_001\_08
    - @packagemap=t2\_sec/validation/schema\_only/maps/t2\_sec\_mx\_schema\_validation\_xsd/t2se9508\_camt\_009\_001\_07
    - @packagemap=t2\_sec/validation/schema\_only/maps/t2\_sec\_mx\_schema\_validation\_xsd/t2se9509\_camt\_010\_001\_08
    - @packagemap=t2\_sec/validation/schema\_only/maps/t2\_sec\_mx\_schema\_validation\_xsd/t2se9510\_camt\_019\_001\_07
    - @packagemap=t2\_sec/validation/schema\_only/maps/t2\_sec\_mx\_schema\_validation\_xsd/t2se9511\_camt\_025\_001\_05
    - @packagemap=t2\_sec/validation/schema\_only/maps/t2\_sec\_mx\_schema\_validation\_xsd/t2se9512\_camt\_050\_001\_05
    - @packagemap=t2\_sec/validation/schema\_only/maps/t2\_sec\_mx\_schema\_validation\_xsd/t2se9513\_camt\_051\_001\_05
    - @packagemap=t2\_sec/validation/schema\_only/maps/t2\_sec\_mx\_schema\_validation\_xsd/t2se9514\_camt\_052\_001\_08
    - @packagemap=t2\_sec/validation/schema\_only/maps/t2\_sec\_mx\_schema\_validation\_xsd/t2se9515\_camt\_053\_001\_08
    - @packagemap=t2\_sec/validation/schema\_only/maps/t2\_sec\_mx\_schema\_validation\_xsd/t2se9516\_camt\_054\_001\_08
    - @packagemap=t2\_sec/validation/schema\_only/maps/t2\_sec\_mx\_schema\_validation\_xsd/t2se9517\_camt\_064\_001\_01
    - @packagemap=t2\_sec/validation/schema\_only/maps/t2\_sec\_mx\_schema\_validation\_xsd/t2se9518\_camt\_065\_001\_01
    - @packagemap=t2\_sec/validation/schema\_only/maps/t2\_sec\_mx\_schema\_validation\_xsd/t2se9519\_camt\_066\_001\_01
    - @packagemap=t2\_sec/validation/schema\_only/maps/t2\_sec\_mx\_schema\_validation\_xsd/t2se9520\_camt\_067\_001\_01
    - @packagemap=t2\_sec/validation/schema\_only/maps/t2\_sec\_mx\_schema\_validation\_xsd/t2se9521\_camt\_068\_001\_01
    - @packagemap=t2\_sec/validation/schema\_only/maps/t2\_sec\_mx\_schema\_validation\_xsd/t2se9522\_camt\_069\_001\_03
    - @packagemap=t2\_sec/validation/schema\_only/maps/t2\_sec\_mx\_schema\_validation\_xsd/t2se9523\_camt\_070\_001\_04
    - @packagemap=t2\_sec/validation/schema\_only/maps/t2\_sec\_mx\_schema\_validation\_xsd/t2se9524\_camt\_072\_001\_01
    - @packagemap=t2\_sec/validation/schema\_only/maps/t2\_sec\_mx\_schema\_validation\_xsd/t2se9525\_camt\_073\_001\_01
    - @packagemap=t2\_sec/validation/schema\_only/maps/t2\_sec\_mx\_schema\_validation\_xsd/t2se9526\_camt\_074\_001\_01
    - @packagemap=t2\_sec/validation/schema\_only/maps/t2\_sec\_mx\_schema\_validation\_xsd/t2se9527\_camt\_075\_001\_01
    - @packagemap=t2\_sec/validation/schema\_only/maps/t2\_sec\_mx\_schema\_validation\_xsd/t2se9528\_camt\_078\_001\_01
    - @packagemap=t2\_sec/validation/schema\_only/maps/t2\_sec\_mx\_schema\_validation\_xsd/t2se9529\_camt\_079\_001\_01
    - @packagemap=t2\_sec/validation/schema\_only/maps/t2\_sec\_mx\_schema\_validation\_xsd/t2se9530\_camt\_080\_001\_01
    - @packagemap=t2\_sec/validation/schema\_only/maps/t2\_sec\_mx\_schema\_validation\_xsd/t2se9531\_camt\_081\_001\_01
    - @packagemap=t2\_sec/validation/schema\_only/maps/t2\_sec\_mx\_schema\_validation\_xsd/t2se9532\_camt\_082\_001\_01
    - @packagemap=t2\_sec/validation/schema\_only/maps/t2\_sec\_mx\_schema\_validation\_xsd/t2se9533\_camt\_083\_001\_01
    - @packagemap=t2\_sec/validation/schema\_only/maps/t2\_sec\_mx\_schema\_validation\_xsd/t2se9534\_camt\_084\_001\_01
    - @packagemap=t2\_sec/validation/schema\_only/maps/t2\_sec\_mx\_schema\_validation\_xsd/t2se9535\_camt\_085\_001\_01
    - @packagemap=t2\_sec/validation/schema\_only/maps/t2\_sec\_mx\_schema\_validation\_xsd/t2se9536\_colr\_001\_001\_01
    - @packagemap=t2\_sec/validation/schema\_only/maps/t2\_sec\_mx\_schema\_validation\_xsd/t2se9537\_colr\_002\_001\_01
    - @packagemap=t2\_sec/validation/schema\_only/maps/t2\_sec\_mx\_schema\_validation\_xsd/t2se9538\_semt\_002\_001\_10
    - @packagemap=t2\_sec/validation/schema\_only/maps/t2\_sec\_mx\_schema\_validation\_xsd/t2se9539\_semt\_013\_001\_04
    - @packagemap=t2\_sec/validation/schema\_only/maps/t2\_sec\_mx\_schema\_validation\_xsd/t2se9540\_semt\_014\_001\_06
    - @packagemap=t2\_sec/validation/schema\_only/maps/t2\_sec\_mx\_schema\_validation\_xsd/t2se9541\_semt\_015\_001\_07
    - @packagemap=t2\_sec/validation/schema\_only/maps/t2\_sec\_mx\_schema\_validation\_xsd/t2se9542\_semt\_016\_001\_07

- @packagemap=t2\_sec/validation/schema\_only/maps/t2\_sec\_mx\_schema\_validation\_xsd/t2se9543\_semt\_017\_001\_09
- @packagemap=t2\_sec/validation/schema\_only/maps/t2\_sec\_mx\_schema\_validation\_xsd/t2se9544\_semt\_018\_001\_10
- @packagemap=t2\_sec/validation/schema\_only/maps/t2\_sec\_mx\_schema\_validation\_xsd/t2se9545\_semt\_019\_001\_08
- @packagemap=t2\_sec/validation/schema\_only/maps/t2\_sec\_mx\_schema\_validation\_xsd/t2se9546\_semt\_020\_001\_05
- @packagemap=t2\_sec/validation/schema\_only/maps/t2\_sec\_mx\_schema\_validation\_xsd/t2se9547\_semt\_022\_001\_04
- @packagemap=t2\_sec/validation/schema\_only/maps/t2\_sec\_mx\_schema\_validation\_xsd/t2se9548\_semt\_025\_001\_01
- @packagemap=t2\_sec/validation/schema\_only/maps/t2\_sec\_mx\_schema\_validation\_xsd/t2se9549\_semt\_026\_001\_01
- @packagemap=t2\_sec/validation/schema\_only/maps/t2\_sec\_mx\_schema\_validation\_xsd/t2se9550\_semt\_027\_001\_01
- @packagemap=t2\_sec/validation/schema\_only/maps/t2\_sec\_mx\_schema\_validation\_xsd/t2se9551\_semt\_028\_001\_01
- @packagemap=t2\_sec/validation/schema\_only/maps/t2\_sec\_mx\_schema\_validation\_xsd/t2se9552\_semt\_029\_001\_01
- @packagemap=t2\_sec/validation/schema\_only/maps/t2\_sec\_mx\_schema\_validation\_xsd/t2se9553\_semt\_030\_001\_01
- @packagemap=t2\_sec/validation/schema\_only/maps/t2\_sec\_mx\_schema\_validation\_xsd/t2se9554\_semt\_031\_001\_01
- @packagemap=t2\_sec/validation/schema\_only/maps/t2\_sec\_mx\_schema\_validation\_xsd/t2se9555\_semt\_032\_001\_01
- @packagemap=t2\_sec/validation/schema\_only/maps/t2\_sec\_mx\_schema\_validation\_xsd/t2se9556\_semt\_033\_001\_01
- @packagemap=t2\_sec/validation/schema\_only/maps/t2\_sec\_mx\_schema\_validation\_xsd/t2se9557\_semt\_034\_001\_01
- @packagemap=t2\_sec/validation/schema\_only/maps/t2\_sec\_mx\_schema\_validation\_xsd/t2se9558\_semt\_040\_001\_01
- @packagemap=t2\_sec/validation/schema\_only/maps/t2\_sec\_mx\_schema\_validation\_xsd/t2se9559\_semt\_044\_001\_01
- @packagemap=t2\_sec/validation/schema\_only/maps/t2\_sec\_mx\_schema\_validation\_xsd/t2se9560\_sese\_020\_001\_06
- @packagemap=t2\_sec/validation/schema\_only/maps/t2\_sec\_mx\_schema\_validation\_xsd/t2se9561\_sese\_021\_001\_05
- @packagemap=t2\_sec/validation/schema\_only/maps/t2\_sec\_mx\_schema\_validation\_xsd/t2se9562\_sese\_022\_001\_05
- @packagemap=t2\_sec/validation/schema\_only/maps/t2\_sec\_mx\_schema\_validation\_xsd/t2se9563\_sese\_023\_001\_09
- @packagemap=t2\_sec/validation/schema\_only/maps/t2\_sec\_mx\_schema\_validation\_xsd/t2se9564\_sese\_024\_001\_10
- @packagemap=t2\_sec/validation/schema\_only/maps/t2\_sec\_mx\_schema\_validation\_xsd/t2se9565\_sese\_025\_001\_09
- @packagemap=t2\_sec/validation/schema\_only/maps/t2\_sec\_mx\_schema\_validation\_xsd/t2se9566\_sese\_027\_001\_05
- @packagemap=t2\_sec/validation/schema\_only/maps/t2\_sec\_mx\_schema\_validation\_xsd/t2se9567\_sese\_028\_001\_08
- @packagemap=t2\_sec/validation/schema\_only/maps/t2\_sec\_mx\_schema\_validation\_xsd/t2se9568\_sese\_029\_001\_04
- @packagemap=t2\_sec/validation/schema\_only/maps/t2\_sec\_mx\_schema\_validation\_xsd/t2se9569\_sese\_030\_001\_08
- @packagemap=t2\_sec/validation/schema\_only/maps/t2\_sec\_mx\_schema\_validation\_xsd/t2se9570\_sese\_031\_001\_08
- @packagemap=t2\_sec/validation/schema\_only/maps/t2\_sec\_mx\_schema\_validation\_xsd/t2se9571\_sese\_032\_001\_09
- @packagemap=t2\_sec/validation/schema\_only/maps/t2\_sec\_mx\_schema\_validation\_xsd/t2se9572\_supl\_021\_001\_01
- @packagemap=t2\_sec/validation/schema\_only/maps/t2\_sec\_mx\_schema\_validation\_xsd/t2se9951\_head\_001\_001\_01

b. It utilizes the following nodes:

i. Source Nodes:

- mx\_input

This node identifies the input data to be validated in the flow. It uses the variable INPUT\_FILE to set the location of the data.

ii. Map Nodes:

- XSD\_VAL

Runs map t2se9600\_val which calls the appropriate map to perform schema validation using head\_002\_001\_01.

Note: The maps t2se9600\_val call all the other maps identified in the above Flow Description step.

iii. Decision Nodes:

- XSD\_RESULT\_FORMAT

This node checks the value of the flow variable REPORT\_FORMAT to determine if the resulting report for schema-only validation should be generated in XML (default) or JSON.

iv. Passthrough Nodes:

- XSD\_XML\_CONVERT

This node receives a schema-only validation report in XML format and does not modify the content, thus passing it as is to the appropriate target node.

v. Format Converter Nodes:

- XSD\_JSON\_CONVERT

This node receives a schema-only validation report in XML format and converts it to JSON before passing it to the appropriate target node.

vi. Target Nodes:

- xsd\_xml

This node contains the resulting schema-only validation report in XML format and creates the output as defined by the variable OUTPUT\_RESULT\_XML.

- xsd\_json

This node contains the resulting schema-only validation report in JSON format and creates the output as defined by the variable OUTPUT\_RESULT\_JSON.

c. It utilizes the following variables:

i. Flow variables:

- VALIDATION\_TYPE

Default value is schema. The value extended is not supported at this time. This variable is currently not being used in any node.

- REPORT\_FORMAT

Default value is xml. It can be changed to json. This is used in Decision node XSD\_RESULT\_FORMAT.

- INPUT\_FILE

Default value is ../tools/mxservice/data/t2se\_head\_002.xml. This is the data file to be used for validation and can be customized. It is used in the Source node mx\_input.

- BIC\_FILE

Default value is ../data/bic.xml. This is the location of the bic cross-reference file that is used in validation maps. Since this flow is only doing schema validation, this variable is not used.

- CCY\_FILE

Default value is ..//data/currencycodedecimals.xml. This is the location of the country code cross-reference file that is used in validation maps. Since this flow is only doing schema validation, this variable is not used.

- **MXCONFIG\_FILE**  
Default value is ..//data/mxconfig.xml. This is the location of the file containing the rule validation settings that is used in validation maps. Since this flow is only doing schema validation, this variable is not used.
  - **OUTPUT\_RESULT\_XML**  
Default value is validation\_result.xml. This is the location of the validation report file in xml format. It is used in Target node xsd\_xml. It can be customized.
  - **OUTPUT\_RESULT\_JSON**  
Default value is validation\_result.json. This is the location of the validation report file in json format. It is used in Target node xsd\_json. It can be customized.
  - **FAILURE\_LOG**  
Default value is stopMXValidation.json. This is the location of the failure log. It is used in Log node FAILURE. This variable is not used in this flow.
  - **bizSvc**  
For internal use in the Map node MX pre-check to determine the type of data being read in the input file. This variable is not used in this flow.
  - **stopValidation**  
For internal use in the Map node MX pre-check and is checked in Decision node pre-valid chk to cause a Failure log in case a data file was not recognized as a valid T2S message. This variable is not used in this flow.
3. Open the main flow t2\_sec\_head002\_validation\_flow in the Design Server. It utilizes above one Source node, one Map node, one Decision node, one Log node, one Fail node, one Passthrough node, one Format Converter node, and two Target nodes.
4. In Design Server, create a package t2\_sec\_head002\_validation that contains the input files and the flow t2\_sec\_head002\_validation\_flow. The maps will automatically be included during deployment of the package onto the runtime server.  
Note: Not required for running flows on Design Server user interface directly.
- **[Run the example using three different methods](#)**  
You can run the example using three different methods:

## Run the example using three different methods

You can run the example using three different methods:

1. Running the example from the Design Server user interface.
2. Deploying the example to tx-rest and running it using the Swagger interface.
3. Deploying locally or to ftp server and using the flow command server process (flowcmdserver).

- **[Run the example from the Design Server user interface](#)**  
Use the following steps to run the example from the Design Server user interface:
- **[Run the example using the Swagger interface](#)**  
To run the example, deploy to tx-rest and run it using the Swagger interface.
- **[Run the example using the flow command server process \(flowcmdserver\)](#)**  
To run the example, deploy locally or to ftp server and use the flow command server process (flowcmdserver).

## Run the example from the Design Server user interface

Use the following steps to run the example from the Design Server user interface:

1. In the Design Server, open t2\_sec\_head002\_validation\_flow flow and click on Run button.
2. Set the toggle switch Run On Design Server and select flow input file. Example is t2se\_head\_002.xml.  
Note: If a default path is assigned to the variable INPUT\_FILE, the value defined for the variable will be used if an input file is not selected.  
Flow will run and report with the green check box. The report lists execution of all nodes.
3. Right click and select View Link Data on each link to examine the data.
4. View target node by clicking on the node, doing right click on the Input icon and selecting View Data.

## Run the example using the Swagger interface

To run the example, deploy to tx-rest and run it using the Swagger interface.

1. Create a Web type server definition where the runtime tx-rest is installed if you do not have a server definition to deploy in Design Server.
  2. If not running, start tx-rest on the server:  
`<DTX_HOME>/restapi/tomcat/dtxtomcat start tx-rest`
  3. Deploy the created package to the server definition in Design Server.
  4. Bring up the tx-rest Swagger UI:  
`<DTX_HOME>/restapi/tomcat/dtxtomcat showdocs tx-rest`
  5. In the Swagger Explore window, enter /tx-rest/v2/docs and click on the Explore button to view the deployed package.  
You can see a Miscellaneous section having two actions:
- **`PUT /v2/run/t2_sec_validation_flow`**

- **POST** /v2/run/t2\_sec\_head002\_validation\_flow
6. In the **PUT** action:
- Expand the **PUT** action and select the Try it out button.
  - Leave the input and output sections empty.
  - Under flow\_vars section, delete the entire content and replace with the commands to run the flow, for example:

```
{
 "INPUT_FILE": "C:/mydir/data/t2se_head_002.xml",
 "OUTPUT_RESULT_JSON": "C:/mydir/data/validation_result.json",
 "REPORT_FORMAT": "json"
}

or

{
 "INPUT_FILE": "C:/mydir/data/t2se_head_002.xml",
 "OUTPUT_RESULT_XML": "C:/mydir/data/validation_result.xml",
 "REPORT_FORMAT": "xml"
}
```

- d. Click Execute blue bar for running the flow.

When flow completes execution, 200 response code is shown in the Server Response section. This run should generate the output same as above.

7. To delete the deployed package, refresh the browser to get the Swagger and explore back to /tx-rest/openapi.json.

There will be a **DELETE** /v2/packages/{name} action.

8. Expand the **DELETE** /v2/packages/{name} action and select Try it out.

9. In the Name field, enter the name you gave to the created package.

10. Select the true from the stop query drop down, then select the blue Execute bar.

You can see a response of 204 with a timestamp, if it is successful.

## Run the example using the flow command server process (flowcmdserver)

To run the example, deploy locally or to ftp server and use the flow command server process (flowcmdserver).

- Create a server definition where the runtime is installed if you do not have a server local definition or ftp execution definition to deploy in Design Server.
- In Design Server, deploy the created package to the server definition. For example, deploy to c:\ftpserver\deployment directory.
- Execute the flow using the flow command server. The below command is for TARGET2 Securities t2se\_head\_002.xml message.

```
flowcmdserver.exe --dir c:\ftpserver\deployment\flows --flow t2_sec_head002_validation_flow.json --var
"INPUT_FILE=C:\ftpserver\deployment\tools\mx_service\data\t2se_head_002.xml" --audit
"C:\ftpserver\deployment\tools\mx_service\data\t2se_head_002_adt.json" -ad
```

The results will be in C:\ftpserver\deployment\flows as validation\_results.xml or validation\_results.json depending on the value of the REPORT\_FORMAT variable.

This command will execute with the variable VALIDATION\_TYPE value defined in the Flow settings in Design Server (only schema is supported).

To change, the following command can be used:

```
flowcmdserver.exe --dir c:\ftpserver\deployment\flows --flow t2_sec_head002_validation_flow.json --var
"INPUT_FILE=C:\ftpserver\deployment\tools\mx_service\data\t2se_head_002.xml" --var "REPORT_FORMAT=json" --audit
"C:\ftpserver\deployment\tools\mx_service\data\t2se_head_002_adt.json" -ad
```

4. The flow will report status similar to:

```
***Starting flow command server
Flow UUID: 85eabe2c-2302-4543-b347-9e18d65c6816
The flow audit file is: "C:\\\\ftpserver\\\\deployment\\\\tools\\\\mx_service\\\\data\\\\t2se_head_002_adt.json"
Flow completed successfully
Elapsed time: 3048ms
```

5. Examine the flow output result files validation\_results.xml and the audit file t2se\_head\_002\_adt.json.

## T2 Securities (T2S) schema validation (using maps) example

This example demonstrates the TARGET2 Securities (T2S) schema validation for T2S messages using maps only.

The maps can perform following level of validation:

- Schema validation
- [What the example contains](#)
- [Run the example in Design Studio](#)
- [Run the example in Design Server User Interface](#)

## What the example contains

This example contains the following directories and files:

- Maps:  
The maps directory contains the following map sources to use when running under Design Studio:

- t2\_sec\_mx\_schema\_validation\_xsd.mms  
Utility maps called by main map for all T2S messages schema validation.
  - t2\_sec\_schema\_validation\_frmwrk\_map\_xsd.mms  
The main map used to apply schema validation to T2S xml messages.
- Schemas:  
The schemas directory contains the following files:
- XML schemas based on SWIFT MyStandards Readiness T2S Portal:
    - admi.005.001.01.xsd
    - admi.006.001.01.xsd
    - admi.007.001.01.xsd
    - camt.003.001.07.xsd
    - camt.004.001.08.xsd
    - camt.005.001.08.xsd
    - camt.006.001.08.xsd
    - camt.009.001.07.xsd
    - camt.010.001.08.xsd
    - camt.019.001.07.xsd
    - camt.025.001.05.xsd
    - camt.050.001.05.xsd
    - camt.051.001.05.xsd
    - camt.052.001.08.xsd
    - camt.053.001.08.xsd
    - camt.054.001.08.xsd
    - camt.064.001.01.xsd
    - camt.065.001.01.xsd
    - camt.066.001.01.xsd
    - camt.067.001.01.xsd
    - camt.068.001.01.xsd
    - camt.069.001.03.xsd
    - camt.070.001.04.xsd
    - camt.072.001.01.xsd
    - camt.073.001.01.xsd
    - camt.074.001.01.xsd
    - camt.075.001.01.xsd
    - camt.078.001.01.xsd
    - camt.079.001.01.xsd
    - camt.080.001.01.xsd
    - camt.081.001.01.xsd
    - camt.082.001.01.xsd
    - camt.083.001.01.xsd
    - camt.084.001.01.xsd
    - camt.085.001.01.xsd
    - colr.001.001.01.xsd
    - colr.002.001.01.xsd
    - head.001.001.01.xsd
    - head.002.001.01.xsd
    - semt.002.001.10.xsd
    - semt.013.001.04.xsd
    - semt.014.001.06.xsd
    - semt.015.001.07.xsd
    - semt.016.001.07.xsd
    - semt.017.001.09.xsd
    - semt.018.001.10.xsd
    - semt.019.001.08.xsd
    - semt.020.001.05.xsd
    - semt.022.001.04.xsd
    - semt.025.001.01.xsd
    - semt.026.001.01.xsd
    - semt.027.001.01.xsd
    - semt.028.001.01.xsd
    - semt.029.001.01.xsd
    - semt.030.001.01.xsd
    - semt.031.001.01.xsd
    - semt.032.001.01.xsd
    - semt.033.001.01.xsd
    - semt.034.001.01.xsd
    - semt.040.001.01.xsd
    - semt.044.001.01.xsd
    - sese.020.001.06.xsd
    - sese.021.001.05.xsd
    - sese.022.001.05.xsd
    - sese.023.001.09.xsd
    - sese.024.001.10.xsd
    - sese.025.001.09.xsd
    - sese.027.001.05.xsd
    - sese.028.001.08.xsd
    - sese.029.001.04.xsd

- sese.030.001.08.xsd
- sese.031.001.08.xsd
- sese.032.001.09.xsd
- suppl.021.001.01.xsd

Note: The following schemas were modified to import schema suppl.021.001.01 to handle supplementary data:

- camt.067.001.01.xsd
- camt.068.001.01.xsd
- semt.014.001.06.xsd
- semt.015.001.07.xsd
- sese.024.001.10.xsd
- sese.025.001.09.xsd
- sese.032.001.09.xsd

- XML schemas based on W3C XML Signature Syntax and Processing version 1.2:

- xmldsig-core-schema.xsd

Note: The following schema was modified to import the signature schema xmldsig-core-schema.xsd:

- head.001.001.01.xsd

- Trees:

The trees directory contains the following files:

- swiftroute\_funds.mtt

Metadata that is used as an internal element placeholder.

- mxvalErrorReport.mtt

Metadata that represents the xml based structure of the validation report.

- Data:

The data directory contains the following file:

- Sample T2S valid files without header envelope for test purposes:

- admi\_007\_001\_01\_valid.xml

- Sample T2S valid files with header envelope for test purposes:

- bah\_admi\_005\_001\_01\_valid.xml
- bah\_admi\_006\_001\_01\_valid.xml
- bah\_admi\_007\_001\_01\_valid.xml
- bah\_camt\_003\_001\_07\_valid.xml
- bah\_camt\_004\_001\_08\_valid.xml
- bah\_camt\_005\_001\_08\_valid.xml
- bah\_camt\_006\_001\_08\_valid.xml
- bah\_camt\_009\_001\_07\_valid.xml
- bah\_camt\_010\_001\_08\_valid.xml
- bah\_camt\_019\_001\_07\_valid.xml
- bah\_camt\_025\_001\_05\_valid.xml
- bah\_camt\_050\_001\_05\_valid.xml
- bah\_camt\_051\_001\_05\_valid.xml
- bah\_camt\_052\_001\_08\_valid.xml
- bah\_camt\_053\_001\_08\_valid.xml
- bah\_camt\_054\_001\_08\_valid.xml
- bah\_camt\_064\_001\_01\_valid.xml
- bah\_camt\_065\_001\_01\_valid.xml
- bah\_camt\_066\_001\_01\_valid.xml
- bah\_camt\_067\_001\_01\_valid.xml
- bah\_camt\_068\_001\_01\_valid.xml
- bah\_camt\_069\_001\_03\_valid.xml
- bah\_camt\_070\_001\_04\_valid.xml
- bah\_camt\_072\_001\_01\_valid.xml
- bah\_camt\_073\_001\_01\_valid.xml
- bah\_camt\_074\_001\_01\_valid.xml
- bah\_camt\_075\_001\_01\_valid.xml
- bah\_camt\_078\_001\_01\_valid.xml
- bah\_camt\_079\_001\_01\_valid.xml
- bah\_camt\_080\_001\_01\_valid.xml
- bah\_camt\_081\_001\_01\_valid.xml
- bah\_camt\_082\_001\_01\_valid.xml
- bah\_camt\_083\_001\_01\_valid.xml
- bah\_camt\_084\_001\_01\_valid.xml
- bah\_camt\_085\_001\_01\_valid.xml
- bah\_colr\_001\_001\_01\_valid.xml
- bah\_colr\_002\_001\_01\_valid.xml
- bah\_head\_001\_001\_01\_valid.xml
- bah\_semt\_022\_001\_04\_valid.xml
- bah\_semt\_025\_001\_01\_valid.xml
- bah\_semt\_026\_001\_01\_valid.xml
- bah\_semt\_027\_001\_01\_valid.xml
- bah\_semt\_028\_001\_01\_valid.xml
- bah\_semt\_029\_001\_01\_valid.xml
- bah\_semt\_030\_001\_01\_valid.xml
- bah\_semt\_031\_001\_01\_valid.xml
- bah\_semt\_032\_001\_01\_valid.xml
- bah\_semt\_033\_001\_01\_valid.xml
- bah\_semt\_034\_001\_01\_valid.xml
- bah\_semt\_040\_001\_01\_valid.xml

- bah\_semt\_044\_001\_01\_valid.xml
  - bah\_sese\_020\_001\_06\_valid.xml
  - bah\_sese\_021\_001\_05\_valid.xml
  - bah\_sese\_022\_001\_05\_valid.xml
  - bah\_sese\_023\_001\_09\_valid.xml
  - bah\_sese\_024\_001\_10\_valid.xml
  - bah\_sese\_025\_001\_09\_valid.xml
  - bah\_sese\_027\_001\_05\_valid.xml
  - bah\_sese\_028\_001\_08\_valid.xml
  - bah\_sese\_029\_001\_04\_valid.xml
  - bah\_sese\_030\_001\_08\_valid.xml
  - bah\_sese\_031\_001\_08\_valid.xml
  - bah\_sese\_032\_001\_09\_valid.xml
  - bah\_supl\_021\_001\_01\_valid.xml
- Sample T2S valid files using header head.002.001.01 for test purposes:
    - head\_002\_001\_01\_valid.xml

---

## Run the example in Design Studio

1. Build all the maps in the .mms files listed in What the example contains section.
  2. Replace the input card 1 in router map, t2se9500\_val (under t2\_sec\_schema\_validation\_frmwrk\_map\_xsd.mms file) with desired input file.
  3. Run the main router map, t2se9500\_val.  
You will see the output file in the data folder called t2sec\_error\_report.xml.
- [Test with head.002.001.01 schema](#)  
For testing with head.002.001.01 schema, complete the following steps:

---

## Test with head.002.001.01 schema

For testing with head.002.001.01 schema, complete the following steps:

1. Replace input card 1 in router map, t2se9600\_val (under t2\_sec\_schema\_validation\_frmwrk\_map\_xsd.mms file) with desired input file.
2. Run the main router map, t2se9600\_val.  
You will see the output file in the data folder called t2pyld\_error\_report.xml.

---

## Run the example in Design Server User Interface

1. Open a browser pointing to the installation of the IBM Sterling Transformation Extender.
  2. Enter user and password credentials.
  3. If the project does not exist, import t2\_sec.zip.
  4. If absent, create a package containing all the Maps, Files, and Flows.
  5. Build package in desired server (to build all maps).
  6. Open the project.
  7. In the Maps tab, open the map t2se9500\_val.
  8. Modify the settings for input card 1 on design canvas to point to the desired test file.
  9. Save, build, and run the map t2se9500\_val.
  10. Right click on the output card on canvas and select View data.
- [Test with head.002.001.01 schema](#)  
For testing with head.002.001.01 schema, complete the following steps:

---

## Test with head.002.001.01 schema

For testing with head.002.001.01 schema, complete the following steps:

1. In the Maps tab, open the map t2se9600\_val.
2. Modify the settings for input card 1 on design canvas to point to the desired test file.
3. Save, build, and run the map t2se9600\_val.
4. Right click on the output card on canvas and select View data.

---

## SWIFT GPI/UC Examples:

- [SWIFT UC Examples:](#)  
SWIFT UC examples are as follow:
- [SWIFT GPI Examples:](#)  
SWIFT GPI examples are as follow:

## SWIFT UC Examples:

SWIFT UC examples are as follow:

- [\*\*SWIFT UC enhanced MX validation \(using Flow Server\) example\*\*](#)

This example demonstrates the SWIFT Universal Confirmation (UC) enhanced MX validation for SWIFT UC messages using the flow server.

- [\*\*SWIFT UC enhanced MX validation \(using maps\) example\*\*](#)

This example demonstrates the enhanced MX validation for SWIFT Universal Confirmation (UC) messages using maps.

- [\*\*SWIFT UC schema MX validation \(using maps\) example\*\*](#)

This example demonstrates the schema validation for SWIFT Universal Confirmation (UC) messages using maps only.

## SWIFT UC enhanced MX validation (using Flow Server) example

This example demonstrates the SWIFT Universal Confirmation (UC) enhanced MX validation for SWIFT UC messages using the flow server.

The flow can perform following level of validation:

- MX extended validation includes following checks:

- BIC lookup
- Country code lookup
- Currency code lookup
- IBAN format
- Allowed maximum fractional digits per currency
- Usage guideline rules, see mxconfig.xml for list of rules

- Schema validation

- [\*\*What the example contains\*\*](#)

Files included in this example are as follows:

- [\*\*How to run the example\*\*](#)

This SWIFT UC MX Validation will use the sample files to demonstrate the generation of validation report as output from a SWIFT UC XML message.

- [\*\*How to customize the mxconfig.xml file\*\*](#)

Rules enforced by the MX validation framework can be customized by editing the mxconfig.xml located under the validation/mx\_extended/data folder.

## What the example contains

Files included in this example are as follows:

- `swift_uc.zip`

- Test Data Files:

- `swuc_trck_001_uc.xml`
- `swuc_trck_001_uc_bad_rule.xml`
- `swuc_trck_001_uc_bad_schema.xml`

- Validation Files:

- `mxconfig.xml`
- `bic.xml`
- `currencycodedecimals.xml`

- Schemas:

- `bic.xsd`  
Metadata that represents the `bic.xml` repository file structure.

- `ccy.xsd`  
Metadata that represents the `currencycodedecimals.xml` repository file structure.

- `mxconfig.xsd`  
Metadata that represents the `mxconfig.xml` configuration file structure.

- `mxvalErrorReport.mtt`  
Metadata that represents the xml based structure of the validation report.

- `swiftroute_funds.mtt`  
Metadata that is used as an internal element placeholder.

- `trck.001.001.02_uc.xsd`
- `head.001.001.02_trck_001_uc.xsd`

Note: XML schemas were downloaded from SWIFT MyStandards Readiness SWIFT UC Portal.

- Maps:

- For Extended Validation:

- `swuc1000_val`
- `swuc1051_head112_trck001_uc`
- `swuc1001_trck_001_001_02_uc`

- For Schema Validation:

- `swuc1500_val`
- `swuc1551_head112_trck001_uc`
- `swuc1501_trck_001_001_02_uc`

- Common:

- mxut1007\_bizsvc\_swuc
- Flows:
  - swift\_uc\_validation\_flow

## How to run the example

This SWIFT UC MX Validation will use the sample files to demonstrate the generation of validation report as output from a SWIFT UC XML message.

The stopValidation.json file will report the validation failure due to the pre-conversion checks:

**If input file is other than any of the supported SWIFT UC messages as per schema list indicated above.**

1. Import the swift\_uc.zip project into the Design Server.
2. Open the swift\_uc project in Design Server and view the flow swift\_uc\_validation\_flow.
  - a. The Flow Description points to all the maps to be executed during the flow process, which will be called from within some of the map nodes in the flow. The following is a list of maps:
    - @packagemap=swift\_uc/validation/mx\_extended/maps/swift\_uc\_mx\_trck\_validation\_enh/swuc1051\_head112\_trck001\_uc
    - @packagemap=swift\_uc/validation/mx\_extended/maps/swift\_uc\_mx\_trck\_validation\_enh/swuc1001\_trck\_001\_001\_02\_uc
    - @packagemap=swift\_uc/validation/schema\_only/maps/swift\_uc\_mx\_schema\_validation\_xsd/swuc1551\_head112\_trck001\_uc
    - @packagemap=swift\_uc/validation/schema\_only/maps/swift\_uc\_mx\_schema\_validation\_xsd/swuc1501\_trck\_001\_001\_02\_uc
  - b. It utilizes the following nodes:
    - i. Source Nodes:
      - mx\_input  
This node identifies the input data to be validated in the flow. It uses the INPUT\_FILE variable to set the location of the data.
    - ii. Map Nodes:
      - MX pre-check  
This node runs map mxut1007\_bizsvc\_swuc which sets the flow variables, checks pre-validation conditions and creates infoset.json for validation.
      - EXT\_VAL  
This node runs map swuc1100\_val which calls the appropriate map to perform extended validation of a SWIFT UC message.
      - XSD\_VAL  
This node runs map swuc1500\_val which calls the appropriate map to perform schema validation of a SWIFT UC message.
    - Note: The maps swuc1100\_val and swuc1500\_val internally call all the other maps identified in the above Flow Description step.
    - iii. Decision Nodes:
      - pre-valid chk  
This node checks the flow variable stopValidation to decide if further validation/processing is not required.
      - EXT\_RESULT\_FORMAT  
This node checks the value of the flow variable REPORT\_FORMAT to determine if the resulting report for extended validation should be generated in XML (default) or JSON.
      - XSD\_RESULT\_FORMAT  
This node checks the value of the flow variable REPORT\_FORMAT to determine if the resulting report for schema-only validation should be generated in XML (default) or JSON.
    - iv. Route Nodes:
      - VAL\_TYPE  
This node checks the variable VALIDATION\_TYPE to determine if the process will do extended validation or schema-only validation on the data.
    - v. Log Nodes:
      - FAILURE  
This node creates a log file specified by the flow variable FAILURE\_LOG in case of no validation due to precondition check failures.
    - vi. Fail Node:
      - STOP  
It generates error message setup in the node in case of no validation performed.
    - vii. Passthrough Nodes
      - EXT\_XML\_CONVERT  
This node receives an extended validation report in XML format and does not modify the content, thus passing it as is to the appropriate target node.
      - XSD\_XML\_CONVERT  
This node receives a schema-only validation report in XML format and does not modify the content, thus passing it as is to the appropriate target node.
    - viii. Format Converter Nodes
      - EXT\_JSON\_CONVERT  
This node receives an extended validation report in XML format and converts it to JSON before passing it to the appropriate target node.
      - XSD\_JSON\_CONVERT  
This node receives a schema-only validation report in XML format and converts it to JSON before passing it to the appropriate target node.
    - ix. Target Nodes
      - ext\_xml  
This node contains the resulting extended validation report in XML format and creates the output as defined by variable OUTPUT\_RESULT\_XML.
      - ext\_json

This node contains the resulting extended validation report in JSON format and creates the output as defined by the variable OUTPUT\_RESULT\_JSON.

- xsd\_xml  
This node contains the resulting schema-only validation report in XML format and creates the output as defined by the variable OUTPUT\_RESULT\_XML.
- xsd\_json  
This node contains the resulting schema-only validation report in JSON format and creates the output as defined by the variable OUTPUT\_RESULT\_JSON.

c. It utilizes the following variables:

Flow variables:

- VALIDATION\_TYPE  
The default value is `extended`.  
  
For schema-only validation, it can be changed to `schema`. This is used in Route node VAL\_TYPE.
- REPORT\_FORMAT  
The default value is `xml`.  
  
It can be changed to `json`. This is used in Decision nodes EXT\_RESULT\_FORMAT and XSD\_RESULT\_FORMAT.
- INPUT\_FILE  
The default value is `../tools/mx_service/data/swuc_trck_001_uc.xml`.  
  
This is the data file to be used for validation and can be customized. This is used in Source node mx\_input.
- BIC\_FILE  
The default value is `../data/bic.xml`.  
  
This is the location of the bic cross-reference file that is used in all the maps executed by the map in node EXT\_VAL. It can be customized.
- CCY\_FILE  
The default value is `../data/currencycodedecimals.xml`.  
  
This is the location of the country code cross-reference file that is used in all the maps executed by the map in node EXT\_VAL. It can be customized.
- MXCONFIG\_FILE  
The default value is `../data/mxconfig.xml`.  
  
This is the location of the file containing the rule validation settings that is used in all the maps executed by the map in node EXT\_VAL. It can be customized.
- OUTPUT\_RESULT\_XML  
The default value is `validation_result.xml`.  
  
This is the location of the validation report file in xml format. It is used in Target nodes ext\_xml and xsd\_xml. It can be customized.
- OUTPUT\_RESULT\_JSON  
The default value is `validation_result.json`.  
  
This is the location of the validation report file in json format. It is used in Target nodes ext\_json and xsd\_json. It can be customized.
- FAILURE\_LOG  
The default value is `stopMXValidation.json`.  
  
This is the location of the failure log. It is used in Log node FAILURE. It can be customized.
- bizSvc  
This is for internal use in the Map node MX pre-check to determine the type of data being read in the input file.
- stopValidation  
This is for internal use in the Map node MX pre-check and is checked in Decision node pre-valid chk to cause a Failure log in case a data file was not recognized as a valid SWIFT UC message.

3. Open the main flow swift\_uc\_validation\_flow in the Design Server. It utilizes above one source node, three map nodes, one route node, three decision nodes, one log node, one fail node, two passthrough nodes, two format converter nodes and four target nodes.

4. In Design Server, create a package swift\_uc\_mx\_validation that contains the input files and one flow swift\_uc\_validation\_flow. The maps will automatically be included during deployment of the package onto the runtime server.

Note: Not required for running flows on Design Server user interface directly.

- **Run the example using three different methods**

You can run the example using three different methods:

---

## Run the example using three different methods

You can run the example using three different methods:

1. Running the example from the Design Server user interface.
2. Deploying the example to tx-rest and running it using the Swagger interface.
3. Deploying locally or to ftp server and using the flow command server process (flowcmdserver).

- Run the example from the Design Server user interface**  
Use the following steps to run the example from the Design Server user interface:
- Run the example using the swagger interface**  
To run the example, deploy to tx-rest and run it using the swagger interface.
- Run the example using the flow command server process (flowcmdserver)**  
To run the example, deploy locally or to ftp server and use the flow command server process (flowcmdserver).

## Run the example from the Design Server user interface

Use the following steps to run the example from the Design Server user interface:

- In the Design Server, open swift\_uc\_validation\_flow flow and click on Run button.
- Set toggle switch Run On Design Server and select flow input file. Example is swuc\_trck\_001\_uc.xml.  
Note: If a default path is assigned to the variable INPUT\_FILE, not selecting an input file will use the value defined for the variable.  
Flow will run and report with the green check box. The report list execution of all nodes.
- Right click and select View Link Data on each link to examine the data.
- Right click on the node and select View Log to see the logs.

## Run the example using the swagger interface

To run the example, deploy to tx-rest and run it using the swagger interface.

- Create a Web type server definition where the runtime tx-rest is installed if you do not have a server definition to deploy in Design Server.
- If not running, start tx-rest on the server: <DTX\_HOME>/restapi/tomcat/dtxtomcat start tx-rest.
- Deploy the created package to the server definition in Design Server.
- Bring up the tx-rest Swagger UI: <DTX\_HOME>/restapi/tomcat/dtxtomcat showdocs tx-rest.
- In the Swagger Explore window, enter /tx-rest/v2/docs and click on the Explore button to view the deployed package.  
You should see a Miscellaneous section having two actions: A **PUT** /v2/run/swift\_uc\_validation\_flow and a **POST** /v2/run/swift\_uc\_validation\_flow.
- In the **PUT** action:
  - Expand the **PUT** action and select the Try it out button.
  - Leave the input and output sections empty.
  - Under flow\_vars section, delete the entire content and replace with the commands to run the flow, for example:

```
{
 "INPUT_FILE": "C:/mydir/data/swuc_trck_001_uc.xml",
 "OUTPUT_RESULT_JSON": "C:/mydir/data/validation_result.json",
 "REPORT_FORMAT": "json",
 "VALIDATION_TYPE": "extended"
}

or

{
 "INPUT_FILE": "C:/mydir/data/swuc_trck_001_uc.xml",
 "OUTPUT_RESULT_XML": "C:/mydir/data/validation_result.xml",
 "REPORT_FORMAT": "xml",
 "VALIDATION_TYPE": "extended"
}
```

- Click Execute blue bar for running the flow. When flow completes execution, 200 response code is shown in the Server Response section. This run should generate the output same as above.
- Refresh the browser to delete the deployed package and get the Swagger Explore back to /tx-rest/openapi.json.  
There will be a DELETE /v2/packages/{name} action.
- Expand the DELETE /v2/packages/{name} action and select Try it out.
- In the Name field, enter the name you gave to the created package.
- Select the true option from the stop query drop down, select the blue Execute bar.  
You can see a response of 204 with a timestamp, if it is successful.

## Run the example using the flow command server process (flowcmdserver)

To run the example, deploy locally or to ftp server and use the flow command server process (flowcmdserver).

- Create a server definition where the runtime is installed, if you do not have a server local definition or ftp execution definition to deploy to, in Design Server.
- In Design Server, deploy the created package to the server definition.
- Execute the flow using the flow command server. The below command is for SWIFT UC swuc\_trck\_001\_uc.xml message.

```
flowcmdserver.exe --dir c:\ftpserver\deployment\flows --flow swift_uc_validation_flow.json
--var "INPUT_FILE=C:\ftpserver\deployment\tools\mx_service\data\swuc_trck_001_uc.xml"
--audit "C:\ftpserver\deployment\tools\mx_service\data\swuc_trck_001_uc_adt.json" -ad
```

- The results will be in C:\ftpserver\deployment\flows as validation\_results.xml or validation\_results.json depending on the value of the REPORT\_FORMAT variable.
- Use the below command to execute with the variable VALIDATION\_TYPE value defined in the Flow settings in Design Server (extended or schema).

```
flowcmdserver.exe --dir c:\ftpserver\deployment\flows
--flow swift_uc_validation_flow.json
--var "INPUT_FILE=C:\ftpserver\deployment\tools\mx_service\data\swuc_trck_001_uc.xml"
--var "VALIDATION_TYPE=schema"
```

```
--var "REPORT_FORMAT=json"
--audit "C:\ftpserver\deployment\tools\mx_service\data\swuc_trck_001_uc_adt.json" -ad

5. The flow will report status similar to:

***Starting flow command server

Flow UUID: 85eabe2c-2302-4543-b347-9e18d65c6816
The flow audit file is: "C:\\\\ftpserver\\\\deployment\\\\tools\\\\mx_service\\\\data\\\\swuc_trck_001_uc_adt.json"
Flow completed successfully
Elapsed time: 3048ms
```

6. Examine the flow output result file validation\_results.xml and the audit file swuc\_trck\_001\_uc\_adt.json.

## How to customize the mxconfig.xml file

Rules enforced by the MX validation framework can be customized by editing the mxconfig.xml located under the validation/mx\_extended/data folder.

All the rules enforced by the MX validation map are under element ExtendedValidation.

Any of the sub-elements under ExtendedValidation can be enabled by setting the value to T or disabled by setting to F.

### Example

To disable the BIC lookup validation feature.

### About this task

Change sub-element ExtendedValidation/BICValidation value from T to F.

### Procedure

1. Download the mxconfig.xml file from the Files tab.
2. Open the mxconfig.xml file and edit the sub-element ExtendedValidation/BICValidation value setting from T to F, then save the file.
3. Upload the edited mxconfig.xml file to the project.
4. Run the flow.  
Run using an input with BIC lookup validation issue.

Right click on the output card on canvas and select View data. No BIC lookup validation reported.

## SWIFT UC enhanced MX validation (using maps) example

This example demonstrates the enhanced MX validation for SWIFT Universal Confirmation (UC) messages using maps.

The maps can perform following level of validation:

- MX extended validation includes following checks:
  - BIC lookup
  - Country code lookup
  - Currency code lookup
  - IBAN format
  - Allowed maximum fractional digits per currency
  - Usage guideline rules, see mxconfig.xml for list of rules
- Schema validation
- [What the example contains](#)  
Files and directories included in this example are as follows:
- [Run the example in Design Studio](#)
- [Run the example in Design Server User Interface](#)
- [How to customize the mxconfig.xml file](#)

Rules enforced by the MX validation framework can be customized by editing the mxconfig.xml located under the validation/mx\_extended/data folder.

### What the example contains

Files and directories included in this example are as follows:

- Maps:  
The maps directory contains the following map sources to use when running under Design Studio:
  - swift\_uc\_mx\_trck\_validation\_enh.mms  
Utility maps called by main map for all the trck SWIFT UC xml message MX validation.
  - swift\_uc\_mx\_validation\_frmwrk\_map\_enh.mms  
The main map used to apply MX validation to SWIFT UC xml messages.

Note: When running under Design Server, the main framework map is swift\_uc\_mx\_validation\_frmwrk\_map\_flw\_enh.mms.

- Schemas:  
The schemas directory contains the following files:

- bic.xsd  
Metadata that represents the bic.xml repository file structure.
- ccy.xsd  
Metadata that represents the currencycodedecimals.xml repository file structure.
- mxconfig.xsd  
Metadata that represents the mxconfig.xml configuration file structure.
- trck.001.001.02\_uc.xsd
- head.001.001.02\_trck\_001\_uc.xsd

Note: XML schemas were downloaded from SWIFT MyStandards Readiness SWIFT UC Portal.

- Trees:  
The trees directory contains the following files:

- mxvalErrorReport.mtt  
Metadata that represents the xml based structure of the validation report.
- swiftroute\_funds.mtt  
Metadata that is used as an internal element placeholder.

- Data:  
The data directory contains the following file:

- bic.xml  
Repository file listing all BICs which are used during validation.
- currencycodedecimals.xml  
Repository file listing country codes, currency codes and corresponding maximum fractionally digits, used as reference for validation.
- mxconfig.xml  
Holds the MX configuration information on how to process the message.

Sample SWIFT UC valid files with header envelope for test purposes:

- bah\_trck\_001\_001\_02\_uc\_valid.xml

---

## Run the example in Design Studio

1. Build all the maps in the .mms files listed in What the example contains section.
2. Replace the input card 1 in router map, swuc1000\_val (under swift\_uc\_mx\_validation\_frmwrk\_map\_enh.mms or swift\_uc\_mx\_validation\_frmwrk\_map\_flw\_enh.mms file if running under Design Server) with the desired input file.
3. Run the main router map, swuc1000\_val.  
You will see the output file in the data folder called swuc\_error\_report.xml.

---

## Run the example in Design Server User Interface

1. Open a browser pointing to the installation of the IBM Sterling Transformation Extender.
2. Enter the user and password credentials.
3. If the project does not exist, import swift\_uc.zip.
4. If absent, create a package containing all the Maps, Files and Flows.
5. Build the package in the desired server (to build all maps).
6. Open the project.
7. In the Maps tab, open the map swuc1000\_val.
8. Modify the settings for input card 1 on design canvas to point to the desired test file.
9. Save, build, and run the map swuc1000\_val.
10. Right click on the output card on canvas and select View data.

---

## How to customize the mxconfig.xml file

Rules enforced by the MX validation framework can be customized by editing the mxconfig.xml located under the validation/mx\_extended/data folder.

All the rules enforced by the MX validation map are under element ExtendedValidation.

Any of the sub-elements under ExtendedValidation can be enabled by setting the value to T or disabled by setting to F.

---

## Example

To disable the BIC lookup validation feature.

## About this task

---

Change sub-element ExtendedValidation/BICValidation value from T to F.

## Procedure

---

1. Download the mxconfig.xml file from the Files tab.
2. Open the mxconfig.xml file and edit the sub-element ExtendedValidation/BICValidation value setting from T to F, then save the file.
3. Upload the edited mxconfig.xml file to the project.
4. Run the map.  
Run using an input with BIC lookup validation issue.

Right click on the output card on canvas and select View data. No BIC lookup validation reported.

---

## SWIFT UC schema MX validation (using maps) example

This example demonstrates the schema validation for SWIFT Universal Confirmation (UC) messages using maps only.

The maps can perform following level of validation:

- Schema validation
- [\*\*What the example contains\*\*](#)  
Files and directories included in this example are as follows:
- [\*\*Run the example in Design Studio\*\*](#)
- [\*\*Run the example in Design Server User Interface\*\*](#)

---

## What the example contains

Files and directories included in this example are as follows:

- Maps:  
The maps directory contains the following map sources:
  - swift\_uc\_mx\_schema\_validation\_xsd.mms  
Utility maps called by main map for all SWIFT UC xml messages schema validation.
  - swift\_uc\_schema\_validation\_frmwrk\_map\_xsd.mms  
The main map used to apply schema only validation to SWIFT UC xml messages.
- Schemas:  
The schemas directory contains the following files:
  - trck.001.001.02\_uc.xsd
  - head.001.001.02\_trck\_001\_uc.xsd
- Note: XML schemas were downloaded from SWIFT MyStandards Readiness SWIFT UC Portal.
- Trees:  
The trees directory contains the following files:
  - mxvalErrorReport.mtt  
Metadata that represents the xml based structure of the validation report.
  - swiftroute\_funds.mtt  
Metadata that is used as an internal element placeholder.
- Data:  
The data directory contains the following file:  
Sample SWIFT UC valid files with header envelope for test purposes:
  - bah\_trck\_001\_001\_02\_uc\_valid.xml

---

## Run the example in Design Studio

1. Build all the maps in the .mms files listed in What the example contains section.
2. Replace the input card 1 in router map, swuc1500\_val (under swift\_uc\_schema\_validation\_frmwrk\_map\_xsd.mms file) with the desired input file.
3. Run the main router map, swuc1500\_val.  
You will see the output file in the data folder called swuc\_error\_report.xml.

---

## Run the example in Design Server User Interface

1. Open a browser pointing to the installation of the IBM Sterling Transformation Extender.
  2. Enter the user and password credentials.
  3. If the project does not exist, import swift\_uc.zip.
  4. If absent, create a package containing all the Maps, Files and Flows.
  5. Build the package in the desired server (to build all maps).
  6. Open the project.
  7. In the Maps tab, open the map swuc1500\_val.
  8. Modify the settings for input card 1 on design canvas to point to the desired test file.
  9. Save, build, and run the map swuc1500\_val.
  10. Right click on the output card on canvas and select View data.
- 

## SWIFT GPI Examples:

SWIFT GPI examples are as follow:

- [\*\*SWIFT GPI enhanced MX validation \(using Flow Server\) example\*\*](#)

This example demonstrates the SWIFT GPI enhanced MX validation for SWIFT GPI messages using the flow server.

- [\*\*SWIFT GPI enhanced MX validation \(using maps\) example\*\*](#)

This example demonstrates the enhanced MX validation for SWIFT GPI messages using the maps.

- [\*\*SWIFT GPI schema MX validation \(using maps\) example\*\*](#)

This example demonstrates the schema validation for SWIFT gpi messages using the maps.

---

## SWIFT GPI enhanced MX validation (using Flow Server) example

This example demonstrates the SWIFT GPI enhanced MX validation for SWIFT GPI messages using the flow server.

The flow can perform following level of validation:

- MX extended validation includes following checks:

- BIC lookup
- Country code lookup
- Currency code lookup
- IBAN format
- Allowed maximum fractional digits per currency
- Usage guideline rules, see mxconfig.xml for list of rules

- Schema validation

- [\*\*What the example contains\*\*](#)

Files included in this example are as follows:

- [\*\*How to run the example\*\*](#)

This SWIFT GPI MX Validation will use the sample files to demonstrate the generation of validation report as output from a SWIFT GPI XML message.

- [\*\*How to customize the mxconfig.xml file\*\*](#)

Rules enforced by the MX validation framework can be customized by editing the mxconfig.xml located under the validation/mx\_extended/data folder.

---

## What the example contains

Files included in this example are as follows:

- [\*\*swift\\_gpi.zip\*\*](#)

- Test Data Files:

- swgp\_trck\_001\_gCCTINST.xml
- swgp\_trck\_001\_gCCTINST\_bad\_rule.xml
- swgp\_trck\_001\_gCCTINST\_bad\_schema.xml

- Validation Files:

- mxconfig.xml
- bic.xml
- currencycodedecimals.xml

- Schemas:

- bic.xsd  
Metadata that represents the bic.xml repository file structure.

- ccy.xsd

Metadata that represents the currencycodedecimals.xml repository file structure.

- mxconfig.xsd

Metadata that represents the mxconfig.xml configuration file structure.

- mxvalErrorReport.mtt

Metadata that represents the xml based structure of the validation report.

- swiftroute\_funds.mtt

Metadata that is used as an internal element placeholder.

- head.001.001.02\_trck\_001\_gCCT.xsd

- head.001.001.02\_trck\_001\_gCCTINST.xsd
- head.001.001.02\_trck\_001\_gCOV.xsd
- head.001.001.02\_trck\_001\_gFIT.xsd
- head.001.001.02\_trck\_001\_Swiftgo.xsd
- head.001.001.02\_trck\_002\_gCCT.xsd
- head.001.001.02\_trck\_002\_gCCTINST.xsd
- head.001.001.02\_trck\_002\_gCOV.xsd
- head.001.001.02\_trck\_002\_gFIT.xsd
- head.001.001.02\_trck\_002\_Swiftgo.xsd
- head.001.001.02\_trck\_003.xsd
- head.001.001.02\_trck\_004\_g4C.xsd
- head.001.001.02\_trck\_005.xsd
- trck.001.001.02\_gCCTINST.xsd
- trck.001.001.03\_gCCT.xsd
- trck.001.001.03\_gCOV.xsd
- trck.001.001.03\_gFIT.xsd
- trck.001.001.03\_Swiftgo.xsd
- trck.002.001.01\_gCCTINST.xsd
- trck.002.001.02\_gCCT.xsd
- trck.002.001.02\_gCOV.xsd
- trck.002.001.02\_gFIT.xsd
- trck.002.001.02\_Swiftgo.xsd
- trck.003.001.02.xsd
- trck.004.001.02\_g4C.xsd
- trck.005.001.02.xsd

Note: XML schemas were downloaded from SWIFT MyStandards Readiness SWIFT GPI Portal.

o Maps:

- For Extended Validation:
  - swgp1100\_val
  - swgp1051\_head112\_trck001\_gINST
  - swgp1052\_head112\_trck002\_gINST
  - swgp1053\_head112\_trck001\_gCCT
  - swgp1054\_head112\_trck002\_gCCT
  - swgp1055\_head112\_trck001\_gCOV
  - swgp1056\_head112\_trck002\_gCOV
  - swgp1057\_head112\_trck001\_gFIT
  - swgp1058\_head112\_trck002\_gFIT
  - swgp1059\_head112\_trck001\_Swiftgo
  - swgp1060\_head112\_trck002\_Swiftgo
  - swgp1061\_head112\_trck004\_g4C
  - swgp1062\_head112\_trck003
  - swgp1063\_head112\_trck005
  - swgp1001\_trck\_001\_001\_02\_gINST
  - swgp1002\_trck\_002\_001\_01\_gINST
  - swgp1003\_trck\_001\_001\_03\_gCCT
  - swgp1004\_trck\_002\_001\_02\_gCCT
  - swgp1005\_trck\_001\_001\_03\_gCOV
  - swgp1006\_trck\_002\_001\_02\_gCOV
  - swgp1007\_trck\_001\_001\_03\_gFIT
  - swgp1008\_trck\_002\_001\_02\_gFIT
  - swgp1009\_trck\_001\_001\_03\_Swiftgo
  - swgp1010\_trck\_002\_001\_02\_Swiftgo
  - swgp1011\_trck\_004\_001\_02\_g4C
  - swgp1012\_trck\_003\_001\_02
  - swgp1013\_trck\_005\_001\_02
- For Schema Validation:
  - swgp1500\_val
  - swgp1551\_head112\_trck001\_gINST
  - swgp1552\_head112\_trck002\_gINST
  - swgp1553\_head112\_trck001\_gCCT
  - swgp1554\_head112\_trck002\_gCCT
  - swgp1555\_head112\_trck001\_gCOV
  - swgp1556\_head112\_trck002\_gCOV
  - swgp1557\_head112\_trck001\_gFIT
  - swgp1558\_head112\_trck002\_gFIT
  - swgp1559\_head112\_trck001\_Swiftgo
  - swgp1560\_head112\_trck002\_Swiftgo
  - swgp1561\_head112\_trck004\_g4C
  - swgp1562\_head112\_trck003
  - swgp1563\_head112\_trck005
  - swgp1501\_trck\_001\_001\_03\_gINST
  - swgp1502\_trck\_002\_001\_02\_gINST
  - swgp1503\_trck\_001\_001\_03\_gCCT
  - swgp1504\_trck\_002\_001\_02\_gCCT
  - swgp1505\_trck\_001\_001\_03\_gCOV
  - swgp1506\_trck\_002\_001\_02\_gCOV
  - swgp1507\_trck\_001\_001\_03\_gFIT
  - swgp1508\_trck\_002\_001\_02\_gFIT
  - swgp1509\_trck\_001\_001\_03\_Swiftgo
  - swgp1510\_trck\_002\_001\_02\_Swiftgo

- swgp1511\_trck\_004\_001\_02\_g4C
- swgp1512\_trck\_003\_001\_02
- swgp1513\_trck\_005\_001\_02
- Common:
  - mxut1008\_bizsvc\_swgp
- Flows:
  - swift\_gpi\_validation\_flow

## How to run the example

This SWIFT GPI MX Validation will use the sample files to demonstrate the generation of validation report as output from a SWIFT GPI XML message.

The stopValidation.json file will report the validation failure due to the pre-conversion checks:

**If input file is other than any of the supported SWIFT GPI messages as per schema list indicated above.**

1. Import the swift\_gpi.zip project into the Design Server.

2. Open the swift\_gpi project in Design Server and view the flow swift\_gpi\_validation\_flow.

a. The Flow Description points to all the maps to be executed during the flow process, which will be called from within some of the map nodes in the flow. The following is a list of maps:

- @packagemap=swift\_gpi/validation/mx\_extended/maps/swift\_gpi\_mx\_trck\_validation\_enh/swgp1051\_head112\_trck001\_gINST
- @packagemap=swift\_gpi/validation/mx\_extended/maps/swift\_gpi\_mx\_trck\_validation\_enh/swgp1052\_head112\_trck002\_gINST
- @packagemap=swift\_gpi/validation/mx\_extended/maps/swift\_gpi\_mx\_trck\_validation\_enh/swgp1053\_head112\_trck001\_gCCT
- @packagemap=swift\_gpi/validation/mx\_extended/maps/swift\_gpi\_mx\_trck\_validation\_enh/swgp1054\_head112\_trck002\_gCCT
- @packagemap=swift\_gpi/validation/mx\_extended/maps/swift\_gpi\_mx\_trck\_validation\_enh/swgp1055\_head112\_trck001\_gCOV
- @packagemap=swift\_gpi/validation/mx\_extended/maps/swift\_gpi\_mx\_trck\_validation\_enh/swgp1056\_head112\_trck002\_gCOV
- @packagemap=swift\_gpi/validation/mx\_extended/maps/swift\_gpi\_mx\_trck\_validation\_enh/swgp1057\_head112\_trck001\_gFIT
- @packagemap=swift\_gpi/validation/mx\_extended/maps/swift\_gpi\_mx\_trck\_validation\_enh/swgp1058\_head112\_trck002\_gFIT
- @packagemap=swift\_gpi/validation/mx\_extended/maps/swift\_gpi\_mx\_trck\_validation\_enh/swgp1059\_head112\_trck001\_Swiftgo
- @packagemap=swift\_gpi/validation/mx\_extended/maps/swift\_gpi\_mx\_trck\_validation\_enh/swgp1060\_head112\_trck002\_Swiftgo
- @packagemap=swift\_gpi/validation/mx\_extended/maps/swift\_gpi\_mx\_trck\_validation\_enh/swgp1061\_head112\_trck004\_g4C
- @packagemap=swift\_gpi/validation/mx\_extended/maps/swift\_gpi\_mx\_trck\_validation\_enh/swgp1062\_head112\_trck003
- @packagemap=swift\_gpi/validation/mx\_extended/maps/swift\_gpi\_mx\_trck\_validation\_enh/swgp1063\_head112\_trck005
- @packagemap=swift\_gpi/validation/mx\_extended/maps/swift\_gpi\_mx\_trck\_validation\_enh/swgp1001\_trck\_001\_001\_02\_gINST
- @packagemap=swift\_gpi/validation/mx\_extended/maps/swift\_gpi\_mx\_trck\_validation\_enh/swgp1002\_trck\_002\_001\_01\_gINST
- @packagemap=swift\_gpi/validation/mx\_extended/maps/swift\_gpi\_mx\_trck\_validation\_enh/swgp1003\_trck\_001\_001\_03\_gCCT
- @packagemap=swift\_gpi/validation/mx\_extended/maps/swift\_gpi\_mx\_trck\_validation\_enh/swgp1004\_trck\_002\_001\_02\_gCCT
- @packagemap=swift\_gpi/validation/mx\_extended/maps/swift\_gpi\_mx\_trck\_validation\_enh/swgp1005\_trck\_001\_001\_03\_gCOV
- @packagemap=swift\_gpi/validation/mx\_extended/maps/swift\_gpi\_mx\_trck\_validation\_enh/swgp1006\_trck\_002\_001\_02\_gCOV
- @packagemap=swift\_gpi/validation/mx\_extended/maps/swift\_gpi\_mx\_trck\_validation\_enh/swgp1007\_trck\_001\_001\_03\_gFIT
- @packagemap=swift\_gpi/validation/mx\_extended/maps/swift\_gpi\_mx\_trck\_validation\_enh/swgp1008\_trck\_002\_001\_02\_gFIT
- @packagemap=swift\_gpi/validation/mx\_extended/maps/swift\_gpi\_mx\_trck\_validation\_enh/swgp1009\_trck\_001\_001\_03\_Swiftgo
- @packagemap=swift\_gpi/validation/mx\_extended/maps/swift\_gpi\_mx\_trck\_validation\_enh/swgp1010\_trck\_002\_001\_02\_Swiftgo
- @packagemap=swift\_gpi/validation/mx\_extended/maps/swift\_gpi\_mx\_trck\_validation\_enh/swgp1011\_trck\_004\_001\_02\_g4C
- @packagemap=swift\_gpi/validation/mx\_extended/maps/swift\_gpi\_mx\_trck\_validation\_enh/swgp1012\_trck\_003\_001\_02
- @packagemap=swift\_gpi/validation/mx\_extended/maps/swift\_gpi\_mx\_trck\_validation\_enh/swgp1013\_trck\_005\_001\_02
- @packagemap=swift\_gpi/validation/schema\_only/maps/swift\_gpi\_mx\_schema\_validation\_xsd/swgp1551\_head112\_trck001\_gINST
- @packagemap=swift\_gpi/validation/schema\_only/maps/swift\_gpi\_mx\_schema\_validation\_xsd/swgp1552\_head112\_trck002\_gINST
- @packagemap=swift\_gpi/validation/schema\_only/maps/swift\_gpi\_mx\_schema\_validation\_xsd/swgp1553\_head112\_trck001\_gCCT
- @packagemap=swift\_gpi/validation/schema\_only/maps/swift\_gpi\_mx\_schema\_validation\_xsd/swgp1554\_head112\_trck002\_gCCT
- @packagemap=swift\_gpi/validation/schema\_only/maps/swift\_gpi\_mx\_schema\_validation\_xsd/swgp1555\_head112\_trck001\_gCOV
- @packagemap=swift\_gpi/validation/schema\_only/maps/swift\_gpi\_mx\_schema\_validation\_xsd/swgp1556\_head112\_trck002\_gCOV
- @packagemap=swift\_gpi/validation/schema\_only/maps/swift\_gpi\_mx\_schema\_validation\_xsd/swgp1557\_head112\_trck001\_gFIT
- @packagemap=swift\_gpi/validation/schema\_only/maps/swift\_gpi\_mx\_schema\_validation\_xsd/swgp1558\_head112\_trck002\_gFIT
- @packagemap=swift\_gpi/validation/schema\_only/maps/swift\_gpi\_mx\_schema\_validation\_xsd/swgp1559\_head112\_trck001\_Swiftgo
- @packagemap=swift\_gpi/validation/schema\_only/maps/swift\_gpi\_mx\_schema\_validation\_xsd/swgp1560\_head112\_trck002\_Swiftgo
- @packagemap=swift\_gpi/validation/schema\_only/maps/swift\_gpi\_mx\_schema\_validation\_xsd/swgp1561\_head112\_trck004\_g4C
- @packagemap=swift\_gpi/validation/schema\_only/maps/swift\_gpi\_mx\_schema\_validation\_xsd/swgp1562\_head112\_trck003
- @packagemap=swift\_gpi/validation/schema\_only/maps/swift\_gpi\_mx\_schema\_validation\_xsd/swgp1563\_head112\_trck005
- @packagemap=swift\_gpi/validation/schema\_only/maps/swift\_gpi\_mx\_schema\_validation\_xsd/swgp1501\_trck\_001\_001\_02\_gINST
- @packagemap=swift\_gpi/validation/schema\_only/maps/swift\_gpi\_mx\_schema\_validation\_xsd/swgp1502\_trck\_002\_001\_01\_gINST
- @packagemap=swift\_gpi/validation/schema\_only/maps/swift\_gpi\_mx\_schema\_validation\_xsd/swgp1503\_trck\_001\_001\_03\_gCCT
- @packagemap=swift\_gpi/validation/schema\_only/maps/swift\_gpi\_mx\_schema\_validation\_xsd/swgp1504\_trck\_002\_001\_02\_gCCT
- @packagemap=swift\_gpi/validation/schema\_only/maps/swift\_gpi\_mx\_schema\_validation\_xsd/swgp1505\_trck\_001\_001\_03\_gCOV
- @packagemap=swift\_gpi/validation/schema\_only/maps/swift\_gpi\_mx\_schema\_validation\_xsd/swgp1506\_trck\_002\_001\_02\_gCOV
- @packagemap=swift\_gpi/validation/schema\_only/maps/swift\_gpi\_mx\_schema\_validation\_xsd/swgp1507\_trck\_001\_001\_03\_gFIT
- @packagemap=swift\_gpi/validation/schema\_only/maps/swift\_gpi\_mx\_schema\_validation\_xsd/swgp1508\_trck\_002\_001\_02\_gFIT
- @packagemap=swift\_gpi/validation/schema\_only/maps/swift\_gpi\_mx\_schema\_validation\_xsd/swgp1509\_trck\_001\_001\_03\_Swiftgo
- @packagemap=swift\_gpi/validation/schema\_only/maps/swift\_gpi\_mx\_schema\_validation\_xsd/swgp1510\_trck\_002\_001\_02\_Swiftgo
- @packagemap=swift\_gpi/validation/schema\_only/maps/swift\_gpi\_mx\_schema\_validation\_xsd/swgp1511\_trck\_004\_001\_02\_g4C
- @packagemap=swift\_gpi/validation/schema\_only/maps/swift\_gpi\_mx\_schema\_validation\_xsd/swgp1512\_trck\_003\_001\_02
- @packagemap=swift\_gpi/validation/schema\_only/maps/swift\_gpi\_mx\_schema\_validation\_xsd/swgp1513\_trck\_005\_001\_02

b. It utilizes the following nodes:

i. Source Nodes:

- mx\_input

This node identifies the input data to be validated in the flow. It uses the INPUT\_FILE variable to set the location of the data.

ii. Map Nodes:

- MX pre-check  
This node runs map mxut1008\_bizsvc\_swgp which sets the flow variables, checks pre-validation conditions and creates `infoset.json` for validation.

- EXT\_VAL  
This node runs map swgp1100\_val which calls the appropriate map to perform extended validation of a SWIFT gpi message.
- XSD\_VAL  
This node runs map swgp1500\_val which calls the appropriate map to perform schema validation of a SWIFT gpi message.

Note: The maps `swgp1100_val` and `swgp1500_val` internally call all the other maps identified in the above Flow Description step.

iii. Decision Nodes:

- pre-valid chk  
This node checks the flow variable `stopValidation` to decide if further validation/processing is not required.
- EXT\_RESULT\_FORMAT  
This node checks the value of the flow variable `REPORT_FORMAT` to determine if the resulting report for extended validation should be generated in XML (default) or JSON.
- XSD\_RESULT\_FORMAT  
This node checks the value of the flow variable `REPORT_FORMAT` to determine if the resulting report for schema-only validation should be generated in XML (default) or JSON.

iv. Route Nodes:

- VAL\_TYPE  
This node checks the variable `VALIDATION_TYPE` to determine if the process will do extended validation or schema-only validation on the data.

v. Log Nodes:

- FAILURE  
This node creates a log file specified by the flow variable `FAILURE_LOG` in case of no validation due to precondition check failures.

vi. Fail Node:

- STOP  
It generates error message setup in the node in case of no validation performed.

vii. Passthrough Nodes

- EXT\_XML\_CONVERT  
This node receives an extended validation report in XML format and does not modify the content, thus passing it as is to the appropriate target node.
- XSD\_XML\_CONVERT  
This node receives a schema-only validation report in XML format and does not modify the content, thus passing it as is to the appropriate target node.

viii. Format Converter Nodes

- EXT\_JSON\_CONVERT  
This node receives an extended validation report in XML format and converts it to JSON before passing it to the appropriate target node.
- XSD\_JSON\_CONVERT  
This node receives a schema-only validation report in XML format and converts it to JSON before passing it to the appropriate target node.

ix. Target Nodes

- ext\_xml  
This node contains the resulting extended validation report in XML format and creates the output as defined by variable `OUTPUT_RESULT_XML`.
- ext\_json  
This node contains the resulting extended validation report in JSON format and creates the output as defined by the variable `OUTPUT_RESULT_JSON`.
- xsd\_xml  
This node contains the resulting schema-only validation report in XML format and creates the output as defined by the variable `OUTPUT_RESULT_XML`.
- xsd\_json  
This node contains the resulting schema-only validation report in JSON format and creates the output as defined by the variable `OUTPUT_RESULT_JSON`.

c. It utilizes the following variables:

Flow variables:

- VALIDATION\_TYPE  
The default value is `extended`.  
For schema-only validation, it can be changed to `schema`. This is used in Route node `VAL_TYPE`.
- REPORT\_FORMAT  
The default value is `xml`.  
It can be changed to `json`. This is used in Decision nodes `EXT_RESULT_FORMAT` and `XSD_RESULT_FORMAT`.
- INPUT\_FILE  
The default value is `../tools/mx_service/data/swgp_trck_001_gCCTINST.xml`.  
This is the data file to be used for validation and can be customized. This is used in Source node `mx_input`.
- BIC\_FILE

The default value is `../data/bic.xml`.

This is the location of the bic cross-reference file that is used in all the maps executed by the map in node EXT\_VAL. It can be customized.

- CCY\_FILE

The default value is `../data/currencycodedecimals.xml`.

This is the location of the country code cross-reference file that is used in all the maps executed by the map in node EXT\_VAL. It can be customized.

- MXCONFIG\_FILE

The default value is `../data/mxconfig.xml`.

This is the location of the file containing the rule validation settings that is used in all the maps executed by the map in node EXT\_VAL. It can be customized.

- OUTPUT\_RESULT\_XML

The default value is `validation_result.xml`.

This is the location of the validation report file in xml format. It is used in Target nodes ext\_xml and xsd\_xml. It can be customized.

- OUTPUT\_RESULT\_JSON

The default value is `validation_result.json`.

This is the location of the validation report file in json format. It is used in Target nodes ext\_json and xsd\_json. It can be customized.

- FAILURE\_LOG

The default value is `stopMXValidation.json`.

This is the location of the failure log. It is used in Log node FAILURE. It can be customized.

- bizSvc

This is for internal use in the Map node MX pre-check to determine the type of data being read in the input file.

- stopValidation

This is for internal use in the Map node MX pre-check and is checked in Decision node pre-valid chk to cause a Failure log in case a data file was not recognized as a valid SWIFT gpi message.

3. Open the main flow `swift_gpi_validation_flow` in the Design Server. It utilizes above one source node, three map nodes, one route node, three decision nodes, one log node, one fail node, two passthrough nodes, two format converter nodes and four target nodes.

4. In Design Server, create a package `swift_gpi_mx_validation` that contains the input files and one flow `swift_gpi_validation_flow`. The maps will automatically be included during deployment of the package onto the runtime server.

Note: Not required for running flows on Design Server user interface directly.

- [Run the example using three different methods](#)

You can run the example using three different methods:

---

## Run the example using three different methods

You can run the example using three different methods:

1. Running the example from the Design Server user interface.
2. Deploying the example to tx-rest and running it using the Swagger interface.
3. Deploying locally or to ftp server and using the flow command server process (flowcmdserver).

- [Run the example from the Design Server user interface](#)

Use the following steps to run the example from the Design Server user interface:

- [Run the example using the Swagger interface](#)

To run the example, deploy to tx-rest and run it using the Swagger interface.

- [Run the example using the flow command server process \(flowcmdserver\)](#)

To run the example, deploy locally or to ftp server and use the flow command server process (flowcmdserver).

---

## Run the example from the Design Server user interface

Use the following steps to run the example from the Design Server user interface:

1. In the Design Server, open `swift_gpi_validation_flow` flow and click on Run button.
2. Set toggle switch Run On Design Server and select flow input file. Example is `swgp_trck_001_gCCTINST.xml`.  
Note: If a default path is assigned to the variable INPUT\_FILE, not selecting an input file will use the value defined for the variable.  
Flow will run and report with the green check box. The report lists execution of all nodes.
3. Right click and select View Link Data on each link to examine the data.
4. Right click on the node and select View Log to see the logs.

---

## Run the example using the Swagger interface

To run the example, deploy to tx-rest and run it using the Swagger interface.

- Create a Web type server definition where the runtime tx-rest is installed if you do not have a server definition to deploy in Design Server.
- If not running, start tx-rest on the server: <DTX\_HOME>/restapi/tomcat/dtxtomcat start tx-rest.
- Deploy the created package to the server definition in Design Server.
- Bring up the tx-rest Swagger UI: <DTX\_HOME>/restapi/tomcat/dtxtomcat showdocs tx-rest.
- In the Swagger Explore window, enter /tx-rest/v2/docs and click on the Explore button to view the deployed package.  
You should see a Miscellaneous section having two actions: A **PUT** /v2/run/swift\_gpi\_validation\_flow and a **POST** /v2/run/swift\_gpi\_validation\_flow.
- In the **PUT** action:
  - Expand the **PUT** action and select the Try it out button.
  - Leave the input and output sections empty.
  - Under flow\_vars section, delete the entire content and replace with the commands to run the flow, for example:

```
{
 "INPUT_FILE": "C:/mydir/data/swgp_trck_001_gCCTINST.xml",
 "OUTPUT_RESULT_JSON": "C:/mydir/data/validation_result.json",
 "REPORT_FORMAT": "json",
 "VALIDATION_TYPE": "extended"
}

or

{
 "INPUT_FILE": "C:/mydir/data/swgp_trck_001_gCCTINST.xml",
 "OUTPUT_RESULT_XML": "C:/mydir/data/validation_result.xml",
 "REPORT_FORMAT": "xml",
 "VALIDATION_TYPE": "extended"
}
```

- Click Execute blue bar for running the flow. When flow completes execution, 200 response code is shown in the Server Response section. This run should generate the output same as above.
- Refresh the browser to delete the deployed package and get the Swagger Explore back to /tx-rest/openapi.json. There will be a **DELETE** /v2/packages/{name} action.
- Expand the **DELETE** /v2/packages/{name} action and select Try it out.
- In the Name field, enter the name you gave to the created package.
- Select the true option from the stop query drop down, select the blue Execute bar.  
You can see a response of 204 with a timestamp, if it is successful.

## Run the example using the flow command server process (flowcmdserver)

To run the example, deploy locally or to ftp server and use the flow command server process (flowcmdserver).

- Create a server definition where the runtime is installed, if you do not have a server local definition or ftp execution definition to deploy to, in Design Server.
- In Design Server, deploy the created package to the server definition. For example, deploy to c:\ftpserver\deployment directory.
- Execute the flow using the flow command server. The below command is for SWIFT GPI swgp\_trck\_001\_gCCTINST.xml message.

```
flowcmdserver.exe --dir c:\ftpserver\deployment\flows --flow swift_gpi_validation_flow.json
--var "INPUT_FILE=C:\ftpserver\deployment\tools\mx_service\data\swgp_trck_001_gCCTINST.xml"
--audit "C:\ftpserver\deployment\tools\mx_service\data\swgp_trck_001_gCCTINST_adt.json" -ad
```

The results will be in C:\ftpserver\deployment\flows as validation\_results.xml or validation\_results.json depending on the value of the REPORT\_FORMAT variable.

- Use the below command to execute with the variable VALIDATION\_TYPE value defined in the Flow settings in Design Server ('extended' or 'schema').

```
flowcmdserver.exe --dir c:\ftpserver\deployment\flows
--flow swift_gpi_validation_flow.json
--var "INPUT_FILE=C:\ftpserver\deployment\tools\mx_service\data\swgp_trck_001_gCCTINST.xml"
--var "VALIDATION_TYPE=schema"
--var "REPORT_FORMAT=json"
--audit "C:\ftpserver\deployment\tools\mx_service\data\swgp_trck_001_gCCTINST_adt.json" -ad
```

- The flow will report status similar to:

```
***Starting flow command server

Flow UUID: 85eabe2c-2302-4543-b347-9e18d65c6816
The flow audit file is:
"C:\ftpserver\deployment\tools\mx_service\data\swgp_trck_001_gCCTINST_adt.json"
Flow completed successfully
Elapsed time: 3048ms
```

- Examine the flow output result file validation\_results.xml and the audit file swgp\_trck\_001\_gCCTINST\_adt.json.

## How to customize the mxconfig.xml file

Rules enforced by the MX validation framework can be customized by editing the mxconfig.xml located under the validation/mx\_extended/data folder.

All the rules enforced by the MX validation map are under element ExtendedValidation.

Any of the sub-elements under ExtendedValidation can be enabled by setting the value to T or disabled by setting to F.

## Example

To disable the BIC lookup validation feature.

## About this task

---

Change sub-element ExtendedValidation/BICValidation value from T to F.

## Procedure

---

1. Download the mxconfig.xml file from the Files tab.
2. Open the mxconfig.xml file and edit the sub-element ExtendedValidation/BICValidation value setting from T to F, then save the file.
3. Upload the edited mxconfig.xml file to the project.
4. Run the flow.  
Run using an input with BIC lookup validation issue.

Right click on the output card on canvas and select View data. No BIC lookup validation reported.

---

## SWIFT GPI enhanced MX validation (using maps) example

This example demonstrates the enhanced MX validation for SWIFT GPI messages using the maps.

The flow can perform following level of validation:

- MX extended validation includes following checks:
  - BIC lookup
  - Country code lookup
  - Currency code lookup
  - IBAN format
  - Allowed maximum fractional digits per currency
  - Usage guideline rules, see mxconfig.xml for list of rules
- Schema validation
- [\*\*What the example contains\*\*](#)  
Files and directories included in this example are as follows:
- [\*\*Run the example in Design Studio\*\*](#)
- [\*\*Run the example in Design Server User Interface\*\*](#)
- [\*\*How to customize the mxconfig.xml file\*\*](#)

Rules enforced by the MX validation framework can be customized by editing the mxconfig.xml located under the validation/mx\_extended/data folder.

---

## What the example contains

Files and directories included in this example are as follows:

- Maps:  
The maps directory contains the following map sources:
  - swift\_gpi\_mx\_trck\_validation\_enh.mms  
Utility maps called by main map for all the trck SWIFT GPI xml message MX validation.
  - swift\_gpi\_mx\_validation\_frmwrk\_map\_enh.mms  
The main map used to apply MX validation to SWIFT GPI xml messages.

Note: When running under Design Server, the main framework map is swift\_gpi\_mx\_validation\_frmwrk\_map\_flw\_enh.mms.

- Schemas:  
The schemas directory contains the following files:
  - bic.xsd  
Metadata that represents the bic.xml repository file structure.
  - ccy.xsd  
Metadata that represents the currencycodedecimals.xml repository file structure.
  - mxconfig.xsd  
Metadata that represents the mxconfig.xml configuration file structure.
  - head.001.001.02\_trck\_001\_gCCT.xsd
  - head.001.001.02\_trck\_001\_gCCTINST.xsd
  - head.001.001.02\_trck\_001\_gCOV.xsd
  - head.001.001.02\_trck\_001\_gFIT.xsd
  - head.001.001.02\_trck\_001\_Swiftgo.xsd
  - head.001.001.02\_trck\_002\_gCCT.xsd
  - head.001.001.02\_trck\_002\_gCCTINST.xsd
  - head.001.001.02\_trck\_002\_gCOV.xsd
  - head.001.001.02\_trck\_002\_gFIT.xsd
  - head.001.001.02\_trck\_002\_Swiftgo.xsd
  - head.001.001.02\_trck\_003.xsd
  - head.001.001.02\_trck\_004\_g4C.xsd
  - head.001.001.02\_trck\_005.xsd
  - trck.001.001.02\_gCCTINST.xsd

- trck.001.001.03\_gCCT.xsd
- trck.001.001.03\_gCOV.xsd
- trck.001.001.03\_gFIT.xsd
- trck.001.001.03\_Swiftgo.xsd
- trck.002.001.01\_gCCTINST.xsd
- trck.002.001.02\_gCCT.xsd
- trck.002.001.02\_gCOV.xsd
- trck.002.001.02\_gFIT.xsd
- trck.002.001.02\_Swiftgo.xsd
- trck.003.001.02.xsd
- trck.004.001.02\_g4C.xsd
- trck.005.001.02.xsd

Note: XML schemas were downloaded from SWIFT MyStandards Readiness SWIFT GPI Portal.

- Trees:

The trees directory contains the following files:

- mxvalErrorReport.mtt  
Metadata that represents the xml based structure of the validation report.
- swiftroute\_funds.mtt  
Metadata that is used as an internal element placeholder.

- Data:

The data directory contains the following files:

- bic.xml  
Repository file listing all BICs which are used during validation.
- currencycodedecimals.xml  
Repository file listing country codes, currency codes and corresponding maximum fractionally digits, used as reference for validation.
- mxconfig.xml  
Holds the MX configuration information on how to process the message.

Sample SWIFT GPI valid files for test purpose with header envelope:

- bah\_trck\_001\_001\_02\_gCCTINST\_valid.xml
- bah\_trck\_001\_001\_03\_gCCT\_valid.xml
- bah\_trck\_001\_001\_03\_gCOV\_valid.xml
- bah\_trck\_001\_001\_03\_gFIT\_valid.xml
- bah\_trck\_001\_001\_03\_Swiftgo\_valid.xml
- bah\_trck\_002\_001\_01\_gCCTINST\_valid.xml
- bah\_trck\_002\_001\_02\_gCCT\_valid.xml
- bah\_trck\_002\_001\_02\_gCOV\_valid.xml
- bah\_trck\_002\_001\_02\_gFIT\_valid.xml
- bah\_trck\_002\_001\_02\_Swiftgo\_valid.xml
- bah\_trck\_003\_001\_02\_valid.xml
- bah\_trck\_004\_001\_02\_g4C\_valid.xml
- bah\_trck\_005\_001\_02\_valid.xml

## Run the example in Design Studio

1. Build all the maps in the .mms files listed in What the example contains section.
2. Replace the input card 1 in router map, swgp1000\_val (under swift\_gpi\_mx\_validation\_frmwrk\_map\_enh.mms file or swift\_gpi\_mx\_validation\_frmwrk\_map\_flw\_enh.mms file if running under Design Server) with the desired input file.
3. Run the main router map, swgp1000\_val.  
You will see the output file in the data folder called swgp\_error\_report.xml.

## Run the example in Design Server User Interface

1. Open a browser pointing to the installation of the IBM Sterling Transformation Extender.
2. Enter the user and password credentials.
3. If the project does not exist, import swift\_gpi.zip.
4. If absent, create a package containing all the Maps, Files and Flows.
5. Build the package in the desired server (to build all maps).
6. Open the project.
7. In the Maps tab, open the map swgp1000\_val.
8. Modify the settings for input card 1 on design canvas to point to the desired test file.
9. Save, build, and run the map swgp1000\_val.
10. Right click on the output card on canvas and select View data.

## How to customize the mxconfig.xml file

Rules enforced by the MX validation framework can be customized by editing the mxconfig.xml located under the validation/mx\_extended/data folder.

All the rules enforced by the MX validation map are under element ExtendedValidation.

Any of the sub-elements under ExtendedValidation can be enabled by setting the value to T or disabled by setting to F.

## Example

---

To disable the BIC lookup validation feature.

## About this task

---

Change sub-element ExtendedValidation/BICValidation value from T to F.

## Procedure

---

1. Download the mxconfig.xml file from the Files tab.
2. Open the mxconfig.xml file and edit the sub-element ExtendedValidation/BICValidation value setting from T to F, then save the file.
3. Upload the edited mxconfig.xml file to the project.
4. Run the map.  
Run using an input with BIC lookup validation issue.

Right click on the output card on canvas and select View data. No BIC lookup validation reported.

## SWIFT GPI schema MX validation (using maps) example

---

This example demonstrates the schema validation for SWIFT gpi messages using the maps.

The maps can perform following level of validation:

- Schema validation
- [What the example contains](#)  
Files and directories included in this example are as follows:
- [Run the example in Design Studio](#)
- [Run the example in Design Server User Interface](#)

## What the example contains

---

Files and directories included in this example are as follows:

- Maps:  
The maps directory contains the following map sources to use when running under Design Studio:
  - swift\_gpi\_mx\_schema\_validation\_xsd.mms  
Utility maps called by main map for all SWIFT gpi xml messages schema validation.
  - swift\_gpi\_schema\_validation\_frmwrk\_map\_xsd.mms  
The main map used to apply schema validation to SWIFT GPI xml messages.
- Schemas:  
The schemas directory contains the following files:
  - head.001.001.02\_trck\_001\_gCCT.xsd
  - head.001.001.02\_trck\_001\_gCCTINST.xsd
  - head.001.001.02\_trck\_001\_gCOV.xsd
  - head.001.001.02\_trck\_001\_gFIT.xsd
  - head.001.001.02\_trck\_001\_Swiftgo.xsd
  - head.001.001.02\_trck\_002\_gCCT.xsd
  - head.001.001.02\_trck\_002\_gCCTINST.xsd
  - head.001.001.02\_trck\_002\_gCOV.xsd
  - head.001.001.02\_trck\_002\_gFIT.xsd
  - head.001.001.02\_trck\_002\_Swiftgo.xsd
  - head.001.001.02\_trck\_003.xsd
  - head.001.001.02\_trck\_004\_g4C.xsd
  - head.001.001.02\_trck\_005.xsd
  - trck.001.001.02\_gCCTINST.xsd
  - trck.001.001.03\_gCCT.xsd
  - trck.001.001.03\_gCOV.xsd
  - trck.001.001.03\_gFIT.xsd
  - trck.001.001.03\_Swiftgo.xsd
  - trck.002.001.01\_gCCTINST.xsd
  - trck.002.001.02\_gCCT.xsd
  - trck.002.001.02\_gCOV.xsd
  - trck.002.001.02\_gFIT.xsd
  - trck.002.001.02\_Swiftgo.xsd
  - trck.003.001.02.xsd

- trck.004.001.02\_g4C.xsd
- trck.005.001.02.xsd

Note: XML schemas were downloaded from SWIFT MyStandards Readiness SWIFT GPI Portal.

- Trees:  
The trees directory contains the following files:

- mxvalErrorReport.mtt  
Metadata that represents the xml based structure of the validation report.
- swiftroute\_funds.mtt  
Metadata that is used as an internal element placeholder.

- Data:  
The data directory contains the following file:

Sample SWIFT gpi valid files with header envelope for test purposes:

- bah\_trck\_001\_001\_02\_gCCTINST\_valid.xml
- bah\_trck\_001\_001\_03\_gCCT\_valid.xml
- bah\_trck\_001\_001\_03\_gCOV\_valid.xml
- bah\_trck\_001\_001\_03\_gFIT\_valid.xml
- bah\_trck\_001\_001\_03\_Swiftgo\_valid.xml
- bah\_trck\_002\_001\_01\_gCCTINST\_valid.xml
- bah\_trck\_002\_001\_02\_gCCT\_valid.xml
- bah\_trck\_002\_001\_02\_gCOV\_valid.xml
- bah\_trck\_002\_001\_02\_gFIT\_valid.xml
- bah\_trck\_002\_001\_02\_Swiftgo\_valid.xml
- bah\_trck\_003\_001\_02\_valid.xml
- bah\_trck\_004\_001\_02\_g4C\_valid.xml
- bah\_trck\_005\_001\_02\_valid.xml

## Run the example in Design Studio

1. Build all the maps in the .mms files listed in What the example contains section.
2. Replace the input card 1 in router map, swgp1500\_val (under swift\_gpi\_schema\_validation\_frmwrk\_map\_xsd.mms file) with the desired input file.
3. Run the main router map, swgp1500\_val.  
You will see the output file in the data folder called swgp\_error\_report.xml.

## Run the example in Design Server User Interface

1. Open a browser pointing to the installation of the IBM Sterling Transformation Extender.
2. Enter the user and password credentials.
3. If the project does not exist, import swift\_gpi.zip.
4. If absent, create a package containing all the Maps, Files and Flows.
5. Build package in the desired server (to build all maps).
6. Open the project.
7. In the Maps tab, open the map swgp1500\_val.
8. Modify the settings for input card 1 on design canvas to point to the desired test file.
9. Save, build, and run the map swgp1500\_val.
10. Right click on the output card on canvas and select View data.

## SIC (SIX Interbank Clearing) Validation Examples:

- [\*\*SIC \(SIX Interbank Clearing\) Schema validation \(using flows\) Example\*\*](#)  
This example demonstrates the SIC schema validation for SIC messages using flow server.
- [\*\*SIC \(SIX Interbank Clearing\) Schema validation \(using maps\) Example\*\*](#)  
This example demonstrates the schema validation for SIC messages using maps only.

## SIC (SIX Interbank Clearing) Schema validation (using flows) Example

This example demonstrates the SIC schema validation for SIC messages using flow server.

The flow can perform following level of validation:

- Schema validation
- [\*\*What the example contains\*\*](#)  
Files included in this example are as follows:
- [\*\*How to run the example\*\*](#)  
This SIC schema validation will use the sample files to demonstrate the generation of validation report as output from a SIC XML message.

## What the example contains

Files included in this example are as follows:

- sic\_rtgs.zip
  - Test Data Files:
    - sirt\_pacs\_009\_bad\_schema.xml
    - sirt\_pacs\_009.xml
  - Schemas:
    - mxvalErrorReport.mtt  
Metadata that represents the xml based structure of the validation report.
    - swiftroute\_funds.mtt  
Metadata that is used as an internal element placeholder.
  - Maps:
    - For Schema Validation:
      - sirt3500\_val
      - sirt3501\_acmt\_010\_001\_03
      - sirt3502\_acmt\_011\_001\_03
      - sirt3503\_acmt\_015\_001\_03
      - sirt3504\_camt\_003\_001\_07
      - sirt3505\_camt\_004\_001\_08
      - sirt3506\_camt\_005\_001\_08
      - sirt3507\_camt\_006\_001\_08
      - sirt3508\_camt\_007\_001\_08
      - sirt3509\_camt\_008\_001\_08
      - sirt3510\_camt\_011\_001\_07
      - sirt3511\_camt\_019\_001\_07
      - sirt3512\_camt\_025\_001\_05
      - sirt3513\_camt\_027\_001\_07
      - sirt3514\_camt\_029\_001\_09
      - sirt3515\_camt\_048\_001\_05
      - sirt3516\_camt\_050\_001\_05
      - sirt3517\_camt\_052\_001\_08
      - sirt3518\_camt\_054\_001\_08
      - sirt3519\_camt\_056\_001\_08
      - sirt3520\_camt\_087\_001\_06
      - sirt3521\_pacs\_002\_001\_10
      - sirt3522\_pacs\_004\_001\_09
      - sirt3523\_pacs\_008\_001\_08
      - sirt3524\_pacs\_009\_001\_08
      - sirt3525\_pacs\_028\_001\_03
      - sirt3526\_reda\_015\_001\_01
      - sirt3527\_reda\_017\_001\_01
    - Common:
      - mxut1010\_business\_svc\_sicrt
    - Flows:

- sic\_rtgs\_validation\_flow

## How to run the example

This SIC schema validation will use the sample files to demonstrate the generation of validation report as output from a SIC XML message.

The stopValidation.json file will report the validation failure due to the pre-conversion checks:

**If input file is other than any of the supported SIC messages as per schema list indicated above.**

1. Import the sic\_rtgs.zip project into the Design Server.
  2. Open the sic\_rtgs\_validation\_flow project in Design Server and view the sic\_rtgs\_validation\_flow.
- a. The Flow Description points to all the maps to be executed during the flow process, which will be called from within some of the map nodes in the flow. The following is a list of maps invoked using RUN built-in function during the flow process:

- @packagemap=sic\_rtgs/validation/schema\_only/maps/sic\_rtgs\_mx\_schema\_validation\_xsd/sirt3501\_acmt\_010\_001\_03
- @packagemap=sic\_rtgs/validation/schema\_only/maps/sic\_rtgs\_mx\_schema\_validation\_xsd/sirt3502\_acmt\_011\_001\_03
- @packagemap=sic\_rtgs/validation/schema\_only/maps/sic\_rtgs\_mx\_schema\_validation\_xsd/sirt3503\_acmt\_015\_001\_03
- @packagemap=sic\_rtgs/validation/schema\_only/maps/sic\_rtgs\_mx\_schema\_validation\_xsd/sirt3504\_camt\_003\_001\_07
- @packagemap=sic\_rtgs/validation/schema\_only/maps/sic\_rtgs\_mx\_schema\_validation\_xsd/sirt3505\_camt\_004\_001\_08
- @packagemap=sic\_rtgs/validation/schema\_only/maps/sic\_rtgs\_mx\_schema\_validation\_xsd/sirt3506\_camt\_005\_001\_08
- @packagemap=sic\_rtgs/validation/schema\_only/maps/sic\_rtgs\_mx\_schema\_validation\_xsd/sirt3507\_camt\_006\_001\_08
- @packagemap=sic\_rtgs/validation/schema\_only/maps/sic\_rtgs\_mx\_schema\_validation\_xsd/sirt3508\_camt\_007\_001\_08
- @packagemap=sic\_rtgs/validation/schema\_only/maps/sic\_rtgs\_mx\_schema\_validation\_xsd/sirt3509\_camt\_008\_001\_08
- @packagemap=sic\_rtgs/validation/schema\_only/maps/sic\_rtgs\_mx\_schema\_validation\_xsd/sirt3510\_camt\_011\_001\_07
- @packagemap=sic\_rtgs/validation/schema\_only/maps/sic\_rtgs\_mx\_schema\_validation\_xsd/sirt3511\_camt\_019\_001\_07
- @packagemap=sic\_rtgs/validation/schema\_only/maps/sic\_rtgs\_mx\_schema\_validation\_xsd/sirt3512\_camt\_025\_001\_05
- @packagemap=sic\_rtgs/validation/schema\_only/maps/sic\_rtgs\_mx\_schema\_validation\_xsd/sirt3513\_camt\_027\_001\_07
- @packagemap=sic\_rtgs/validation/schema\_only/maps/sic\_rtgs\_mx\_schema\_validation\_xsd/sirt3514\_camt\_029\_001\_09
- @packagemap=sic\_rtgs/validation/schema\_only/maps/sic\_rtgs\_mx\_schema\_validation\_xsd/sirt3515\_camt\_048\_001\_05
- @packagemap=sic\_rtgs/validation/schema\_only/maps/sic\_rtgs\_mx\_schema\_validation\_xsd/sirt3516\_camt\_050\_001\_05
- @packagemap=sic\_rtgs/validation/schema\_only/maps/sic\_rtgs\_mx\_schema\_validation\_xsd/sirt3517\_camt\_052\_001\_08
- @packagemap=sic\_rtgs/validation/schema\_only/maps/sic\_rtgs\_mx\_schema\_validation\_xsd/sirt3518\_camt\_054\_001\_08
- @packagemap=sic\_rtgs/validation/schema\_only/maps/sic\_rtgs\_mx\_schema\_validation\_xsd/sirt3519\_camt\_056\_001\_08
- @packagemap=sic\_rtgs/validation/schema\_only/maps/sic\_rtgs\_mx\_schema\_validation\_xsd/sirt3520\_camt\_087\_001\_06
- @packagemap=sic\_rtgs/validation/schema\_only/maps/sic\_rtgs\_mx\_schema\_validation\_xsd/sirt3521\_pacs\_002\_001\_10
- @packagemap=sic\_rtgs/validation/schema\_only/maps/sic\_rtgs\_mx\_schema\_validation\_xsd/sirt3522\_pacs\_004\_001\_09
- @packagemap=sic\_rtgs/validation/schema\_only/maps/sic\_rtgs\_mx\_schema\_validation\_xsd/sirt3523\_pacs\_008\_001\_08
- @packagemap=sic\_rtgs/validation/schema\_only/maps/sic\_rtgs\_mx\_schema\_validation\_xsd/sirt3524\_pacs\_009\_001\_08
- @packagemap=sic\_rtgs/validation/schema\_only/maps/sic\_rtgs\_mx\_schema\_validation\_xsd/sirt3525\_pacs\_028\_001\_03
- @packagemap=sic\_rtgs/validation/schema\_only/maps/sic\_rtgs\_mx\_schema\_validation\_xsd/sirt3526\_reda\_015\_001\_01
- @packagemap=sic\_rtgs/validation/schema\_only/maps/sic\_rtgs\_mx\_schema\_validation\_xsd/sirt3527\_reda\_017\_001\_01

- b. It utilizes the following nodes:

i. Source Nodes:

- mx\_input

This node identifies the input data to be validated in the flow. It uses the INPUT\_FILE variable to set the location of the data.

ii. Map Nodes:

- MX pre-check

Runs map mxut1010\_bizsvc\_sirt which sets flow variables, checks pre-validation conditions and creates infoset.json for validation.

- XSD\_VAL

Runs map sirt3500\_val which calls the appropriate map to perform schema validation of a SIC RTGS message.

Note: The map sirt3500\_val internally calls all the other maps identified in the above Flow Description step.

iii. Decision Nodes:

- pre-valid chk

This node checks the flow variable stopValidation to decide if further validation/processing is not required.

- XSD\_RESULT\_FORMAT

This node checks the value of the flow variable REPORT\_FORMAT to determine if the resulting report for schema-only validation should be generated in XML (default) or JSON.

iv. Log Nodes:

- FAILURE

In case of no validation due to precondition check failures, this node creates a log file specified by the flow variable FAILURE\_LOG.

v. Fail Node:

- STOP

It generates error message setup in the node in case of no validation performed.

vi. Passthrough Nodes:

- XSD\_XML\_CONVERT

This node receives a schema-only validation report in XML format and does not modify the content, thus passing it as is to the appropriate target node.

vii. Format Converter Nodes:

- XSD\_JSON\_CONVERT

This node receives a schema-only validation report in XML format and converts it to JSON before passing it to the appropriate target node.

viii. Target Nodes:

- xsd\_xml  
This node contains the resulting schema-only validation report in XML format and creates the output as defined by variable OUTPUT\_RESULT\_XML.
  - xsd\_json  
This node contains the resulting schema-only validation report in JSON format and creates the output as defined by variable OUTPUT\_RESULT\_JSON.
- c. It utilizes the following variables:
- Flow variables:
- VALIDATION\_TYPE  
The default value is schema.  
  
The value extended is not supported at this time. This variable is currently not being used in any node.
  - REPORT\_FORMAT  
The default value is xml.  
  
It can be changed to json. This is used in Decision node XSD\_RESULT\_FORMAT.
  - INPUT\_FILE  
The default value is ../tools/mxservice/data/sirt\_pacs\_009.xml.  
  
This is the data file to be used for validation and can be customized. This is used in Source node mx\_input.
  - OUTPUT\_RESULT\_XML  
The default value is validation\_result.xml.  
  
This is the location of the validation report file in xml format. It is used in Target node xsd\_xml. It can be customized.
  - OUTPUT\_RESULT\_JSON  
The default value is validation\_result.json.  
  
This is the location of the validation report file in json format. It is used in Target node xsd\_json. It can be customized.
  - FAILURE\_LOG  
The default value is stopMXValidation.json.  
  
This is the location of the failure log. It is used in Log node FAILURE. It can be customized.
  - bizSvc  
For internal use in the Map node MX pre-check to determine the type of data being read in the input file.
  - stopValidation  
For internal use in the Map node MX pre-check and is checked in Decision node pre-valid chk to cause a Failure log in case a data file was not recognized as a valid SIC message.
3. Open the main flow sic\_rtgs\_validation\_flow in the Design Server. It utilizes above one source node, two map nodes, two decision nodes, one log node, one fail node, one passthrough node, one format converter node and two target nodes
4. In Design Server, create a package sic\_rtgs\_mx\_validation that contains the input files and one flow sic\_rtgs\_validation\_flow. The maps will automatically be included during deployment of the package onto the runtime server.  
Note: Not required for running flows on Design Server user interface directly.
- [Run the example using three different methods](#)  
You can run the example using three different methods:

---

## Run the example using three different methods

You can run the example using three different methods:

1. Running the example from the Design Server user interface.
  2. Deploying the example to tx-rest and running it using the Swagger interface.
  3. Deploying locally or to ftp server and using the flow command server process (flowcmdserver).
- [Run the example from the Design Server user interface](#)  
Use the following steps to run the example from the Design Server user interface:
  - [Run the example using the Swagger interface](#)  
To run the example, deploy to tx-rest and run it using the Swagger interface.
  - [Run the example using the flow command server process \(flowcmdserver\)](#)  
To run the example, deploy locally or to ftp server and use the flow command server process (flowcmdserver).

---

## Run the example from the Design Server user interface

Use the following steps to run the example from the Design Server user interface:

1. In the Design Server, open sic\_rtgs\_validation\_flow and click on Run button.
2. Set toggle switch Run On Design Server and select flow input file. Example is sirt\_pacs\_009.xml.  
Note: If a default path is assigned to variable INPUT\_FILE, not selecting an input file will use the value defined for the variable.  
Flow will run and report with the green check box. The report lists the execution of all nodes.

3. To examine the data passed on each link, right click on each link and select View Link Data.
4. To view the target node, right click on the node input icon and select View Data.

## Run the example using the Swagger interface

To run the example, deploy to tx-rest and run it using the Swagger interface.

1. Create a Web type server definition where the runtime tx-rest is installed if you do not have a server definition to deploy in Design Server.
2. If not running, start tx-rest on the server: <DTX\_HOME>/restapi/tomcat/dtxtomcat start tx-rest.
3. Deploy the created package to the server definition in Design Server.
4. Bring up the tx-rest Swagger UI: <DTX\_HOME>/restapi/tomcat/dtxtomcat showdocs tx-rest.
5. In the Swagger Explore window, enter /tx-rest/v2/docs and click on the Explore button to view the deployed package.
- You should see a Miscellaneous section having two actions: A **PUT** /v2/run/sic\_rtgs\_validation\_flow and a **POST** /v2/run/sic\_rtgs\_validation\_flow.
6. In the **PUT** action:
  - a. Expand the **PUT** action and select the Try it out button.
  - b. Leave the input and output sections empty.
  - c. Under flow\_vars section, delete the entire content and replace it with the commands to run the flow. For example:

```
{
 "INPUT_FILE": "C:/mydir/data/sirt_pacs_009.xml",
 "OUTPUT_RESULT_JSON": "C:/mydir/data/validation_result.json",
 "REPORT_FORMAT": "json"
}

or

{
 "INPUT_FILE": "C:/mydir/data/sirt_pacs_009.xml",
 "OUTPUT_RESULT_XML": "C:/mydir/data/validation_result.xml",
 "REPORT_FORMAT": "xml"
}
```

- d. Click Execute blue bar for running the flow. When flow completes execution, 200 response code is shown in the Server Response section. This run should generate the output same as above.
7. Refresh the browser to delete the deployed package and get the Swagger Explore back to /tx-rest/openapi.json. There will be a **DELETE** /v2/packages/{name} action.
8. Expand the **DELETE** /v2/packages/{name} action and select Try it out.
9. In the Name field, enter the name you gave to the created package.
10. Select the true option from the stop query drop down, select the blue Execute bar.

You can see a response of 204 with a timestamp, if it is successful.

## Run the example using the flow command server process (flowcmdserver)

To run the example, deploy locally or to ftp server and use the flow command server process (flowcmdserver).

1. Create a server definition where the runtime is installed, if you do not have a server local definition or ftp execution definition to deploy to, in Design Server.
2. In Design Server, deploy the created package to the server definition. For example, deploy to c:\ftpserver\deployment directory.
3. Execute the flow using the flow command server. The below command is for SIC SIC sirt\_pacs\_009.xml message.

```
flowcmdserver.exe --dir c:\ftpserver\deployment\flows
--flow sic_rtgs_validation_flow.json
--var "INPUT_FILE=C:\ftpserver\deployment\tools\mx_service\data\sirt_pacs_009.xml"
--audit "C:\ftpserver\deployment\tools\mx_service\data\sirt_pacs_009_adt.json" -ad
```

- The results will be in C:\ftpserver\deployment\flows as validation\_result.xml or validation\_result.json depending on the value of the REPORT\_FORMAT variable.
4. Use the following command to execute with the variable VALIDATION\_TYPE value defined in the Flow settings in Design Server (extended or schema).

```
flowcmdserver.exe --dir c:\ftpserver\deployment\flows
--flow sic_rtgs_validation_flow.json
--var "INPUT_FILE=C:\ftpserver\deployment\tools\mx_service\data\sirt_pacs_009.xml"
--var "VALIDATION_TYPE=schema"
--var "REPORT_FORMAT=xml"
--audit "C:\ftpserver\deployment\tools\mx_service\data\sirt_pacs_009_adt.json" -ad
```

5. The flow will report status similar to:

```
***Starting flow command server

Flow UUID: 85eabe2c-2302-4543-b347-9e18d65c6816
The flow audit file is: "C:\ftpserver\deployment\tools\mx_service\data\sirt_pacs_009_adt.json"
Flow completed successfully
Elapsed time: 3048ms
```

6. Examine the flow output result file validation\_results.xml and the audit file sirt\_pacs\_009\_adt.json.

## SIC (SIX Interbank Clearing) Schema validation (using maps) Example

This example demonstrates the schema validation for SIC messages using maps only.

The maps can perform following level of validation:

- Schema validation
- [What the example contains](#)

Files and directories included in this example are as follows:

---

## What the example contains

Files and directories included in this example are as follows:

- Maps:  
The maps directory contains the following map sources:

- sic\_rtgs\_mx\_schema\_validation\_xsd.mms  
Contains maps for schema validation of any SIC message.
- sic\_rtgs\_schema\_validation\_frmwrk\_map\_xsd.mms  
The main map used to apply schema validation to SIC xml messages.

- Schemas:  
The schemas directory contains the following files:

[Based from Implementation Guidelines provided by SIX Interbank Clearing Ltd Platform Releases 4.11 (RTGS services SIC and euroSIC) and 5.1 (SIC IP service)]

- acmt.010.001.03.ch.01.xsd
- acmt.011.001.03.ch.01.xsd
- acmt.015.001.03.ch.01.xsd
- camt.003.001.07.ch.02.xsd
- camt.004.001.08.ch.02.xsd
- camt.005.001.08.ch.01.xsd
- camt.006.001.08.ch.02.xsd
- camt.007.001.08.ch.01.xsd
- camt.008.001.08.ch.01.xsd
- camt.011.001.07.ch.01.xsd
- camt.019.001.07.ch.02.xsd
- camt.025.001.05.ch.02.xsd
- camt.027.001.07.ch.01.xsd
- camt.029.001.09.ch.03.xsd
- camt.048.001.05.ch.01.xsd
- camt.050.001.05.ch.01.xsd
- camt.052.001.08.ch.02.xsd
- camt.054.001.08.ch.02.xsd
- camt.056.001.08.ch.04.xsd
- camt.087.001.06.ch.01.xsd
- pacs.002.001.10.ch.02.xsd
- pacs.004.001.09.ch.02.xsd
- pacs.008.001.08.ch.02.xsd
- pacs.009.001.08.ch.03.xsd
- pacs.028.001.03.ch.01.xsd
- reda.015.001.01.ch.01.xsd
- reda.017.001.01.ch.02.xsd

- Trees:  
The trees directory contains the following files:

- mxvalErrorReport.mtt  
Metadata that represents the xml based structure of the validation report.
- swiftroute\_funds.mtt  
Metadata that is used as an internal element placeholder.

- Data:  
The data directory contains the following files:

Sample SIC valid files for test purpose:

- acmt\_010\_001\_03\_valid.xml
- acmt\_011\_001\_03\_valid.xml
- acmt\_015\_001\_03\_valid.xml
- camt\_003\_001\_07\_valid.xml
- camt\_004\_001\_08\_valid.xml
- camt\_005\_001\_08\_valid.xml
- camt\_006\_001\_08\_valid.xml
- camt\_007\_001\_08\_valid.xml
- camt\_008\_001\_08\_valid.xml
- camt\_011\_001\_07\_valid.xml
- camt\_019\_001\_07\_valid.xml
- camt\_025\_001\_05\_valid.xml
- camt\_027\_001\_07\_valid.xml
- camt\_029\_001\_09\_valid.xml

- camt\_048\_001\_05\_valid.xml
- camt\_050\_001\_05\_valid.xml
- camt\_052\_001\_08\_valid.xml
- camt\_054\_001\_08\_valid.xml
- camt\_056\_001\_08\_valid.xml
- camt\_087\_001\_06\_valid.xml
- pacs\_002\_001\_10\_valid.xml
- pacs\_004\_001\_09\_valid.xml
- pacs\_008\_001\_08\_valid.xml
- pacs\_009\_001\_08\_valid.xml
- pacs\_028\_001\_03\_valid.xml
- reda\_015\_001\_01\_valid.xml
- reda\_017\_001\_01\_valid.xml

- [Run the example in Design Studio](#)
  - [Run the example in Design Server User Interface](#)
- 

## Run the example in Design Studio

1. Build all the maps in the .mms files listed in What the example contains section.
  2. Replace the input card 1 in router map, sirt3500\_val (under sic\_rtgs\_schema\_validation\_frmwrk\_map\_xsd.mms file) with desired input file.
  3. Run the main router map, sirt3500\_val.
- You will see the output file in the data folder called sirt\_error\_report.xml.
- 

## Run the example in Design Server User Interface

1. Open a browser pointing to the installation of the IBM Sterling Transformation Extender.
  2. Enter the user and password credentials.
  3. If the project does not exist, import sic\_rtgs.zip.
  4. If absent, create a package containing all the Maps, Files and Flows.
  5. Build the package in the desired server (to build all maps).
  6. Open the project.
  7. In the Maps tab, open the map sirt3500\_val.
  8. Modify the settings for input card 1 on design canvas to point to the desired test file.
  9. Save, build, and run the map sirt3500\_val.
  10. Right click on the output card on canvas and select View data.
- 

## Overview

There are three Pack for Healthcare components:

- HIPAA EDI component
- HL7 component
- NCPDP component

The Pack for Healthcare components provide healthcare and insurance-payer organizations with an infrastructure that can do the following:

- Enable compliance with government and industry mandates.
- Control administrative costs.
- Streamline business processes.
- Facilitate accuracy and timeliness of information.
- Offer a competitive advantage.
- Conform to existing systems.
- Adapt to new technologies as they emerge.
- Integrate multiple systems and standards.

The Pack for Healthcare is based on technology that allows your applications and systems to take advantage of the full range of integration capabilities including:

- Powerful any-to-any and many-to-many data transformation.
  - Multi-platform deployment and execution.
  - Management of transaction flows between trading partners.
  - Importers for interface creation from IBM DTDs, copybooks, DBMS catalogs, and other metadata.
  - Adapters for commercial applications, messaging middleware, Internet transports, relational databases, file systems, and utilities.
  - **[Installation location](#)**  
Installation details for the Pack for Healthcare are contained in the product release notes. See <https://www.ibm.com/support/pages/node/318755>.
  - **[General constraints](#)**  
All healthcare-related schemas that are developed by IBM adhere to the current, official implementation guides or published standards documents.
- 

## Installation location

Installation details for the Pack for Healthcare are contained in the product release notes. See <https://www.ibm.com/support/pages/node/318755>.

The Pack for Healthcare contains objects, including schemas, sample data, map source files, and compiled image files for utility modules.

These objects are installed within the imported project. The schema installs within the imported project.

## General constraints

All healthcare-related schemas that are developed by IBM adhere to the current, official implementation guides or published standards documents.

Healthcare data transformation maps are expected to be accurate implementations of published crosswalks or data transformation specifications. However, the examples provided do not account for your specific requirements. Therefore, all schemas and maps that are provided are intended for use as examples only.

Your responsibility as a user is to assess suitability and conduct appropriate tests before you place the Pack for Healthcare objects into a production environment. You must also ensure that you adequately address security and privacy considerations in the applications and systems you develop using the Pack for Healthcare.

## Healthcare industry

This documentation discusses technical terms and other information used to exchange data in the Healthcare industry.

- [Healthcare information exchange participants](#)
- [Healthcare transactions and standards](#)
- [Healthcare electronic data interchange initiatives](#)

The entire healthcare industry is facing increased pressure to implement productivity and quality improvements while reducing costs. The use of electronic data interchange and industry-specific data exchange standards for healthcare transaction data is a potential source of significant benefits in these areas.

## Healthcare information exchange participants

Every encounter between a patient and a healthcare provider involves exchanging information. In addition to the basics of patient demographics, symptoms, diagnoses, and treatments, the typical scenario requires the exchange of claims and payment data and associated payer, subscriber, eligibility, and authorization information. A single encounter can involve the transmission of large volumes of information among several participants.

## Healthcare transactions and standards

The healthcare information exchange can generally be viewed as a transaction between the sender and receiver participants.

Healthcare transactions include (but are not limited to):

- Healthcare claims or encounter
- Claim payment and remittance advice
- Healthcare claim status
- Coordination of benefits
- Eligibility for a health plan
- Referral certification and authorization
- Enrollment and un-enrollment in a health plan
- Premium payments
- Claim adjudication reporting

These transactions can be transmitted electronically in compliance with healthcare transaction standards.

Healthcare data exchange standards allow the accurate and timely exchange of information between healthcare organizations. For example, a simple benefits inquiry can take 20 minutes on the phone. By using Electronic Data Interchange (EDI), this type of request can be processed almost immediately, without the need for a call to the customer service center.

## Healthcare electronic data interchange initiatives

The entire healthcare industry is facing increased pressure to implement productivity and quality improvements while reducing costs. The use of electronic data interchange and industry-specific data exchange standards for healthcare transaction data is a potential source of significant benefits in these areas.

Healthcare industry electronic interchange initiatives include the following:

- HIPAA legislation
- HIPAA transactions
- WEDI SNIP transaction types
- Council for Affordable Quality Healthcare (CAQH)
  - Operating Rules
  - Companion Guide Template
- [HIPAA legislation](#)

- [HIPAA transactions](#)
- [WEDI SNIP transaction compliance types](#)

The HIPAA Compliance Checker supports the data validation and compliance reporting for Types 1 through 7. WEDI SNIP Type 1 through 7 validation 'rules' are included in the components of the product. The rules for the higher levels of validation are user-configurable.

## HIPAA legislation

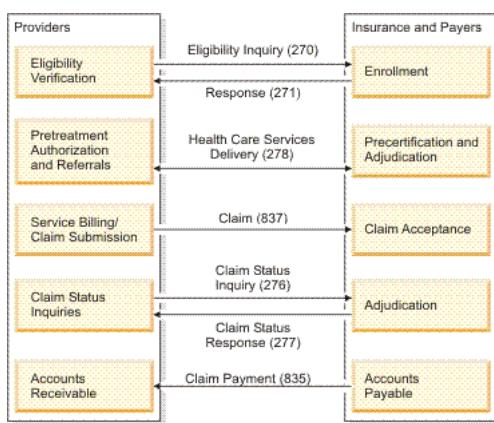
In 1996, legislation was passed to improve the overall healthcare administrative system. This legislation is known as the Health Insurance Portability and Accountability Act (HIPAA).

This pack addresses the administrative simplification aspects of HIPAA legislation, including the electronic standardization of patient health information, as well as administrative and financial data.

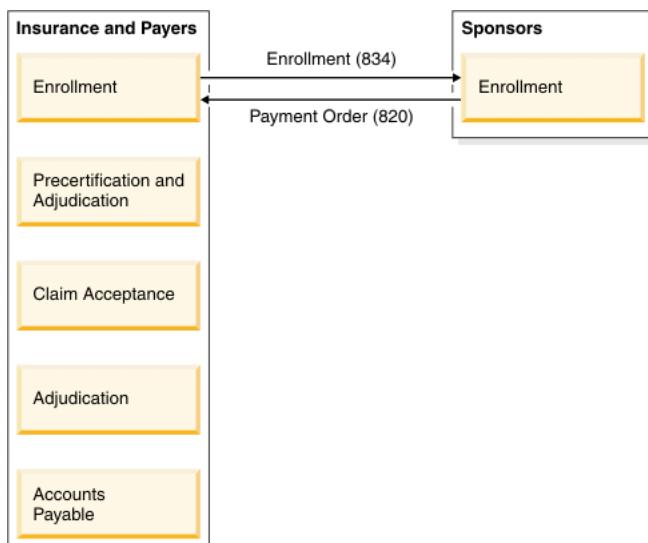
HIPAA regulations affect payers, health plans, clearinghouses, and those providers who conduct financial and administrative transactions electronically.

## HIPAA transactions

The following diagram shows the flow of information between healthcare providers, insurance, and payers and the associated HIPAA-mandated transaction numbers. These numbers refer to specific HIPAA X12 transaction sets supported in the Packs for Healthcare.



The following diagram shows the flow of information between insurance and payers and healthcare plan sponsors and the associated HIPAA-mandated transactions. Again, the parenthetical numbers refer to specific HIPAA X12 transaction sets.



## WEDI SNIP transaction compliance types

The HIPAA Compliance Checker supports the data validation and compliance reporting for Types 1 through 7. WEDI SNIP Type 1 through 7 validation 'rules' are included in the components of the product. The rules for the higher levels of validation are user-configurable.

Types 1 through 7 of HIPAA data transmission testing are defined as follows:

- Type 1 EDI Standard Integrity Testing
- Type 2 HIPAA Implementation Guide Requirement Testing
- Type 3 HIPAA Balance Testing
- Type 4 HIPAA Inter-Segment Situation Testing
- Type 5 HIPAA External Code Set Testing
- Type 6 HIPAA Line of Service Testing
- Type 7 Companion Guide Requirement Testing

## HIPAA EDI component

The documentation provided here describes the HIPAA EDI component.

Standards information and related data embedded in the [Licensee] Software, Outsourcing Services, and Combined Software Solutions are protected by copyright and other intellectual property of X12 Incorporated, a non-profit organization chartered by the American National Standards Institute to develop and maintain EDI standards and XML schemas. In order to provide and fund these activities in serving the American industry, X12 is willing to grant to all end users of the Licensee Software, Outsourcing Services, or Combined Software Solutions, a license at a royalty of \$180 per year. IBM strongly recommends that you contact X12 at [Licensing Partner: IBM](#) to obtain a license to use and access X12 metadata, represented in, for example, more than 30 versions of more than 300 transaction sets and implementation guides, through X12's online viewer, Glass.

In addition to the HIPAA EDI component, there are two other healthcare components:

- The HL7 component
- The NCPDP component
- **[Objects included in the HIPAA EDI component](#)**  
The data exchange transformation and integration requirements of the healthcare industry range from simple to complex and can vary greatly from one organization to another. The HIPAA EDI component contains predefined objects that give you the functionality and flexibility to develop a wide variety of applications and systems that meet your specific transaction and production requirements.
- **[HIPAA EDI schemas](#)**  
The Pack for Healthcare includes data transformation maps and maps for data validation. Data validation maps are either pass-through maps or related to the Compliance Check application.
- **[Maps](#)**  
The Pack for Healthcare includes data transformation maps and maps for data validation. Data validation maps are either pass-through maps or related to the Compliance Check application.
- **[Sample data](#)**  
Sample data is provided for use with the schemas and maps in the Pack for Healthcare.
- **[HIPAA Compliance Checking](#)**  
The Pack for Healthcare uses the types of HIPAA data transmission testing documented in the "Transaction Compliance and Certification" white paper developed by the workgroup for Electronic Data Interchange (WEDI) Strategic National Implementation Process (SNIP) Transactions Work Group - Testing Sub Work-Group; Dated June 11, 2002.
- **[Claim Level Rejection](#)**  
The Pack for Healthcare allows invalid claims to be rejected and valid claims to be processed. Validation is run on the document level (claim) for the 5010 837 Claims Transactions.
- **[Configurable Rules](#)**  
The Pack for Healthcare allows you to enable or disable WEDI/SNIP extended Level 2 through Level 7, as defined in the Compliance Check module.
- **[Type 5 Validation](#)**  
This support is only available with the HIPAA Data Compliance application.
- **[Type 6 Validation](#)**  
This support is only available with the HIPAA Data Compliance application.
- **[Type 7 Validation](#)**  
This support is only available with the HIPAA Data Compliance application.
- **[HIPAA EDI component examples](#)**  
This documentation describes the HIPAA EDI component examples.
- **[Performance tuning](#)**  
The HIPAA X12 data structures are hierarchical in nature, and the semantic and business rules can be complex. Some of the data files can be quite large. Given these factors, it's clear that performance is an important consideration when processing HIPAA data. There are various techniques that can be used to improve the performance of HIPAA data processing within Transformation Extender. These techniques involve separating, combining, validating, and transforming data. Some of these methods are based on the standard IBM performance tuning recommendations.
- **[Acknowledgment errors](#)**

## Objects included in the HIPAA EDI component

The data exchange transformation and integration requirements of the healthcare industry range from simple to complex and can vary greatly from one organization to another. The HIPAA EDI component contains predefined objects that give you the functionality and flexibility to develop a wide variety of applications and systems that meet your specific transaction and production requirements.

The HIPAA EDI component contains schemas, maps, sample data, and utility modules. These predefined, industry-specific objects are organized to provide flexibility to create and deploy a wide variety of applications and systems that address healthcare integration requirements. These objects are constructed to allow consistent behavior and results across all supported platforms and operating systems.

The schemas define the more commonly used healthcare industry data standards. The maps, data, and utility modules embrace typical data validation and transformation scenarios.

- **[Schemas](#)**
- **[Sample data](#)**

- [Maps](#)  
The following types of maps are included in this package:
  - [Utility modules](#)  
The utility modules are discussed in "Compliance Checking".
- 

## Schemas

The schemas in the Pack for Healthcare support the following 4050 and 5010 healthcare standards:

- HIPAA
  - PACDR (Post Adjudicated Claims Data Reporting)
  - HIX (Health Insurance Exchange)
  - CMS
  - 277CA
  - 4050 Claims Attachments
- 

## Sample data

Data files are included for the Pack for Healthcare. The data files fall into one of two general categories. Data files contain either transaction data in one or more of the data exchange formats associated with a particular healthcare industry standard; or application data used by the Compliance Check application for validating HIPAA X12 data.

The sample data files in the Pack for Healthcare conform to the following standards:

- HIPAA
  - CMS
  - HIX
  - PACDR
- 

## Maps

The following types of maps are included in this package:

- HIPAA X12 validation and reporting
  - Pass-through maps
  - Data transformation
- 

## Utility modules

The utility modules are discussed in "Compliance Checking".

---

## HIPAA EDI schemas

The documentation provided here discusses the schemas included with the Pack for HIPAA EDI.

The schemas are organized under the following headings:

- HIPAA X12 schemas
  - Utility schemas
  - CMS schemas
  - [Overview](#)
  - [What the schemas support](#)
  - [General schema information](#)  
Most of the schemas in the Pack for Healthcare include a category: %\_Type\_Tree\_Information.
  - [HIPAA X12 schemas](#)  
HIPAA X12 schemas describe the data exchange formats associated with HIPAA X12 transaction sets.
  - [ICD-10 support](#)  
The use of the International Classification of Disease, 10th Revision (ICD-10) in HIPAA has a compliance date of October 1, 2015. To be compliant, the schemas in the Pack for Healthcare have been modified to allow only ICD-10 codes.
- 

## Overview

The Pack for Healthcare includes schemas with data definitions for significant healthcare industry standards. The schema in these packages depicts the structure, relationships, and data element attributes as described in published specification documents. In practical terms, this means data that meets the specification requirements is considered valid, while data that does not meet the requirements is rejected. Other schemas attributes include the following:

- Compatibility with other IBM product offerings.
- Flexibility to modify or merge with other schemas.
- Ability to support reliable and efficient mapping.
- Easily maintainable.

The schemas are installed within the imported project.

There are two kinds of schemas in the packages: the schemas that define the data exchange formats associated with Healthcare industry data standards and the schemas that describe data formats used by application maps in the packages.

The schemas for healthcare application maps are described in "Compliance Checking" documentation.

## What the schemas support

The schemas included with the Pack for Healthcare support WEDI/SNIP Type 1, Type 2, Type 3, and Type 4, for HIPAA 5010 finalized transactions.

The schema variants that enforce the HIPAA Type 2, Type 3, and Type 4 tests are designed to work interchangeably (that is, "plug-and-play") in data transformation maps. Wherever possible, HIPAA X12 transaction set definitions and their corresponding addenda definitions are consistent with the intent to preserve the ability to merge and modify HIPAA X12 transaction set definitions and allow maximum flexibility.

The schemas do not produce acknowledgement responses. To generate acknowledgement responses, the Compliance Check application map must be used.

## General schema information

Most of the schemas in the Pack for Healthcare include a category: %\_Type\_Tree\_Information.

You can single-click on the %\_Description or %\_Revision\_History, then select **Restrictions**, to see a narrative description of the schema, or the schema revision history.

## HIPAA X12 schemas

HIPAA X12 schemas describe the data exchange formats associated with HIPAA X12 transaction sets.

The following schemas are included with the Pack for Healthcare:

- hipaa\_x12
- hipaa\_x12\_type\_1
- hipaa\_x12\_type\_2
- hipaa\_x12\_ruleless

The definitions of the HIPAA X12 schemas in the Pack for Healthcare are based on the HIPAA X12 Implementation Guide specification documents published by Washington Publishing Company.

For the Pack for Healthcare, published documents take precedence in cases where the table data and the standards documents do not agree.

The origin of the HIPAA X12 schemas in this package is from the X12 standard schema definitions in the Pack for Healthcare. There have been HIPAA-specific modifications to the X12 standard definitions in accordance with the HIPAA X12 Implementation Guides.

The HIPAA X12 schemas are described in this documentation.

- [hipaa\\_x12 schema](#)
- [Center for Medicare and Medicaid Services \(CMS\) schemas](#)
- [Utility schema](#)

## hipaa\_x12 schema

The hipaa\_x12 schema provides type 3 and 4 checking for 4050 and 5010 data and also provides type 2 and type 1 checking. The top-level group is HeaderInfo. This schema is used in x12t4precontentcheck.mmc for type 4 pass-through validation.

The hipaa\_x12 schema describes the HIPAA X12 transaction sets and enveloping structures associated with the following finalized HIPAA X12 transaction sets:

Transaction	Description	4050 version	5010 version	6020 version
270	Health Plan Eligibility Inquiry	-	005010X279A1	-
271	Health Plan Eligibility Response	-	005010X279A1	-
275	Additional Information to Support a Health Care Services Review	-	-	006020X316
276	Claim Status Request	-	005010X212	-
277	Claim Status Response	-	005010X212	-
278(N/A)	Review Notification and Acknowledgment	-	005010X216	-
278(Q/R)	Referral Request and Response	-	005010X217	006020X315
278(I/R)	Health Care Services Review - Inquiry and Response	-	005010X215	-
820	Health Plan Premium Payment	-	005010X218	-
834	Health Plan Enrollment	-	005010X220A1	-

<b>Transaction</b>	<b>Description</b>	<b>4050 version</b>	<b>5010 version</b>	<b>6020 version</b>
834	Plan Member Reporting	-	005010X318	-
835	Health Care Claim Payment	-	005010X221A1	-
837(I)	Health Care Claim and COB: Institutional	-	005010X223A2	-
837(D)	Health Care Claim and COB: Dental	-	005010X224A2	-
837(P)	Health Care Claim and COB: Professional	-	005010X222A1	-
837(I)	Post Adjudicated Claims Data Reporting Institutional	-	V5010X299A1	-
837(D)	Post Adjudicated Claims Data Reporting Dental	-	V5010X300A1	-
837(P)	Post Adjudicated Claims Data Reporting Professional	-	V5010X298A1	-
275	Additional Information to support a health care claim or Encounter	004050X151	005010X210	006020X314
277	Claim Request for Additional Information	004050X150	005010X213	006020X313
820	Health Insurance Exchange Related	-	005010X306	-
997	Functional Acknowledgment for Health Care Insurance	-	005010X230	-
999	Implementation Acknowledgment	-	005010X231A1	-
277CA	Health Care Claim Acknowledgment	-	005010X214	-
277DRA	Data Reporting Acknowledgment	-	005010X365	-

This schema enforces compliance of HIPAA X12 transmission data with the following WEDI/SNIP transaction compliance types:

- Type 1 - EDI Standard Integrity Testing
- Type 2 - HIPAA Implementation Guide Requirement Testing rules
- Type 3 - HIPAA balance testing
- Type 4 - HIPAA inter-segment situation testing

For a detailed description of Type 1, 2, 3 and 4 testing, see WEDI/SNIP HIPAA Transaction Compliance Types.

Partner X12 Inbound and Outbound Transmission EDI are the top-level definitions in this schema.

The hipaa\_x12 schema is utilized in the following source maps:

- hipaa\_x12\_pass\_through
- hipaa\_276\_5010\_to\_cms\_flat
- cms\_276\_5010\_flat\_to\_hipaa
- hipaa\_277\_5010\_to\_cms\_flat
- cms\_277\_5010\_flat\_to\_hipaa
- hipaa\_835\_5010\_to\_cms\_flat
- cms\_835\_5010\_flat\_to\_hipaa
- hipaa\_837i\_5010\_to\_cms\_flat
- cms\_837i\_5010\_flat\_to\_hipaa
- hipaa\_837p\_5010\_to\_cms\_flat
- cms\_837p\_5010\_flat\_to\_hipaa
- cms\_277ca\_5010\_flat\_to\_hipaa
- hipaa\_277ca\_5010\_to\_cms\_flat
- **hipaa\_x12\_type\_1\_schema**
- **hipaa\_x12\_type\_2\_schema**  
The hipaa\_x12\_type\_2 schema contains 5010 HIPAA X12 transaction standards.
- **hipaa\_x12\_ruleless\_schema**  
This hipaa\_x12\_ruleless schema contains the minimum number of component rules and restrictions required to properly identify the structure of a HIPAA X12 Transmission for the 4050 and 5010 HIPAA transaction sets.

## hipaa\_x12\_type\_1 schema

The hipaa\_x12\_type\_1 schema describes the HIPAA X12 transaction sets and enveloping structures associated with the finalized HIPAA X12 transaction standards. This schema enforces compliance of HIPAA X12 transmission data with WEDI/SNIP Type 1 EDI Standard Integrity Testing rules, including checks for:

- Valid segments
- Segment order
- Maximum segment and loop occurrences
- Element attributes (for example, size and data type)
- Adherence to X12 syntax rules as defined in the X12 Standard

Partner X12 Inbound and Outbound Transmission EDI are the top-level definitions of this schema. To prevent data from being rejected by the data transformation map, the hipaa\_x12\_type\_1 schema should be used as the input schema whenever the HIPAA X12 data has been validated against only the Type 1 - EDI Standard Integrity Testing Compliance checks.

## hipaa\_x12\_type\_2 schema

The hipaa\_x12\_type\_2 schema contains 5010 HIPAA X12 transaction standards.

This schema enforces compliance with HIPAA X12 transmission data for the following:

- Type 1 Standard Integrity Testing
- Type 2 HIPAA Implementation Guide Requirement Testing rules
- HIPAA Implementation Guide Requirements Testing. Partner X12 Inbound and Outbound Transmission EDI are the top-level definitions in this schema.

---

## hipaa\_x12\_ruleless schema

This hipaa\_x12\_ruleless schema contains the minimum number of component rules and restrictions required to properly identify the structure of a HIPAA X12 Transmission for the 4050 and 5010 HIPAA transaction sets.

The hipaa\_x12\_ruleless schema can be used for transformation when the data is known to be valid after it has been checked for compliance. Performance is therefore enhanced as the data does not need re-checking for validity.

---

## Center for Medicare and Medicaid Services (CMS) schemas

The CMS schemas describe the healthcare data interface formats as specified by the CMS. The following schemas are delivered with the Pack for Healthcare:

Note: The Center for Medicare and Medicaid Services (CMS) was formerly the Health Care Financing Administration (HCFA).

- cms\_276\_277\_5010\_flat
- cms\_277ca\_5010\_flat
- cms\_835\_5010a1\_flat
- cms\_837i\_5010a2\_flat
- cms\_837p\_5010a1\_flat

- **[cms\\_276\\_277\\_5010\\_flat schema](#)**

The cms\_276\_277\_5010\_flat schema defines the contents of the CMS (formerly HCFA) Part A and Part B 276 and 277 Health Care Claim Status Request and Response Flat files that correspond to the HIPAA 5010X212 formats.

- **[cms\\_277ca\\_5010\\_flat schema](#)**

The cms\_277ca\_5010\_flat schema defines the content of the CMS (formerly HCFA) and 277ca Health Care Claim Acknowledgment Flat files that correspond to the HIPAA 5010X214 formats.

- **[cms\\_835\\_5010a1\\_flat schema](#)**

The cms\_835\_5010a1\_flat schema defines the contents of the CMS (formerly HCFA) Part A and Part B 835 Health Care Claim Payment Flat file that corresponds to the HIPAA 5010X221 Addenda format.

- **[cms\\_837i\\_5010a2\\_flat schema](#)**

The cms\_837i\_5010a2\_flat schema defines the contents of the CMS (formerly HCFA) Part A 837 Institutional Claim Flat file that corresponds to the HIPAA 5010X223 Addenda format.

- **[cms\\_837p\\_5010a1\\_flat schema](#)**

The cms\_837p\_5010a1\_flat schema defines the contents of the CMS (formerly HCFA) Part B 837 Professional Claim Flat file that corresponds to the HIPAA 5010X222 Addenda format.

---

## cms\_276\_277\_5010\_flat schema

The cms\_276\_277\_5010\_flat schema defines the contents of the CMS (formerly HCFA) Part A and Part B 276 and 277 Health Care Claim Status Request and Response Flat files that correspond to the HIPAA 5010X212 formats.

The cms\_276\_277\_5010\_flat schema is used in the following source maps:

- cms\_276\_277\_5010\_flat\_pass\_through
- cms\_277\_5010\_flat\_pass\_through
- hipaa\_276\_5010\_to\_cms\_flat
- hipaa\_277\_5010\_to\_cms\_flat
- cms\_276\_277\_5010\_flat\_to\_hipaa
- cms\_277\_5010\_flat\_to\_hipaa

---

## cms\_277ca\_5010\_flat schema

The cms\_277ca\_5010\_flat schema defines the content of the CMS (formerly HCFA) and 277ca Health Care Claim Acknowledgment Flat files that correspond to the HIPAA 5010X214 formats.

The cms\_277ca\_5010\_flat schema is used in the following source maps:

- cms\_277ca\_5010\_flat\_pass\_through
- cms\_277ca\_5010\_flat\_to\_hipaa
- hipaa\_277ca\_5010\_to\_cms\_flat

---

## cms\_835\_5010a1\_flat schema

The cms\_835\_5010a1\_flat schema defines the contents of the CMS (formerly HCFA) Part A and Part B 835 Health Care Claim Payment Flat file that corresponds to the HIPAA 5010X221 Addenda format.

The cms\_835\_5010a1\_flat schema is used in the following source maps:

- cms\_835\_5010\_flat\_pass\_through

- cms\_835\_5010\_flat\_to\_hipaa
- hipaa\_835\_5010\_to\_cms

---

## cms\_837i\_5010a2\_flat schema

The cms\_837i\_5010a2\_flat schema defines the contents of the CMS (formerly HCFA) Part A 837 Institutional Claim Flat file that corresponds to the HIPAA 5010X223 Addenda format.

The cms\_837i\_5010a2\_flat schema is used in the following source maps:

- cms\_837i\_flat\_5010\_pass\_through
- cms\_837i\_5010\_flat\_to\_hipaa
- hipaa\_837i\_5010\_to\_cms\_flat

---

## cms\_837p\_5010a1\_flat schema

The cms\_837p\_5010a1\_flat schema defines the contents of the CMS (formerly HCFA) Part B 837 Professional Claim Flat file that corresponds to the HIPAA 5010X222 Addenda format.

The cms\_837p\_5010a1\_flat schema is used in the following source maps:

- cms\_837p\_5010\_flat\_pass\_through
- cms\_837p\_5010\_flat\_to\_hipaa
- hipaa\_837p\_5010\_to\_cms\_flat

---

## Utility schema

The hipaa\_x12\_v5010x214\_277ca utility schema is included with the Pack for Healthcare.

- [hipaa\\_x12\\_v5010x214\\_277ca schema](#)

The hipaa\_x12\_v5010x214\_277ca schema contains the definition of the X12N 005010X214 format for the Health Care Claim Acknowledgement (277CA). This schema can be used for input validation, transformation, or output generation of the 277 Claims Acknowledgment.

---

## hipaa\_x12\_v5010x214\_277ca schema

The hipaa\_x12\_v5010x214\_277ca schema contains the definition of the X12N 005010X214 format for the Health Care Claim Acknowledgement (277CA). This schema can be used for input validation, transformation, or output generation of the 277 Claims Acknowledgment.

The hipaa\_x12\_v5010x214\_277ca schema remains in the Pack. However, the structure of the 005010X214 277CA is now included in the following schemas:

- hipaa\_x12
- hipaa\_x12\_type\_2
- hipaa\_x12\_ruleless

The following source map files use the hipaa\_x12 schemas:

- hipaa\_277ca\_to\_cms\_flat
- cms\_277ca\_5010\_flat\_pass\_through
- cms\_277ca\_5010\_flat\_to\_hipaa

---

## ICD-10 support

The use of the International Classification of Disease, 10th Revision (ICD-10) in HIPAA has a compliance date of October 1, 2015. To be compliant, the schemas in the Pack for Healthcare have been modified to allow only ICD-10 codes.

To assist with the transition to ICD-10, the following set of schemas are provided in the pack to allow both the ICD-9 and ICD-10 qualifiers.

- hipaa\_x12\_type\_2\_icd9\_icd10
- hipaa\_x12\_icd9\_icd10
- hipaa\_x12\_ruleless\_icd9\_icd10
- hipaa\_x12\_segment\_icd9\_icd10

If these schemas are required, the following steps must be followed to replace the schemas in the compliance check application.

- [Replacing the ICD-10 compliant schemas](#)

If both the ICD-9 and ICD-10 codes are required, you must replace the schemas in the compliance\_check source map and the pass-through maps.

---

## Replacing the ICD-10 compliant schemas

If both the ICD-9 and ICD-10 codes are required, you must replace the schemas in the compliance\_check source map and the pass-through maps.

- [Changing maps impacted in the compliance\\_check source map](#)  
If both the ICD-9 and ICD-10 codes are required, the schemas in the compliance\_check source map file must be replaced.
- [Changing maps impacted in the hipaa\\_x12\\_pass\\_through source map](#)  
If both the ICD-9 and ICD-10 codes are required, the schemas in the hipaa\_x12\_pass\_through source map must be replaced.
- [Changing other maps](#)  
This documentation describes changes to other maps in the Pack for Healthcare.

## Changing maps impacted in the compliance\_check source map

If both the ICD-9 and ICD-10 codes are required, the schemas in the compliance\_check source map file must be replaced.

### About this task

Follow the procedure to replace the default schema with the appropriate ICD-9 and ICD-10 schema.

Maps impacted	Input card number and name	Default type schema	ICD-9 and ICD-10 type schema
x12t2precontentcheck	#1 X12_Func_Group	hipaa_x12_type_2	hipaa_x12_type_2_icd9_icd10
x12t4precontentcheck	#1 X12_Func_Group	hipaa_x12	hipaa_x12_icd9_icd10
xhipaasegmentdataaudit	#1 Transaction_Set_Segments	hipaa_x12_segment	hipaa_x12_segment_icd9_icd10

### Procedure

1. Open map source file compliance\_check in the Design Server.
2. Double-click the executable map to select the map for modification.
3. Edit the input card for the map to point to the appropriate \*\_icd9\_icd10 schema instead of the default schema.
4. Select YES when prompted, then replace all similar schema paths in all maps.
5. Build all maps.

## Changing maps impacted in the hipaa\_x12\_pass\_through source map

If both the ICD-9 and ICD-10 codes are required, the schemas in the hipaa\_x12\_pass\_through source map must be replaced.

### About this task

Follow this procedure for each map that is impacted in the hipaa\_x12\_pass\_through source map.

Maps impacted	Input card number and name	Default type schema	ICD-9 and ICD-10 schema
hipaa_x12_pass_through	#1 HIPAA_X12_Inbound_Transmission	hipaa_x12	hipaa_x12_icd9_icd10
hipaa_x12_pass_through	#2 HIPAA_X12_Outbound_Transmission	hipaa_x12	hipaa_x12_icd9_icd10
hipaa_x12_type_1	No changes are required.	-	-
hipaa_x12_type_2	#1 HIPAA_X12_Inbound_Transmission	hipaa_x12_type_2	hipaa_x12_type_2_icd9_icd10
hipaa_x12_type_2	#2 HIPAA_X12_Outbound_Transmission	hipaa_x12_type_2	hipaa_x12_type_2_icd9_icd10

### Procedure

1. Open source map hipaa\_x12\_pass\_through by using Design Server.
2. Double-click on the executable map to select it for modification.
3. Edit the input card that is listed to point to the appropriate \*\_icd9\_icd10 schema instead of the default type schema.
4. Select NO when prompted, then replace all similar schema paths in all of the maps.
5. Edit the Output card with the same number as the input card that you modified to point to the appropriate \*\_icd9\_icd10 schema instead of the default schema.
6. Select NO when prompted, then replace all similar schema paths in all maps.
7. Build the maps after all maps are updated.

## Changing other maps

This documentation describes changes to other maps in the Pack for Healthcare.

### About this task

Follow these procedures to change any of the remaining maps.

### Procedure

1. Open the map source by using the Design Server.
2. Double-click the executable map to select it for modification.
3. Edit the input of the output card that is using the hipaa\_x12 schema as the default schema to point to the hipaa\_x12\_icd9\_icd10 schema.

4. Select YES when prompted, then replace all similar schema paths in all maps.
  5. Build the maps after they are updated.
- 

## Maps

The Pack for Healthcare includes data transformation maps and maps for data validation. Data validation maps are either pass-through maps or related to the Compliance Check application.

- **[Data transformation maps](#)**

The data transformation maps that are included in the Pack for Healthcare convert data from one healthcare data format to another. The data transformation maps are described here.

- **[Data validation maps](#)**

The data validation maps that are included in the Pack for Healthcare are used for data validation and reporting.

- **[Compliance checking maps](#)**

The compliance checking maps verify compliance with HIPAA standards.

---

## Data transformation maps

The data transformation maps that are included in the Pack for Healthcare convert data from one healthcare data format to another. The data transformation maps are described here.

All data transformation maps have self-describing names with `_to_` separating the names of the source and target data formats. The executable map name is the same as the map name. By default, an execution summary audit log named `<executable-map-name>.log` is created in the map directory when the data transformation map is executed.

The maps included in the maps folder of the Pack for Healthcare fall into four categories:

- Institutional claims and coordination of benefits
- Professional claims and coordination of benefits
- Healthcare claims payments
- Healthcare claims status request and response

The maps install within the imported project.

- [\*\*Institutional claims and coordination of benefits\*\*](#)
  - [\*\*Professional claims and coordination of benefits maps\*\*](#)
  - [\*\*Healthcare claims payments maps\*\*](#)
  - [\*\*Healthcare claims status request and response maps\*\*](#)
- 

## Institutional claims and coordination of benefits

The following maps are used for institutional claims and benefit coordination:

- `cms_837i_5010_flat_to_hipaa`
  - `hipaa_837i_5010_to_cms_flat`
  - [\*\*`cms\_837i\_5010\_flat\_to\_hipaa.map`\*\*](#)
  - [\*\*`hipaa\_837p\_5010\_to\_cms\_flatmap.map`\*\*](#)
- 

## [\*\*cms\\_837i\\_5010\\_flat\\_to\\_hipaa map\*\*](#)

The `cms_837i_5010_flat_to_hipaa` map accepts CMS (formerly HCFA) Part A 5010 claims in flat-file format and generates a HIPAA transmission containing institutional claim and coordination of benefits (837i) transaction sets.

---

## [\*\*hipaa\\_837p\\_5010\\_to\\_cms\\_flatmap map\*\*](#)

The `hipaa_837p_5010_to_cms_flat` map accepts HIPAA professional claim (837p) data and generates the CMS Part A flat file.

---

## Professional claims and coordination of benefits maps

Professional claims and coordination of benefits maps, include these maps:

- `cms_837p_5010_flat_to_hipaa`
- `hipaa_837p_5010_to_cms_flat`
- [\*\*`cms\_837p\_5010\_flat\_to\_hipaa.map`\*\*](#)

- [hipaa\\_837p\\_5010\\_to\\_cms\\_flatmap.map](#)

---

## cms\_837p\_5010\_flat\_to\_hipaa map

The cms\_837p\_5010\_flat\_to\_hipaa map accepts CMS Part B 5010 claim data in flat file format and generates a HIPAA transmission containing professional claim and coordination of benefits (837p) transaction sets.

---

## hipaa\_837p\_5010\_to\_cms\_flatmap map

The hipaa\_837p\_5010\_to\_cms\_flat map accepts HIPAA professional claim (837p) data and generates the CMS Part A flat file.

---

## Healthcare claims payments maps

The following are healthcare claims payments maps:

- [cms\\_835\\_5010\\_flat\\_to\\_hipaa](#)
- [hipaa\\_835\\_5010\\_to\\_cms\\_flat](#)
- [\*\*cms\\_835\\_5010\\_flat\\_to\\_hipaa map\*\*](#)  
The cms\_835\_5010\_flat\_to\_hipaa map is used for data transformation from the CMS 835 professional and institutional flat file format to HIPAA transmissions containing 835 payment transactions.
- [\*\*hipaa\\_835\\_5010\\_to\\_cms\\_flat map\*\*](#)  
The hipaa\_835\_5010\_to\_cms\_flat map is for HIPAA transmissions containing 835 payment transactions in the CMS 835 professional and institutional flat-file format.

---

## cms\_835\_5010\_flat\_to\_hipaa map

The cms\_835\_5010\_flat\_to\_hipaa map is used for data transformation from the CMS 835 professional and institutional flat file format to HIPAA transmissions containing 835 payment transactions.

---

## hipaa\_835\_5010\_to\_cms\_flat map

The hipaa\_835\_5010\_to\_cms\_flat map is for HIPAA transmissions containing 835 payment transactions in the CMS 835 professional and institutional flat-file format.

---

## Healthcare claims status request and response maps

This section describes the following maps:

- [hipaa\\_276\\_5010\\_to\\_cms\\_flat](#)
- [cms\\_276\\_5010\\_flat\\_to\\_hipaa](#)
- [hipaa\\_277\\_5010\\_to\\_cms\\_flat](#)
- [cms\\_277\\_5010\\_flat\\_to\\_hipaa](#)
- [hipaa\\_277ca\\_5010\\_to\\_cms\\_flat](#)
- [cms\\_277ca\\_5010\\_flat\\_to\\_hipaa](#)
- [\*\*hipaa\\_276\\_5010\\_to\\_cms\\_flat map\*\*](#)  
The hipaa\_276\_5010\_to\_cms\_flat map accepts a HIPAA X12 transmission containing institutional claim status request 276 transaction sets. It generates professional and institutional claim status requests in CMS Part A and Part B flat file formats.
- [\*\*cms\\_276\\_5010\\_flat\\_to\\_hipaa map\*\*](#)  
The cms\_276\_5010\_flat\_to\_hipaa map accepts claim status requests in CMS Part A and Part B flat-file format and generates a HIPAA X12 transmission containing professional and institutional claim status request 276 transaction sets
- [\*\*hipaa\\_277\\_5010\\_to\\_cms\\_flat map\*\*](#)  
The hipaa\_277\_5010\_to\_cms\_flat map accepts HIPAA X12 transmission containing institutional claim status response 277 transaction sets and generates professional and institutional claim status response in CMS Part A and Part B flat file format.
- [\*\*cms\\_277\\_5010\\_flat\\_to\\_hipaa map\*\*](#)  
The cms\_277\_5010\_flat\_to\_hipaa map accepts claim status responses in CMS Part A and Prt B flat file format and generates a HIPAA X12 transmission containing professional and institutional claim status response 277 transaction sets.
- [\*\*hipaa\\_277ca\\_5010\\_to\\_cms\\_flat map\*\*](#)  
The hipaa\_277ca\_5010\_to\_cms\_flat map accepts a HIPAA X12 transmission containing claim acknowledgment 277ca transaction sets and generates claim acknowledgment in CMS flat file format.
- [\*\*cms\\_277ca\\_5010\\_flat\\_to\\_hipaa map\*\*](#)  
The cms\_277ca\_5010\_flat\_to\_hipaa map accepts claim acknowledgment in CMS flat file format and generates a HIPAA X12 transmission containing claim acknowledgment 277ca transaction sets.

## **hipaa\_276\_5010\_to\_cms\_flat map**

The hipaa\_276\_5010\_to\_cms\_flat map accepts a HIPAA X12 transmission containing institutional claim status request 276 transaction sets. It generates professional and institutional claim status requests in CMS Part A and Part B flat file formats.

## **cms\_276\_5010\_flat\_to\_hipaa map**

The cms\_276\_5010\_flat\_to\_hipaa map accepts claim status requests in CMS Part A and Part B flat-file format and generates a HIPAA X12 transmission containing professional and institutional claim status request 276 transaction sets

## **hipaa\_277\_5010\_to\_cms\_flat map**

The hipaa\_277\_5010\_to\_cms\_flat map accepts HIPAA X12 transmission containing institutional claim status response 277 transaction sets and generates professional and institutional claim status response in CMS Part A and Part B flat file format.

## **cms\_277\_5010\_flat\_to\_hipaa map:**

The cms\_277\_5010\_flat\_to\_hipaa map accepts claim status responses in CMS Part A and Prt B flat file format and generates a HIPAA X12 transmission containing professional and institutional claim status response 277 transaction sets.

## **hipaa\_277ca\_5010\_to\_cms\_flat map**

The hipaa\_277ca\_5010\_to\_cms\_flat map accepts a HIPAA X12 transmission containing claim acknowledgment 277ca transaction sets and generates claim acknowledgment in CMS flat file format.

## **cms\_277ca\_5010\_flat\_to\_hipaa map:**

The cms\_277ca\_5010\_flat\_to\_hipaa map accepts claim acknowledgment in CMS flat file format and generates a HIPAA X12 transmission containing claim acknowledgment 277ca transaction sets.

## **Data validation maps**

The data validation maps that are included in the Pack for Healthcare are used for data validation and reporting.

These maps are either pass-through maps, or they are related to the Compliance Check application for HIPAA X12 data validation.

- [Pass-through maps](#)

## **Pass-through maps**

Also known as validation maps, pass-through maps validate data definitions by using the same schema for the input cards as the output cards.

Pass-through maps have self-describing names with \_pass\_through following the name of the data format. The executable map name is the same as the map source file name. By default, an execution summary audit log named <executable-map-name>.log is created in the map directory when the pass-through map is executed.

This section describes the following map source files:

- hipaa\_x12\_pass\_through
- cms\_276\_5010\_flat\_pass\_through
- cms\_277\_5010\_flat\_pass\_through
- cms\_277ca\_5010\_flat\_pass\_through
- cms\_835\_5010\_flat\_pass\_through
- cms\_837i\_5010\_flat\_pass\_through
- cms\_837p\_5010\_flat\_pass\_through
- [Source map: hipaa\\_x12\\_pass\\_through](#)  
The hipaa\_x12\_pass\_through source map uses the hipaa\_x12 schema and checks WEDI/SNIP Type 1 through Type 4. The map validates HIPAA X12 transmission data against the Partner X12 Inbound Transmission EDI and Partner X12 Outbound Transmission EDI definitions in the hipaa\_x12 schema.
- [Source map: cms\\_276\\_5010\\_flat\\_pass\\_through](#)  
The cms\_276\_5010\_flat\_pass\_through.mms source map validates CMS claim status request (276) flat file data against the cms\_276\_5010 flat file data definition

- in the **cms\_276\_277\_5010\_flat** schema.
- **Source map: cms\_277\_5010\_flat\_pass\_through**  
The **cms\_277\_5010\_flat\_pass\_through** source map validates CMS claim status response (277) flat file data against the **cms\_277\_5010\_flat** file data definition in the **cms\_276\_277\_5010\_flat** schema.
  - **Source map: cms\_277ca\_5010\_flat\_pass\_through**  
The **cms\_277ca\_5010\_flat\_pass\_through** source map validates CMS claim acknowledgment (277ca) flat file data against the **cms\_277ca\_5010\_flat** file data definition in the **cms\_277ca\_5010\_flat** schema.
  - **Source map: cms\_835\_5010\_flat\_pass\_through**  
The **cms\_835\_5010\_flat\_pass\_through** source map validates CMS claim payment 835 flat file data against the **cms\_835\_5010a1\_flat\_file** data definition in the **cms\_835\_5010a1\_flat** schema.
  - **Source map: cms\_837i\_5010\_flat\_pass\_through**  
The **cms\_837i\_5010\_flat\_pass\_through** source map validates CMS institutional claim and coordination of benefits 837I flat file data against the **cms\_837i\_5010a2\_flat\_file** data definition in the **cms\_837i\_5010a2\_flat** schema.
  - **Source map: cms\_837p\_5010\_flat\_pass\_through**  
The **cms\_837p\_5010\_flat\_pass\_through** source map validates CMS professional claim and coordination of benefits 837P flat file data against the **cms\_837p\_5010a1\_flat\_file** data definition in the **cms\_837p\_5010a1\_flat** schema.
- 

## Source map: hipaa\_x12\_pass\_through

The **hipaa\_x12\_pass\_through** source map uses the **hipaa\_x12** schema and checks WEDI/SNIP Type 1 through Type 4. The map validates HIPAA X12 transmission data against the Partner X12 Inbound Transmission EDI and Partner X12 Outbound Transmission EDI definitions in the **hipaa\_x12** schema.

This map also includes some pre-defined data audit log settings that can be left as-is or modified and then activated by setting Data SettingsAudit in MapAudit to ON.

## Source map: cms\_276\_5010\_flat\_pass\_through

The **cms\_276\_5010\_flat\_pass\_through.mms** source map validates CMS claim status request (276) flat file data against the **cms\_276\_5010** flat file data definition in the **cms\_276\_277\_5010\_flat** schema.

## Source map: cms\_277\_5010\_flat\_pass\_through

The **cms\_277\_5010\_flat\_pass\_through** source map validates CMS claim status response (277) flat file data against the **cms\_277\_5010\_flat** file data definition in the **cms\_276\_277\_5010\_flat** schema.

## Source map: cms\_277ca\_5010\_flat\_pass\_through

The **cms\_277ca\_5010\_flat\_pass\_through** source map validates CMS claim acknowledgment (277ca) flat file data against the **cms\_277ca\_5010\_flat** file data definition in the **cms\_277ca\_5010\_flat** schema.

## Source map: cms\_835\_5010\_flat\_pass\_through

The **cms\_835\_5010\_flat\_pass\_through** source map validates CMS claim payment 835 flat file data against the **cms\_835\_5010a1\_flat\_file** data definition in the **cms\_835\_5010a1\_flat** schema.

## Source map: cms\_837i\_5010\_flat\_pass\_through

The **cms\_837i\_5010\_flat\_pass\_through** source map validates CMS institutional claim and coordination of benefits 837I flat file data against the **cms\_837i\_5010a2\_flat** file data definition in the **cms\_837i\_5010a2\_flat** schema.

## Source map: cms\_837p\_5010\_flat\_pass\_through

The **cms\_837p\_5010\_flat\_pass\_through** source map validates CMS professional claim and coordination of benefits 837P flat file data against the **cms\_837p\_5010a1\_flat\_file** data definition in the **cms\_837p\_5010a1\_flat** schema.

## Compliance checking maps

The compliance checking maps verify compliance with HIPAA standards.

The following compliance checking maps are provided in the Pack for Healthcare:

- compliance\_check
  - utilities\_validation\_check
- 

## Sample data

Sample data is provided for use with the schemas and maps in the Pack for Healthcare.

- [Overview of sample data](#)

The data files contain either transaction data in one or more data exchange formats or application data used by the Compliance Check application for HIPAA X12 data validation.

- [Transaction data](#)

This contains sample transaction data for HIPAA X12 and CMS.

- [Compliance check data](#)

The Pack for Healthcare contains data files that are used by the Compliance Check application for HIPAA X12 data validation.

---

## Overview of sample data

The data files contain either transaction data in one or more data exchange formats or application data used by the Compliance Check application for HIPAA X12 data validation.

The Pack for Healthcare provides sample data for HIPAA X12, CMS, and Compliance Checker. For more information about the Compliance Checker data files, see [HIPAA Compliance Checking](#).

The data files are installed within the imported project where they are used, as there are multiple projects that reference different sample data.

---

## Transaction data

This contains sample transaction data for HIPAA X12 and CMS.

- [HIPAA X12](#)

The HIPAA X12 sample data includes transmission data for the finalized HIPAA X12 transaction set definitions.

- [CMS sample data](#)

The CMS sample data includes transmission data for the flat file interface data formats associated with the HIPAA 5010 transaction definitions, and the finalized addenda definitions.

---

## HIPAA X12

The HIPAA X12 sample data includes transmission data for the finalized HIPAA X12 transaction set definitions.

There is one file per version, or industry code, and one file that contains all of the sample data.

The naming convention includes the hipaa transaction version, and the x12 transaction number(s).

For example, the sample data for the 276 transaction defined in the HIPAA 5010X212 TR3 is named hipaa\_5010\_x212\_276\_sample.dat.

There is also a sample data file that contains all the transactions supported. This is named all\_hipaa\_5010\_transaction\_sets.dat.

---

## CMS sample data

The CMS sample data includes transmission data for the flat file interface data formats associated with the HIPAA 5010 transaction definitions, and the finalized addenda definitions.

The sample data is based on transformations of the HIPAA X12 transaction sets data included with the Pack for Healthcare.

The naming convention includes the HIPAA transaction version, and x12 transaction number(s).

For example, the sample for the CMS 276 flat file format is cms\_flat\_5010\_x212\_276\_sample.dat.

---

## Compliance check data

The Pack for Healthcare contains data files that are used by the Compliance Check application for HIPAA X12 data validation.

The following Compliance Check data files are provided:

- compliance\_check\_5010\_parameter.dat (default)
- hipaa\_x12\_qualifier.dat
- hipaa\_x12\_structure.dat

- hipaa\_x12\_type\_6\_value.dat
  - hipaa\_x12\_type\_7\_value.dat
  - tack\_codetable.dat
  - compliance\_check\_tack.css
  - plus.gif
  - minus.gif
- 

## HIPAA Compliance Checking

The Pack for Healthcare uses the types of HIPAA data transmission testing documented in the "Transaction Compliance and Certification" white paper developed by the workgroup for Electronic Data Interchange (WEDI) Strategic National Implementation Process (SNIP) Transactions Work Group - Testing Sub Work-Group; Dated June 11, 2002.

HIPAA Compliance Check for HIPAA X12 data validation and compliance reporting contains a series of executable maps used to validate HIPAA X12 transmission data and to report compliance with industry requirements. The compliance\_check file is the source file for the HIPAA Compliance Check. This provides a non-partner configurable way to check WEDI/SNIP Type 1 through Type 4.

HIPAA Data compliance has greater capabilities than the HIPAA Compliance Check. In addition to checking WEDI/SNIP Type 1 through Type 4, it also enables checking WEDI/SNIP Type 5 through Type 7. External Code set values will be checked against values loaded into a MongoDB or similar noSQL Database.

- [\*\*WEDI SNIP transaction compliance types\*\*](#)

The HIPAA Compliance Checker supports the data validation and compliance reporting for Types 1 through 7. WEDI SNIP Type 1 through 7 validation 'rules' are included in the components of the product. The rules for the higher levels of validation are user-configurable.

- [\*\*Type 1 EDI Standard Integrity Testing\*\*](#)

Type 1 EDI Standard Integrity Testing validates the basic syntactical integrity of the EDI submission.

- [\*\*Type 2 HIPAA Implementation Guide Requirement Testing\*\*](#)

Type 2 testing involves testing for HIPAA implementation guide-specific syntax requirements.

- [\*\*Type 3 HIPAA Balance Testing\*\*](#)

Type 3 balance testing ensures that the sum of the claim line item amounts is equal to the total claim amount.

- [\*\*Type 4 HIPAA Inter-segment Situation Testing\*\*](#)

Type 4 involves testing specific inter-segment situations described in the HIPAA implementation guides, such that if A occurs, then B must be populated. This is considered to include the validation of situational fields given values or situational elements present in other data segments. For example, the accident date must be present if the claim is for an accident.

- [\*\*Type 5 HIPAA External Code Set Testing\*\*](#)

Type 5 validation involves validating data content against code sets from external code sources as explicitly defined in the implementation guides for use with HIPAA standards. Examples of HIPAA X12 code sets include the 508 Health Care Claim Status Codes, available from Washington Publishing Company. Another example is the 537 National Provider Identifiers which are available from the Centers for Medicare and Medicaid (CMS).

- [\*\*Type 6 HIPAA Line of Service Testing\*\*](#)

Type 6 validation is used to ensure that HIPAA transaction data for claims meets specific requirements that are related to specialized healthcare service, or products such as medically related transport, durable medical equipment, or chiropractic treatment.

- [\*\*Type 7 Companion Guide Testing\*\*](#)

Some HIPAA specific Medicare, Medicaid and Indian Health requirements are contained in the HIPAA Implementation Guides. Not all trading partners, however, require compliance with these requirements. If a trading partner wants to exchange transactions with one of these payers, Type 7 validation is required.

- [\*\*Parameters for HIPAA processing\*\*](#)

---

## WEDI SNIP transaction compliance types

The HIPAA Compliance Checker supports the data validation and compliance reporting for Types 1 through 7. WEDI SNIP Type 1 through 7 validation 'rules' are included in the components of the product. The rules for the higher levels of validation are user-configurable.

Types 1 through 7 of HIPAA data transmission testing are defined as follows:

- Type 1 EDI Standard Integrity Testing
  - Type 2 HIPAA Implementation Guide Requirement Testing
  - Type 3 HIPAA Balance Testing
  - Type 4 HIPAA Inter-Segment Situation Testing
  - Type 5 HIPAA External Code Set Testing
  - Type 6 HIPAA Line of Service Testing
  - Type 7 Companion Guide Requirement Testing
- 

## Type 1 EDI Standard Integrity Testing

Type 1 EDI Standard Integrity Testing validates the basic syntactical integrity of the EDI submission.

Type 1 testing conducts a test of the EDI file for the following:

- valid segments
- segment order
- element attributes
- numeric values in numeric data elements
- validation of X12 syntax
- compliance with X12 semantic rules

Reporting options for Type 1 exceptions identified by the Transformation Extender HIPAA Compliance Checker includes the TA1 interchange acknowledgment, the standard X12 997 functional acknowledgment, the HIPAA 999 implementation acknowledgment, and the Translated Acknowledgment report.

## Type 2 HIPAA Implementation Guide Requirement Testing

Type 2 testing involves testing for HIPAA implementation guide-specific syntax requirements.

Type 2 testing checks for the following:

- Limits on repeat counts
- Used and not used qualifiers
- Elements and segments

There is also testing for:

- HIPAA required or intra-segment situational data elements
- Values and codes explicitly called out in the HIPAA Implementation Guides.
- Values and codes noted in the implementation guide (by means of an X12 code list or table)
- HL, LX, ENT, IT1 sequencing
- HL parent-child relationships

The options for reporting Type 2 exceptions identified by the Transformation Extender HIPAA Compliance Checker include the TA1 interchange acknowledgment, the HIPAA 999 implementation acknowledgment, and the Translated Acknowledgement report.

## Type 3 HIPAA Balance Testing

Type 3 balance testing ensures that the sum of the claim line item amounts is equal to the total claim amount.

Type 3 testing involves checking a transaction for the following conditions:

- Balanced field totals
- Financial balancing of claims or remittance advice
- Balancing of summary fields (if appropriate)

The options for reporting Type 3 exceptions identified by the Transformation Extender HIPAA Compliance Checker include the HIPAA 999 implementation acknowledgment and the Translated Acknowledgment report.

## Type 4 HIPAA Inter-segment Situation Testing

Type 4 involves testing specific inter-segment situations described in the HIPAA implementation guides, such that if A occurs, then B must be populated. This is considered to include the validation of situational fields given values or situational elements present in other data segments. For example, the accident date must be present if the claim is for an accident.

If the XML\_Validation parameter is set to X or S, validation of the XML data in the BIN segment of the 4050 or 5010 275 (Additional Information to Support a Health Care Claim) is supported under Type 4 testing.

The options for reporting Type 4 exceptions identified by the Transformation Extender HIPAA Compliance Checker includes the HIPAA 999 implementation acknowledgment and the Translated Acknowledgement report.

## Type 5 HIPAA External Code Set Testing

Type 5 validation involves validating data content against code sets from external code sources as explicitly defined in the implementation guides for use with HIPAA standards. Examples of HIPAA X12 code sets include the 508 Health Care Claim Status Codes, available from Washington Publishing Company. Another example is the 537 National Provider Identifiers which are available from the Centers for Medicare and Medicaid (CMS).

The options for reporting Type 5 exceptions include the HIPAA 999 implementation acknowledgment and the Translated Acknowledgment report.

Applicable Type 5 rules for the Claim Transactions (837) will now use the service date from the claim and not the processing date to determine if the code is valid for this particular date.

## Date Segments used for Type 5 Validation

For Type 5 validation for codes within segments in the Claim (2300) Loop, such as HI:

- Institutional Claim validation will use 2300 Statement Dates, DTP\*434, a required segment that specifies a date range.
- Professional Claim validation will use the range of dates from the earliest to the latest Service Level Dates in the 2400 DTP\*472.
- Dental claim validation will use the 2300 Service Date, DTP\*472, if present. This can be a single date or a range. If this is not present, the range of 2400 DTP\*472 Service Dates will be used.

For Type 5 validation for codes within segments in the Service (2400) Loop, such as SV1, SV2, and SV3:

- Institutional claim validation will use 2400 Service Date if present. If this is not present, the 2300 Statement Date DTP\*434 will be used.

- Professional claim validation will use the Service Level Date in 2400 DTP\*472.
- Dental claim validation will use the 2400 Service Date, DTP\*472, if present. If this is not present, the 2300 DTP\*472 Service Date will be used.

## Code list applicable for Service Date used in Type 5 Validation

The following code lists are used in fields present in the claim and service loops of claim transactions, which represent medical code sets that could have start and/or end dates for the code value usage.

For some of these, the default validation is Pattern Match, as the Healthcare Pack does not include these code sets for licensing or other reasons. For the new Service Date functionality to be used with these, the user will need to load those code sets with included start and/or end dates and update the corresponding rules to use CV or Code Value instead of PM or Pattern Match.

Code source	Description
130	Healthcare Common Procedural Coding System
131	International Classification of Diseases, 9th Revision, Clinical Modification (ICD-9-CM)
132	National Uniform Billing Committee
133	Current Procedural Terminology (CPT) Codes
135	American Dental Association
229	Diagnosis-Related Group Number
240	National Drug Code by Format
513	Home Infusion EDI Coalition (HIEC) Product/Service Code List
576	Workers Compensation Specific Procedure and Supply Codes
716	Health Insurance Prospective Payment System (HIPPS) Rate Code for Skilled Nursing Facilities
843	Advanced Billing Concepts (ABC) Codes
896	International Classification of Diseases, 10th Revision, Procedure Coding System (ICD-10-PCS)
897	International Classification of Diseases, 10th Revision, Clinical Modification (ICD-10-CM)

## Type 6 HIPAA Line of Service Testing

Type 6 validation is used to ensure that HIPAA transaction data for claims meets specific requirements that are related to specialized healthcare service, or products such as medically related transport, durable medical equipment, or chiropractic treatment.

For example, the Situational Rule for the Ambulance Transport Information (CR1 Segment in the Claim Information (2300) Loop of the Health Care Claim: Professional (X222A1 -837)) transaction set states that the segment is "Required on all claims involving ambulance transport services."

- **HIPAA operating rules**

HIPAA Operating Rules Section of the Affordable Care Act (ACA) was enacted and amended in March 2010 and requires the adoption of a single set of operating rules for each of the transactions listed in the ACA.

## HIPAA operating rules

HIPAA Operating Rules Section of the Affordable Care Act (ACA) was enacted and amended in March 2010 and requires the adoption of a single set of operating rules for each of the transactions listed in the ACA.

There are several types of Operating Rules in CAQH CORE Phase I and Phase II, and the rules surrounding the standard compliance of the data content are the type of rules that are implemented in the HIPAA Compliance Check module. Other categories of rules (such as rules governing response time and rules related to the formatting of Implementation Guides) are not enforced by the Compliance Check module. These rules are expressed as if they were WEDI/SNIP Type 4 rules, which allows them to be reported on the 999 Business Acknowledgment. By default, these rules are disabled when the product is installed and must be manually enabled in order for these data extra checks to be performed.

The rule text reported on a failure of one of these rules will begin with an Operating Rule Indicator ("OR") plus the phase of CAQH CORE which describes the rule ("I" or "II"). For example:

"ORI - 2110C EB05 required when 2110C present"

## Type 7 Companion Guide Testing

Some HIPAA specific Medicare, Medicaid and Indian Health requirements are contained in the HIPAA Implementation Guides. Not all trading partners, however, require compliance with these requirements. If a trading partner wants to exchange transactions with one of these payers, Type 7 validation is required.

When a trading partner is certified for compliance, the certification service must indicate whether or not the payer specific requirements are met.

Trading partners can have their own business requirements, but the requirements must be listed in the HIPAA Implementation Guides in order to meet HIPAA requirements.

## Parameters for HIPAA processing

- **HIPAA Data Compliance Parameters**

Parameters for the HIPAA Data Compliance are set in the JSON formatted parameter file, hipaa\_data\_compliance\_parameter.json.

- **HIPAA Compliance Check Parameters**

Parameters used for the Compliance Check application are set in the compliance\_check\_5010\_paramater.dat file.

## HIPAA Data Compliance Parameters

Parameters for the HIPAA Data Compliance are set in the JSON formatted parameter file, hipaa\_data\_compliance\_parameter.json.

This file contains a meaningful grouping of parameters:

- GeneralValidation
- ValidationLevels
- Acknowledgments
- Tuning

The following table has details of the setting allowed. If any field is not included in the hipaa\_data\_compliance\_parameter.json file, the default will be used.

In the following table, the \* (asterisk) indicates the default value.

Since the format of this parameter file is JSON, the name and parameter values that are text strings must always be enclosed in double quotes. The value, if Boolean is true or false, does not need to be quoted. A few of the parameters are numeric values, and they do not need to be quoted either.

Examples:

"Enabled": false

"WhenIfEnabled": "Errors"

It is recommended that a JSON aware editor is used, that will flag invalid syntax such as missing or extra commas and quotes.

Parameter tag name	Values	Description
Environment ProductName	H	The ProductName parameter is used to identify the product that has invoked the compliance check application. If parameter is used, it should always be H.
Environment OperatingSystem	L A U M W	The OperatingSystem parameter is used to override the dynamic determination of the UNIX operating system from the format of the input data file. It is always recommended to specify the operating system using this parameter. Allowable settings include: L (Linux), A (AIX), U (USS), M (MVS - z/OS), W (Windows)
Environment BaseMapDirectory	C:\\DIR\\ Or /dir/	The BaseMapDirectory parameter specifies the directory for compiled maps that will be executed by hipaa_data_compliance.mmc via the RUN function. There is NO default setting for this parameter. If the parameter is not specified, then the directory from which hipaa_data_compliance.mmc is being executed will be used. This parameter cannot be used for z/OS. Note: Final slash must be used. Since the input is in JSON Format back slash must be doubled if specifying directory on Windows.
Environment BaseExitDirectory	C:\\DIR\\ Or /dir/	The BaseExitDirectory parameter specifies the directory for executables invoked via the EXIT function. There is NO default setting for this parameter. If the parameter is not specified, then the system library path will be used. This parameter cannot be used for z/OS. Note: Final slash must be used. Since the input is in JSON Format back slash must be doubled if specifying directory on Windows.
Environment DataBaseQueryType	JDBC MONGO*	The DataBaseQueryType is used to indicate the type of database access for Code List validation in Type 5 and higher checks. The MRN file must still be updated for the specific access parameters.
GeneralValidation Validation Method	Map* Compliance Both	Map: This default setting sends all data through the pass through validation maps and only the invalid data goes through the subsequent processing, and also valid data that must be processed for rules only supported by the detailed compliance check processing. This is the recommended, and the default setting.  Compliance: This setting causes all data to be passed to the detailed processing done by the Compliance Check application.  Both: This setting causes all data to be processed both by the pass through validation maps and the detailed compliance check validation. If there is a large amount of data, this option can slow the process. This option is used for troubleshooting purposes, logic modules, or both.
GeneralValidation RejectUnsupported5010Version	true* - Reject unsupported 5010 versions in the envelope checking process.  false - Don't reject unsupported versions, and validate them at currently supported versions.	Indicates whether to reject an unsupported HIPAA 5010 version or whether to validate the data to the currently supported errata version.
GeneralValidation BINSegXMLValidation Enabled	true* - Validate Contents of BIN segment according to SchemaOrWellFormed Value. false - Never validate the contents of the BIN segment.	Indicates the validation level for XML BIN/BDS segment content.

Parameter tag name	Values	Description
GeneralValidation BINSegXMLValidation SchemaOrWellFormed	Schema - Validate the contents of the BIN segment for proper XML formatting and against the schema. XML - Validate the contents of the BIN segment for proper XML formatting and against the schema.	Type of validation to perform on BIN/BDS segment.
GeneralValidation n DuplicateISAChecks	true* false	Indicates whether interchange (ISA) envelopes should be checked for duplicates.
GeneralValidation n DuplicateIGSChecks	true false*	Indicates whether Functional Group control numbers should be checked for duplicates within an interchange.
GeneralValidation n DuplicateSTChecks (No I)	true* false	Indicates whether transaction set (ST) envelopes should be checked for duplicates.
GeneralValidation n Allow_GS_Leading_0	true false*	Indicates whether to allow leading zeros in the Functional Group control number (such as 0001) in the GS and GE segments.
GeneralValidation n ClaimLevelRejection Enabled	true false*	Indicates whether the transaction set is still accepted and the claim(s) in error are removed from the valid data. This parameter is only available to be used when the input HIPAA EDI data is a Claim (837).
GeneralValidation n ClaimLevelRejection Level	All – Only reject transaction set when all claims are in error or when an error is outside the Provider loops. If error is at the provider, subscriber, or patient level, all claims for that provider, subscriber, or patient will be rejected.  Claim - Reject transaction set when all claims are in error, or when error is outside of Claim Loop.	If the ClaimLevelRejection Enabled is true, this controls at which level the valid and invalid claims are separated.
ClaimLevelRejection DuplicateCLMCheckFullSegment	true false*	The Duplicate CLMCheckFullSegment controls whether just CLM01 or the full CLM segment is used to determine if claim level rejection processing can continue for this transaction.  If the parameter is set to true - Disable Claim level rejection, if the entire CLM segment is not unique for all claims in the transaction.  Note: Only CLM01 is used to populate the TRN information in the 277CA, so the user may have to handle any ambiguity in the resultant acknowledgment, if true is used
ClaimLevelRejection ReportErrorInSTC	true false*	Set the sub elements in the STC01 element at all 277CA and 277DR loop levels to reflect if all valid or an error was detected at that level or below and include a valid entity code instead of the generic Ack/Receipt code.
Tack HTMLOrJSON	HTML* JSON BOTH	Use the HTMLOrJSON parameter set to JSON or BOTH to optionally create ..../data/compliance_check_TAck.json in output card 13.  When either HTML or BOTH is used, the /data/compliance_check_TAck.html is produced in output card 11.  Note: Both creation and content are still controlled by the other parameters in the Tack group. This parameter only controls how the Tack is formatted.
ValidationLevels Type1 Enabled	true false*	Indicates whether WEDI/SNIP Type 1 checks should be performed on the input data. (Note any Type 1 error will be found if Type 2 is enabled; this only runs the Type 1 validation as a separate step, so it is not recommended).
ValidationLevels Type2 Enabled	true* false	Indicates whether WEDI/SNIP Type 2 checks should be performed on the input data.
ValidationLevels Type2 RelaxedStructureCheck	true false*	Indicates whether relaxed structure should be performed on the input data if it fails the type 2 structure checking process. This check will only be performed if there are errors found during the type 2 structure checking process - if no errors are found, it will be bypassed. It also can only be executed if type 2 checks are enabled.
ValidationLevels Type3 Enabled	true* false	Indicates whether WEDI/SNIP Type 3 checks should be performed on the input data.
ValidationLevels Type4 Enabled	true* false	Indicates whether WEDI/SNIP Type 4 checks should be performed on the input data.
ValidationLevels Type5 Enabled	true false*	Indicates whether WEDI/SNIP Type 5 checks should be performed on the input data.

Parameter tag name	Values	Description
ValidationLevels Type6 Enabled	true false*	Indicates whether WEDI/SNIP Type 6 checks should be performed on the input data.
ValidationLevels Type7 Enabled	true false*	Indicates whether WEDI/SNIP Type 7 checks should be performed on the input data.
ValidationLevels Type7 RejectLevel	Errors	Exceptions found in the custom user exit are treated as errors, and data will be rejected.
ValidationLevels Type7 ExitType	Java C None*	Indicates the language in which the custom user exit is written.
ValidationLevels Type7 ExitName	(User specific value) None*	Indicates the name of the custom user exit library name if the exit is written in C or jar file if the exit is written in Java.
ValidationLevels Type7 ExitFunction	(User specific value) None*	Indicates the name of the custom user exit function name within the library or jar file.
ValidationLevels Type7 ExitPartnerDetail	true false*	Include ISA Sender and Receiver IDs, ISA Control number, GS Sender and Receiver Codes and GS Control number in the transaction data passed to the Type 7 user exit.
Acknowledgments TA1 Enabled	true false*	Indicates whether the results from the WEDI/SNIP Type 1 interchange envelope checks should be reported using the X12 standard TA1 segment.
Acknowledgments TA1 WhenEnabled	Errors* - Generate an X12 standard TA1 for the interchange only when the results of the compliance check indicate the presence of Type 1 errors in the interchange envelope.  Always - Always generate an X12 standard TA1 for all interchanges even if compliance check does not indicate the presence of Type 1 errors in the interchange envelope.  Requested - Generate an X12 standard TA1 for all interchanges based on the value in the input ISA14.  Input ISA14 value of 0 - No Interchange Acknowledgment Requested will identify the same behavior as Never.  Input ISA14 value of 1 - Interchange Acknowledgment Requested (TA1) will identify the same behavior as Always.	If Enabled, determine when the TA1 is generated.
Acknowledgments TA1 Separate	true* false	Indicates whether TA1 acknowledgments should be created in a separate response transmission or be combined with the response transmission that contains the 997 acknowledgment transaction sets.
Acknowledgments 997 Enabled	true false*	Indicates whether the results from the WEDI/SNIP Type 1 checks should be reported using the X12 standard 997 transaction set.  If Type 1 checks are not enabled, this will be set to false.
Acknowledgments 997 WhenEnabled	Errors* - Generate an X12 standard 997 for the interchange only when the results of the compliance check indicate the presence of Type 1 errors in the interchange  Always - Always generate an X12 standard 997 for all interchanges even if compliance check does not indicate the presence of Type 1 errors.	Indicates when the results from the WEDI/SNIP Type 1 checks should be reported using the X12 standard 997 transaction set.
Acknowledgments 997 DetailMax	(User-specified numeric value) 9999*	Indicates when we should stop producing detailed 997 error information for a single transaction. If the number of error lines, not total errors, to be produced in the 997 for the received transaction exceeds this limit, then the logic to generate the detailed error information in the AK3/AK4 loop will be bypassed. As the number of errors for a single transaction increases, the processing time required to detect and report these also increases, so increasing this limit will decrease performance when processing transactions containing numerous errors.  The setting for this value will determine the maximum number of error lines produced in the 997 for a single transaction.

Parameter tag name	Values	Description
Acknowledgments_HIPAA_TA1_Enabled	true false*	Indicates whether the results from the WEDI/SNIP Type 2 interchange envelope checks using HIPAA specific code values should be reported using the TA1 segment.
Acknowledgments_HIPAA_TA1_WhenIfEnabled	Errors* - Generates an X12 standard TA1 for the interchange only Type 1 errors present in the interchange envelope.  Always - Always generates an X12 standard TA1 for all interchanges, even if Type 1 errors are not present in the interchange envelope.  Requested - Generates an X12 standard TA1 for all interchanges based on the value in the input ISA14.  Input ISA14 value of 0 - No Interchange Acknowledgment Requested will identify the same behavior as Never.  Input ISA14 value of 1 - Interchange Acknowledgment Requested (TA1) will identify the same behavior as Always.	If Enabled, determine when the TA1 is generated.
Acknowledgments_999_Enabled	true false*	Indicates whether the results from the WEDI/SNIP checks should be reported using the HIPAA specific 999 transaction set.
Acknowledgments_999_WhenIfEnabled	Errors - Generates only when WEDI/SNIP type errors are present in the interchange.  Always* - Always generates even if WEDI/SNIP type errors are not present in interchange.	Indicates when the results from the WEDI/SNIP checks should be reported using a HIPAA specific 999.
Acknowledgments_999_IK3MissingSeg	true false*	For implementation dependent missing segments (I6), use true for this setting if the missing segment is to be reported in the IK3. This will result in more information in the 999 by allowing the existing segment to be included in a CTX segment. Since the segment is missing, the segment position indicated will vary – it may be the position of the existing segment that caused the requirement, the segment location in the transaction where it was determined the missing segment was not present, or the end of a loop.  Use false to report the existing segment, which caused the requirement for the missing segment to be reported in the IK3.
Acknowledgments_999_DetailMax	(User specified numeric value) 9999*	Indicates when we should stop producing detailed HIPAA 999 error information for a single transaction. If the number of error lines, not total errors, to be produced in the HIPAA 999 for the received transaction exceeds this limit, then the logic to generate the detailed error information in the AK3/AK4 loop will be bypassed.  As the number of errors for a single transaction increases, the processing time required to detect and report these also increases, so increasing this limit will decrease performance when processing transactions containing numerous errors.  The setting for this value will determine the maximum number of error lines produced in the HIPAA 999 for a single transaction.
Acknowledgments_277CA_Enabled	true false*	Indicates whether the results from the Claim Level Rejection process should be reported using the 5010 277CA transaction set.
Acknowledgments_277CA_WhenIfEnabled	true false*	Indicates when the results from the WEDI/SNIP checks should be reported using a 277CA.
Acknowledgments_277CA_Level	Provider* - 277CA reports details at the Provider Level. This option will not provide specific details on each claim, only on the totals accepted/rejected for each Provider.  Claim - 277CA reports details at the Claim Level. This option will provide specific details on each Claim within the transaction.  All - 277CA reports all details at both Provider and Claim. If there are no errors, then the reporting will include the Patient/Claim details.	Indicates the level at which the HIPAA 277 Claims Acknowledgment reports. It can show details at the Provider level, or it can show details down to the Patient/Claim Level. This parameter will only have an impact if the Claim Level Rejection is enabled and the input HIPAA EDI data is a Claim (837). If there are no errors in the HIPAA EDI 837 data, then reporting will only go to the Provider Level, regardless of the setting.  Using options Claim or All may impact performance if there is a large quantity of claims in the data.

Parameter tag name	Values	Description
Acknowledgments 277CA UniqueBHT03	true false*	Indicates whether to produce a BHT03 value based on the interchange, functional group, and transaction control numbers of the source 837 being validated.  If false, 277X2140001 is always set as a value for the BHT03 element.
Acknowledgments TACK Enabled	true false*	Indicates whether the Translated Acknowledgment Report will be produced. This report transforms the 997 and 999 acknowledgments into a more readable report.
Acknowledgments TACK WhenIfEnabled	Errors* - Generate only when WEDI/SNIP type errors are present in the interchange.  Always - Always generate even if WEDI/SNIP type errors are not present in interchange.	Indicates when the results from the WEDI/SNIP checks should be reported using Tack report.
Acknowledgments TACK IncludeAccepted	true false*	Indicates the level of reporting in the Translated Acknowledgment Report.  If true, report all data, including those accepted and rejected. If false, only generate the report based on the data in error.
Acknowledgments TACK TAck_EDI	Full* – Generates TAck_EDI.html file with all the transactions in the data.  Partial - Generates TAck_EDI.html file with only the transactions in error.  None	Indicates whether the TAck_EDI file, which is a html tagged version of the transactions being validated, is produced with all the original data, or just the transactions that have errors reported.  If Acknowledgments Tack Enabled is false, this parameter is not used.  If Acknowledgments Tack Enabled is true, the default for this is Full.  Options Partial or None may help overall performance.
Acknowledgments LoopIDInAck Enabled	true* false	Include loop identifier in the acknowledgment. If a 997 is requested and this parameter is set to true, then the AK303 will contain the loop identifier of the segment in error. If a 999 is requested and this parameter is set to true, then the IK303 will contain the loop identifier of the segment in error.
Acknowledgments LoopIDInAck includeIDForAll	true false*	Indicates whether to include the functional group control number in the grouping of functional groups within an acknowledgment.  If the setting is false, only the GS02, GS03, GS07, and GS08 are used to determine a unique functional group for reporting in the acknowledgment.  If the setting is true, then the GS06 is also included in the determination.
Acknowledgments IncludeISA13InAcks	true false*	The IncludeISA13InAcks parameter is used when it is desired to use the ISA13 (control number) of the input transmission file for acknowledgments produced. If the parameter is set to true, the ISA13 value of the input transmission file will be used.  If the parameter is set to false, a default control number will be used.  Note: In the case of multiple interchanges in the input transmission, the ISA13 of the first interchange will be used, as only one interchange is produced in the acknowledgments.
Acknowledgments AckFunctionalGroups IncludeGS06	true false*	Indicates whether to include the functional group control number in the grouping of functional groups within an acknowledgment.  If the setting is false, only the GS02, GS03, GS07, and GS08 are used to determine a unique functional group for reporting in the acknowledgment.  If the setting is true, then the GS06 is also included in the determination.
Acknowledgments AckFunctionalGroups UseInputFunctionalGroup	true false*	Create a functional group in acknowledgment based on functional groups in input.
Tuning Paging	-P64:8*	Indicates the page size and page count to be used for the execution of compiled maps that will be executed by hipaa_data_compliance map with the RUN function.  The first number is Page Size, and the second number is page count.  The page size value may be an even integer between 2 and 1024. The page count may be an integer between 1 and 9999.  Note: In general, larger page sizes and count settings will improve overall map performance when input data transmission is large. Determining the optimal page settings may require some experimentation. This setting is not used for all run maps that are called.
Tuning Audit	-AEWU*	Indicates the audit log setting to be used for the execution of compiled maps that will be executed with the RUN function.
Tuning WorkFile	-WDU*	Indicates work file setting to be used for the execution of compiled maps that will be executed with the RUN function.

Parameter tag name	Values	Description
Tuning LengthOfDataElementInError	(User Specified numeric value between 1 and 99) 50*	Used for customizing the length of the value produced in the AK404/IK404 elements of the reporting acknowledgments.  Setting this to a value higher than the default may have a negative impact on performance. There may be cases where the entirety of the data in the file being validated is not placed on the acknowledgment even when this parameter is configured to its maximum value: When the length of data being reported exceeds the maximum parameter value or when the length of data exceeds more than twice the defined element length if the defined element length is > 32 characters.  The maximum size is 99, and the default size is 50.
Acknowledgments 824 Enabled	true false*	Indicates whether the results from the WEDI/SNIP Type 6 and 7 checks reported as a warning should be reported using the 824 transaction set.
Acknowledgments 824 WhenIfEnabled	Errors - Generates only when WEDI/SNIP type 6 or 7 warnings are present in the interchange.  Always* - Always generates even if WEDI/SNIP type 6 or 7 warning are not present in interchange.	Indicates when the results from the WEDI/SNIP checks should be reported using a HIPAA specific 824.
Acknowledgments 824 DetailMax	(User specified numeric value) 9999*	Indicates when we should stop producing detailed 824 warning information for a single transaction. If the number of warning lines, not total errors, to be produced in the 824 for the received transaction exceeds this limit, then the logic to generate the detailed information in the TED loop will be bypassed.  The setting for this value will determine the maximum number of error lines produced in the 824 for a single transaction.
GS08	Standard Industry	Standard - Use the X12 standard value (i.e., 005010) in the GS08 element of the 824.  Industry - Use the HIPAA or industry-specific value (e.g., 005010X186A1) in the GS08 element of the 824.
Acknowledgments 999 DetailMax	(User specified numeric value) 9999*	Note: To further reduce processing time on transactions with many errors, higher level validation may be skipped if the max is hit during Type 2 validation.

## HIPAA Compliance Check Parameters

Parameters used for the Compliance Check application are set in the compliance\_check\_5010\_paramater.dat file.

The settings in this file can be edited, and take effect the next time that the Compliance Check application is executed. There is no need to rebuild or re-deploy maps. This file contains data records that adhere to the convention of Tag\_Name="Value".

The following table describes the parameter tag names and values used by the Compliance Check application.

In the following table, the \* (asterisk) indicates the default value.

Parameter tag name	Values	Description
HIPAA_997_Enabled parameter	No - Never generate a HIPAA-specific 997 transaction even if the results of the compliance check indicate the presence of Type 1 errors.  Yes* - Always generate a HIPAA-specific 997 transaction for all interchanges, even if the compliance check does not indicate the presence of Type 1 errors.	Indicates whether the results from the WEDI/SNIP Type 1 check are reported by the HIPAA-specific 997 transaction set. This parameter is only available to be set when the highest level of HIPAA Validation selected is Type 1. If higher levels of HIPAA validations are selected, this parameter is not used.
HIPAA_Type_n_Checks_Enabled Note: Where n is an integer 1-4.	Yes - Perform Type n checks on the data.  No - Do not perform Type n checks on the data.	Used to indicate whether WEDI/SNIP Type n checks should be performed on the input data. See the HIPAA_Type_n_Checks_Enabled details documentation for more information about this parameter.
XML_Validation	Never* - Never validate the contents of the BIN segment.  XML - Validates the contents of the BIN segment for proper XML formatting.  Schema - Validate the contents of the BIN segment for proper XML formatting and against the schema.	Indicates the validation level for XML BIN/BDS segment content. The parameter is used to indicate whether the data in the BIN segment of the 4050 275 transaction set is to be validated and to which level it is to be validated.

Parameter tag name	Values	Description
HIPAA_Claim_Level_Rejection	N[O] - Reject the Entire transaction regardless of the location of the error  Y[ES] - Reject transaction set when all claims are in error or when an error is outside the Claim Loop.  A[ll] - Only reject transaction set when all claims are in error or when an error is outside the Provider loops. If the error is at the Provider, subscriber or patient level, all claims for that provider, subscriber or patient will be rejected.	Indicates whether the transaction set is still accepted with an AK501=E (errors accepted), and the claim, or claims, in error are removed from the valid data. This parameter is only available when the input HIPAA EDI data is a Claim 837.
HIPAA Claim Level 277CA Reporting	Never* - Never generate a 277CA even if the results of the compliance check indicate the presence of errors in 837 Claims input data.  Exceptions - Generate a 277CA for the interchange only when the results of the compliance check indicate the presence of errors in the 837 Claims input data.  Always - Always generate a 277CA for all interchanges even if the compliance check does not indicate the presence of errors in the 837 Claims input data.	Indicates whether the results from the Claim Level Rejection process are reported using the 5010 277CA transaction set.
HIPAA_Claim_Rejection_Level	Provider* - This option is recommended for performance. The 277CA reports details at the Provider level. This option does not provide specific details on each claim, only on the totals that are accepted and rejected for each Provider.  Claim - The 277CA reports details at the Claim level. This option provides specific details on each claim within the transaction. This option might impact performance if there are a large number of claims in the data.	Indicates the level at which the HIPAA 277 Claims Acknowledgment reports. It can show detail at the Provider level, or it can show details down to the Patient/Claim level.  This parameter has an impact only when the HIPAA Claim Level Rejection envelope parameter is set to Yes, and the input HIPAA EDI data is a Claim (transaction set 837). The default setting is Provider. If there are no errors in the HIPAA EDI 837 data, reporting goes to the Provider level regardless of the setting.
Duplicate_CLM_Check_Full_Segment	Y[ES] N[O]*	The Duplicate_CLM_Check_Full_Segment controls whether just CLMO1 or the full CLM segment is used to determine if claim-level rejection processing can continue for this transaction.  If the parameter is set to Y[ES] - Disable Claim level rejection if the entire CLM segment is not unique for all claims in the transaction.  Note: Only CLM01 is used to populate the TRN information in the 277CA, so the user may have to handle any ambiguity in the resultant acknowledgment if Y[ES] is used
277_REPORT_ERROR_STC	Y[ES] N[O]*	Set the sub elements in the STC01 element at all 277CA and 277DR loop levels to reflect if all valid or an error was detected at that level or below and include a valid entity code instead of the generic Ack/Receipt code.
HIPAA_Tack_Engaged	Yes - Generate No* - Do not generate.  Summary – Generate Translated Acknowledgement file without the Tack_EDI.html  Partial - Generate Translated Acknowledgment file and the Tack_EDI.html file, but the Tack_EDI.html file only contains the transactions in error.	Determines whether the Translated Acknowledgment Report is produced. This report transforms the acknowledgments into a readable report.
HIPAA_Tack_Accepted	Always - Generate accepted error report. Exceptions* - Generate error report only.	Determines what level of reporting appears on the Translated Acknowledgment Report. The options are to produce the report for all data (accepted and rejected) or to only generate the report based on the data in error. This setting only affects the content of the compliance_check_Tack.html file, not the Tack_EDI.html file.
Validation_Method	Map* Compliance  Both	Map - This default setting sends all data through the pass-through validation maps and only the invalid data goes through the subsequent processing. This is the recommended, and the default setting.  Compliance - This setting causes all data to be passed to the detailed processing done by the Compliance Check application.  Both - This setting causes all data to be processed by the pass through validation maps and the detailed compliance check validation. If there is a large amount of data, this option can slow the process. This option is used for troubleshooting purposes, logic modules, or both.

Parameter tag name	Values	Description
HIPAA_Relaxed_T2_Structure_Check_Enabled	Yes - Perform relaxed Type 2 structure checks if WEDI/SNIP Level 2 structure checking fails. No* - Do not perform relaxed Type 2 structure checks on the data.	This parameter indicates whether the relaxed structure will be performed on the input data if the data fails WEDI/SNIP Type 2 structure checking. This processing is performed only if errors are found during Type 2 structure checking.  If errors are not present, this processing is bypassed. Also, this new processing is only executed when Type 2 checks are enabled. If only Type 1 checking is enabled, the setting is ignored.
HIPAA_277CA_Uneque_BHT03	Y[ES] - Produce a BHT03 value based on the interchange, functional group, and transaction control numbers of the source 837 being validated.  N[O] - 277X2140001 is always set as a value for the BHT03 element.	If claim level rejection is enabled, this parameter controls the value created for the BHT03 Element in the 277CA output.
LoopId_In_Acknowledgment	N[O] - No value reported.  Y[ES] - Report Loop ID value (i.e. 2310).  A[LL] - Include loop id for Element level CTX segment and for errors found in structure validation if available.	This parameter indicates whether the loop identifier is required on the acknowledgment. (Will be reported in AK303 of 997 and IK303 in 999.)
Include_ISA13_In_Acks	Y[ES]  N[O]*	The Include_ISA13_In_Acks parameter is used when it is desired to use the ISA13 (control number) of the input transmission file for acknowledgments produced. If the parameter is set to Y[ES], the ISA13 value of the input transmission file will be used.  If the parameter is set to N[O], a default control number will be used.  Note: In the case of multiple interchanges in the input transmission, the ISA13 of the first interchange will be used, as only one interchange is produced in the acknowledgments.
Paging_Parameter	8*	This parameter specifies the page count used for the execution of most run maps from within the compliance check. The page count might be an integer 1 - 9999. The default is 8 pages.
Paging_Parameter	64*	This parameter specifies the page size (in Kb) to be used for the execution of run maps from within the compliance check. The page size might be an even integer 2 - 1024. The default setting is 64 Kb. In general, larger page sizes and count settings improve overall compliance check performance when processing transaction sets are large.
Allow_GS_Leading_0	Y[ES]  N[O]*	The Allow_GS_Leading_0 parameter indicates whether to allow leading zeros in the Functional Group control number in the GS and GE segments.  Allowable settings include:  Y[ES] - Allow leading zeros in group control numbers, such as 0001. N[O] - Do not allow leading zeros.
999_IK3_Missing_Seg	Y[ES]  N[O]*	For implementation dependent missing segments (I6), use Y[ES] to report the missing segment in the IK3. This will result in more information in the 999 by allowing the existing segment to be included in a CTX segment. Since the segment is missing, the segment position indicated will vary - it may be the position of the existing segment that caused the requirement, or the segment location in the transaction where it was determined the missing segment was not present, or the end of a loop.  Use N[O] to report the existing segment that caused the requirement for the missing segment in the IK3.

- [HIPAA\\_Relaxed\\_T2\\_Structure\\_Check\\_Enabled details](#)

The validation of the HIPAA EDI component Compliance Check module is accomplished by multiple validation passes.

## HIPAA\_Relaxed\_T2\_Structure\_Check\_Enabled details

The validation of the HIPAA EDI component Compliance Check module is accomplished by multiple validation passes.

The first step validates the data through a schema to determine which transactions are valid and which transactions are invalid. Invalid transactions are then checked using a more detailed process to determine where the errors occurred within the transaction.

The first part of the detailed validation process is to validate the data structure. The data must pass basic HIPAA structure validation to determine and add loop identifiers in the data. If the data is not well structured and fails this process, the loop identification cannot be determined, and subsequent steps in the complete validation process do not execute. In order to report more errors, a relaxed structure check option has been added to the module to lessen the structure check requirements and to report more errors in such circumstances.

- [Relaxed structure check option](#)

The relaxed structure check option produces more errors, both structural and those related to HIPAA compliance, in a single pass rather than requiring two complete executions of the module.

- [R2 records](#)

Each transaction set in the hipaa\_x12\_structure.dat file has an additional structure record with a validation type of R2 (relaxed Type 2).

## Relaxed structure check option

The relaxed structure check option produces more errors, both structural and those related to HIPAA compliance, in a single pass rather than requiring two complete executions of the module.

Be aware of these points when using the relaxed structure check option:

- The Allowed Relaxed Type 2 Structure Checking in the Transformation Extender UI sets the HIPAA\_Relaxed\_T2\_Structure\_Checking parameter to allow for relaxed level 2 validation, if enabled. The default value for this setting is No.
- If the Allowed Relaxed Type 2 Structure Checking option is enabled, then relaxed structure checking is performed on the input data only if the data fails WEDI/SNIP type 2 structure checking. If errors are not present, this processing is bypassed. Also, this new processing is only executed when Type 2 checks are enabled. If only Type 1 checking is enabled, the setting is ignored.
- In order to implement the relaxed structure check option, a new set of information has been added to enable this processing called, R2 records.

---

## R2 records

Each transaction set in the hipaa\_x12\_structure.dat file has an additional structure record with a validation type of R2 (relaxed Type 2).

These rules are applied to the R2 records:

- The segments that define loops do not change. These segments must be kept intact to determine the correct loop IDs. This includes the segments: HL, NM1, N1, and ENT. If data fails both T2 (standard WEDI/SNIP type 2) and R2 (relaxed Type 2) structure checking, validation is stopped and only the structural errors are reported on the acknowledgment.
- The qualifier is removed from these records, making the checking relaxed for those segments that are used for choice and unordered loops. This includes instances of the following segments: REF, DTP, AMT, CRC, DTM, HI, III, NTE, PER, QTY, PWK. If the qualifier is not checked, any segments with invalid, or too many qualifiers, continues in the validation process.
- The cardinality of any segment that is not a loop-defining segment, is relaxed to match the cardinality of standard X12 (Type 1) validation, except in the cases where Type 2 is more relaxed than Type 1.
- The results of the first Type 2 structure check are reported along with the results of the relaxed structure check. This results in duplicate reporting, but does allow all of the errors to be reported in one pass of the compliance check application.

Any segment that is not allowed for use under HIPAA is not allowed in the relaxed validation and produces an error if encountered.

If the data fails at both standard WEDI/SNIP type 2 and relaxed type 2 structure checking, then validation stops and only the errors as reported in the structure checking are reported.

---

## Claim Level Rejection

The Pack for Healthcare allows invalid claims to be rejected and valid claims to be processed. Validation is run on the document level (claim) for the 5010 837 Claims Transactions.

Claim Level Rejection allows a transaction *not* to be rejected on the basis of one bad claim. Claim Level Rejection places a bad claim, or claims, in a separate transaction while leaving good claims in the original transaction.

Claim Level Rejection is not available for the Post Adjudicated Claims Data Reporting (PACDR) 837 transactions.

Claim Level Rejection rejects only those claims in error when it can absolutely be determined that the error is due to the claim in question. Many X12 or HIPAA compliance errors can force the entire transaction to be rejected. For example, HIPAA type 1 and type 2 structure errors are nonrecoverable errors which cause the entire transaction to be rejected. A Claim Level Rejection is not possible for these nonrecoverable errors.

- **Assumptions**

Claim Level Rejection is only provided for the 837 Professional, Dental, and Institutional claims transactions. It is available for 5010 versions of the claims transactions. It is ignored if the input HIPAA EDI data is not a claim transaction.

- **Acknowledgements**

If a transaction is structurally sound, and all errors within the transaction occur at the claim level or below, then the claim or claims in error are rejected, and the valid claims are accepted. A 997 and/or 999 indicating acceptance with errors is generated for the transaction with the value E for the Transaction Acknowledgement Code, AK501/IK501.

- **Configuration process**

Claim Level Rejection is optional and configurable.

---

## Assumptions

Claim Level Rejection is only provided for the 837 Professional, Dental, and Institutional claims transactions. It is available for 5010 versions of the claims transactions. It is ignored if the input HIPAA EDI data is not a claim transaction.

Assumptions about the Claim Level Rejection parameter are described here.

- When healthcare claims transactions are processed, the HIPAA Guidelines recommend that the value in CLM01 be unique for each claim. Although this is not a HIPAA requirement, it can be used as a means for claims to be uniquely identified and reported in an automated system. However, there are cases where Claims with the same CLM01 could be present in the input transmission being validated. One scenario is when an original and voided claim is present. In this case, the Claim Frequency Code in CLM05-03 would be different (1 and 8), but CLM01 would be the same. Therefore, Compliance Checking requires the entire CLM segment to be unique in reporting results at a claim rejection level. Suppose the concatenated value of all CLM elements is not unique. In that case, Claim Level Rejection is terminated for the transaction with the duplicate CLM segments, and a message will appear in the compliance check summary file that indicates that the Claim Level Rejection process has been terminated.
- If a transaction fails any level of structure checking, then the entire transaction set is rejected. This is the same behavior that was exhibited before Claim Level Rejection was introduced. To successfully run Claim Level Rejection, the loop id must be available for inspection. If the data fails the structure checking step, the loop id is not determinable.

- If a transaction is structurally sound, and all errors within the transaction occur at the claim level or below, then the claim, or claims in error, are rejected, and the valid claims are written to the valid file.
- If any error occurs at a level higher than the provider in the transaction set, the entire transaction is rejected. For the Claim Level Rejection process to extract any rejected claims, all of the errors in the transaction set must occur at the claim level or lower.
- The Claim Level Rejection parameter is configurable and optional. The default parameter setting is to not execute the Claim Level Rejection process.
- All valid claims are reformed into a new transaction with the EDI structures adjusted so that the transaction passes HIPAA compliance to the validation level requested. These valid claims are listed in the compliance\_check\_valid.out file.
- Rejected claims are reformed into a new transaction with the EDI structures adjusted so that the transaction only fails HIPAA compliance for the same reason or reasons as the original claim rejection. Rejected claims are listed in the compliance\_check\_invalid.out file.

## Acknowledgements

If a transaction is structurally sound, and all errors within the transaction occur at the claim level or below, then the claim or claims in error are rejected, and the valid claims are accepted. A 997 and/or 999 indicating acceptance with errors is generated for the transaction with the value E for the Transaction Acknowledgement Code, AK501/IK501.

If the AK501/IK501 is showing a status of "E", it indicates that the transaction set has claim level errors, but if there is only 1 transaction set in the Functional Group, then the AK901 status is set to "E" (Accepted, but errors noted).

The results of Claim Level Rejection are reported in the 5010X214 277 Claims Acknowledgement, 277CA or 005010X364 277DRA. By default transaction details are reported at the Provider Level. This behavior can be changed to report at the claim level if the **Clm\_Lev\_Rej\_Level** parameter is set to **C** to indicate the claim.

Note: The 227 is not generated for any non-837 input data.

If there are no errors in the HIPAA EDI 837 data, then 277 reporting only goes to the Provider Level if the **Clm\_Lev\_Rej\_Level** parameter is set to C. The results of the 277 Report do not appear in the Translated Acknowledgement Report.

If there are no errors and the **Clm\_Lev\_Rej\_Level** parameter setting is A, the 277 reporting will include patient and claim details.

## Configuration process

Claim Level Rejection is optional and configurable.

The following parameters are contained in the parameter file to control the processing.

- **Claim\_Level\_Rejection**
- **Clm\_Lev\_Rej\_Level**
- **HIPAA\_277CA\_Enabled**
- **277\_REPORT\_ERROR\_STC**

See the Compliance check parameter file documentation for details about these parameters. The same configuration is used for both 277CA and 277DRA. If the input is a standard HIPAA mandated claim, a 277CA will be produced if so configured. If the input is one of the PACDR claims, a 277DRA will be produced.

- [Enable Claim Level Rejection](#)
- [Performance tuning and debugging](#)

## Enable Claim Level Rejection

To enable Claim Level Rejection, perform the following activities.

- Set the **Claim\_Level\_Rejection** parameter to Y (the default is N). Ensure that the semicolon before the parameter is removed.
- To produce the 277CA, set the **HIPAA\_277CA\_Enabled** parameter to A or E (the default is N). If the **Claim\_Level\_Rejection** parameter is not set to Y, then the 277CA parameter is set to NEVER at map execution time. This parameter only applies to HIPAA EDI 837 input data. The 277CA is not generated for any non-837 data.
- When the 277CA is produced, the reporting level is at the Provider Level. This can be modified to report at the Claim Level when there are errors, by changing the **Clm\_Lev\_Rej\_Level** to C. Changing this parameter might have an impact on performance if there are many claims in a transaction.

## Performance tuning and debugging

To optimize system performance, it is recommended that Claim-Level Rejection is not enabled unless required by your organization.

If you are concerned about performance and the 277CA is enabled, the reporting level should be left at the Provider level. Changing it to report at the Claim-level might have a significant impact on performance. The size of the output acknowledgement file can also become very large if there are many claims per transaction report.

The results from the Claim Level Rejection process can be found in the **compliance\_check\_summary.out** file, which is output card #2 of the main compliance check map. The results from each Claim Level Rejection run map are available to be reviewed if any issues occur during execution of the compliance check application.

## Configurable Rules

The Pack for Healthcare allows you to enable or disable WEDI/SNIP extended Level 2 through Level 7, as defined in the Compliance Check module.

With the HIPAA data compliance application this can be done at run time with the use of the hipaa\_rules\_to\_change.json input file, which controls which rules are enabled or disabled differently from their default setting.

With the Compliance Check application, the Configurable Rules application, which requires the Pack for Healthcare to be installed on Design Studio, allows you to determine which rules are enabled and disabled for validation. You can then generate the schemas and the qualifier file for use in the Compliance Check application.

- [Enabling Rules with HIPAA DATA compliance](#)

The HIPAA Data compliance allows rules to be enabled or disabled at run time.

- [Enabling Rules with HIPAA Compliance Check](#)

With the Compliance Check application, the Configurable Rules application, allows you to determine which rules are enabled and disabled for validation, and generates the corresponding updated schemas and qualifier file to use with Compliance Check.

- [Configurable rules objects](#)

Configurable Rules application objects consist of data files (.dat), schemas, a type tree script file (.mts), a map source file, batch command files (.bat), and executable map (.mmc) files.

- [Configurable rules reporting](#)

The Configurable Rules report is generated during configurable rules processing and can be used to check the integrity of the customized mini qualifier file. The report that is created can be found in the data directory under configurable\_rules and is called qualifier\_report.html.

- [Basic steps for running the Configurable Rules application](#)

---

## Enabling Rules with HIPAA DATA compliance

The HIPAA Data compliance allows rules to be enabled or disabled at run time.

### Before you begin

---

Objects

- Hipaa\_x12\_qualifier.json – contains all the extended type 2, type 3-7 rules that are configurable, with all the details that are needed for each rule for the hipaa\_data\_compliance application to enforce. This should not be modified by the user except to add user-supported (not product-supported) Type 7 rules – more details on that later.
- Hipaa\_rules\_to\_change.json – this contains ONLY the rules that you wish to DISABLE that are ENABLED by default in the product or that you wish to ENABLE that is DISABLED in the product hipaa\_x12\_qualifier.json file.
- Hipaa\_x12\_rules.json – Contains all the rules in hipaa\_x12\_qualifier.json, but only with a minimum number of attributes, not all the details.
- create\_rules\_to\_change utility map. In Pack for Healthcare Design Studio installs, this is included in the hipaa\_data\_utilities source map in the map directory. This is included in the design server of the hipaa\_data\_compliance project.

Steps to enable or disable rules:

### Procedure

---

1. Make a backup copy of Hipaa\_x12\_rules.json and the hipaa\_rules\_to\_change.json.
2. In the full hipaa\_x12\_rules.json, update the OnOffIndicator to 1 for any rule you need to enable (not done in this example) and to 0 for any rule you need to disable (disable the first rule).
3. Build the create\_rules\_to\_change utility map.
4. Run the create\_rules\_to\_change utility map to generate a hipaa\_rules\_to\_change.json.  
This map uses the hipaa\_x12\_rules.json and the hipaa\_x12\_qualifier.json to generate the hipaa\_rules\_to\_change.json. This will now include the rules configured differently than the default in the hipaa\_x12\_qualifier.json file.

---

## Enabling Rules with HIPAA Compliance Check

With the Compliance Check application, the Configurable Rules application, allows you to determine which rules are enabled and disabled for validation, and generates the corresponding updated schemas and qualifier file to use with Compliance Check.

The parameter file allows the application to be run in one of several ways. A validation method of map (Validation\_Method="M") allows the data to be passed against the schema. It then sends only invalid data through the full compliance check process. This method saves processing time and resources, especially when processing data that is predominately good. As a result, most rules that are enforced for HIPAA must be defined two times. The rules are defined once during schema checking and again in the Compliance Check Rules system.

---

## Configurable rules objects

Configurable Rules application objects consist of data files (.dat), schemas, a type tree script file (.mts), a map source file, batch command files (.bat), and executable map (.mmc) files.

Because the objects listed here are used in the Configurable Rules application, changes made to them might result in inconsistent operation or might cause them to stop working completely. For this reason, do not modify Configurable Rules application objects. In the list provided here, only those files denoted with an asterisk (\*) after the file name can be safely modified.

- **Data files:**

- configurable\_rules\_mini\_qualifier.dat \*
- configurable\_rules\_component\_rules\_only.dat
- plus.gif
- minus.gif

- qualifier\_rules\_report.css
  - directory\_paths.dat
- **Schemas:**
    - t3t4\_utility
    - rule\_less
  - **Schemas script file:**
    - hipaa\_x12\_master\_rule\_stub.mts
  - **Map source file:**
    - configurable\_rules
  - **Batch command files:**
    - import\_trees.bat
    - configurable\_rules.bat
  - **Executable map files:**
    - create\_env\_check\_ruleless\_tree
    - separate\_type\_2\_tree
    - remove\_first\_cr
    - create\_rule\_less\_tree
    - separate\_tree
    - remove\_comments
    - hipaa\_type\_2
    - create\_new\_tree
  - **Data files**
  - **Schemas**

These schemas are used in the Global Configurable Rules application.
  - **Schemas script file**

The hipaa\_x12\_master\_rule\_stub.mts schema script file is updated by the configurable rules application.
  - **Map source file**
  - **Configurable Rules batch command files**
  - **Compliance Check application qualifier file**

The hipaa\_x12\_qualifier.dat file is the qualifier file that is used by the Compliance Check application. It contains a list of rules used for performing Type 2, Type 3 and Type 4 WEDI/SNIP validations.
  - **Executable map files**
- 

## Data files

The Global Configurable Rules application includes the following files. Most of these files are required as input to the map.

- **configurable\_rules\_component\_rules\_only.dat**

The configurable\_rules\_component\_rules\_only.dat file contains the schema component rules for all of the rules in the mini qualifier file.
  - **configurable\_rules\_mini\_qualifier.dat**

The configurable\_rules\_mini\_qualifier.dat file contains a listing of all of the rules and validations that can be configured in the Pack for Healthcare HIPAA EDI component.
  - **configurable\_rules\_type\_5\_mini\_qualifier.dat**
  - **minus.gif**

The minus.gif file is used to visually display a - (minus) on the report which indicates that the data can be closed.
  - **plus.gif**

The plus.gif file visually displays a + (plus) on the report, which indicates that the data can be opened.
  - **qualifier\_rules\_report.css**

The qualifier\_rules\_report.css file is a cascading style sheet which allows you to modify colors, fonts, margins, scroll regions and other display related features.
  - **directory\_paths.dat**

The directory\_paths.dat file contains the directory paths for the location of map executable files. The map executable files are used to import the schemas created by the Configurable Rules application.
- 

## configurable\_rules\_component\_rules\_only.dat

The configurable\_rules\_component\_rules\_only.dat file contains the schema component rules for all of the rules in the mini qualifier file.

This file installs within the imported project.

This file is used by the Configurable Rules application to merge the component rules for the validations that are enabled in the mini-qualifier file.

Do not modify this file.

---

## configurable\_rules\_mini\_qualifier.dat

The configurable\_rules\_mini\_qualifier.dat file contains a listing of all of the rules and validations that can be configured in the Pack for Healthcare HIPAA EDI component.

This file installs within the imported project.

Each record is delimited with a ^, and terminated with a ~<NL>.

Each record in the file constitutes an extended Type 2, Type 3, or Type 4 WEDI/SNIP level validation with the following layout:

- Rule Sequence #
- Version / Release / Identifier Code
- Transaction Set ID
- Level of Validation
- ON/OFF Indicator. This is the only information that can be modified in this file:
  - 1 = ON
  - 0 = OFF
- Rule Description Text

This is the only file in the Configurable Rules application that can be modified.

---

## configurable\_rules\_type\_5\_mini\_qualifier.dat

The configurable\_rules\_type\_5\_mini\_qualifier.dat file contains a list of all of the type 5 validations that can be configured by the Standard Processing Engine. These rules cannot be configured in the Pack for Healthcare. Do not modify the file.

---

## minus.gif

The minus.gif file is used to visually display a - (minus) on the report which indicates that the data can be closed.

This file installs within the imported project.

---

## plus.gif

The plus.gif file visually displays a + (plus) on the report, which indicates that the data can be opened.

This file installs within the imported project.

---

## qualifier\_rules\_report.css

The qualifier\_rules\_report.css file is a cascading style sheet which allows you to modify colors, fonts, margins, scroll regions and other display related features.

This file installs within the imported project.

---

## directory\_paths.dat

The directory\_paths.dat file contains the directory paths for the location of map executable files. The map executable files are used to import the schemas created by the Configurable Rules application.

This file installs within the imported project.

---

## Schemas

These schemas are used in the Global Configurable Rules application.

- [t3t4\\_utility schema](#)  
The t3t4\_utility schema is used by the executable maps in the configurable\_rules map source file.
- [Rule\\_less schema](#)  
The Rule\_less schema is based on the ttmaker60 schema, which is included in the examples directory in the Transformation Extender base product installation.

---

## t3t4\_utility schema

The t3t4\_utility schema is used by the executable maps in the configurable\_rules map source file.

The schema contains the following definitions:

- Configurable Rules results
- DirectoryPaths file
- Type Tree Component Rule

- Qualifier Report

This schema installs within the imported project.

## Rule\_less schema

The Rule\_less schema is based on the ttmaker60 schema, which is included in the examples directory in the Transformation Extender base product installation.

The Rule\_less schema contains the definition for the schema script files used in the configurable rules.

## Schemas script file

The hipaa\_x12\_master\_rule\_stub.mts schema script file is updated by the configurable rules application.

- [hipaa\\_x12\\_master\\_rule\\_stub.mts schema script file](#)

## hipaa\_x12\_master\_rule\_stub.mts schema script file

If the hipaa\_x12\_master\_rule\_stub.mts file is imported, it is missing all of the component rules that are defined as configurable, and only the rule description for a given rule appears in their place.

This file installs within the imported project.

Do not modify this file.

## Map source file

The configurable\_rules map source file included with the Configurable Rules application contains several executable maps. All of the maps are described in the documentation provided here. The configurable rules maps are used by this application. Modification of these maps might cause inconsistent operation, or might cause them to fail completely.

The configurable\_rules map source file installs within the imported project.

You must build all of the executable maps to create the compiled map files (\*.mmc) used by the RUN function calls before you run the configurable\_rules.mmc executable map.

- [Description of executable maps](#)

The Configurable Rules application executable maps are described in the table provided here.

## Description of executable maps

The Configurable Rules application executable maps are described in the table provided here.

Executable map	Output card	Description
configurablerules	1	Top-level map for the configurable_rules application. This map keeps track of the results for all steps and generates output. The output defaults to the Sink Adapter.
	2	The updated qualifier file, hipaa_x12_qualifier_updated.dat is renamed and can be reused in the Transformation Extender, stand-alone Compliance Check application.
	3	MergedRules output contains the enabled business rules and their associated component rules. The output defaults to the Sink Adapter.
	4	The qualifier_report.html file is a report that shows the results, including errors, of the Configurable Rules application.
createminifiles	1-3	Invoked from the configurable_rules map. This map reads in the Transformation Extender exported business rules and generates the mini qualifier of the Type 2 - 6 enabled business rules. The results are sent back to the main configurable rules map.
	4-6	These output cards contain the component rules taken from the Transformation Extender business rules export file for all Type 2 - 6 enabled business rules. The results are sent to the main configurable rules map.
	7-9	Generation of the qualifier report. The results are sent to the main configurable rules map.
qualifierreport	1	Invoked from the configurable rules map, this map checks the mini qualifier, qualifier, and component rules only files for content and integrity. It also reports rules that have had their status modified. The output contains potential errors from the input file.
	2	If there are no errors in output card #1, the files are reviewed to determine the differences in the enabled state of each rule.
	3	Sorts the rules from output card #2.
	4	Determines all of the enabled rules as found in the Transformation Extender business rules export data.
	5	Sorts the business rules by rule sequence number
updatemaster	4	Produces the report of the results from output cards #1, #2, and #4.
	1	Invoked from configurable_rules, this map updates the schema script file to contain the component rules for all of the Type 2 - 6 business rules that were enabled in the Transformation Extender exported business rules.

<b>Executable map</b>	<b>Output card</b>	<b>Description</b>
updateet3t4componentrules	1	Invoked from updatemaster and executed once per component rule. This map updates the enabled component rule in the schema script file.
	2	Determines if BINDABS functionality is required for the component rule.
	3	Strips off unused ampersands (&) from the component rule.
	4	Strips off unused tags from the component rule.

## Configurable Rules batch command files

- **[import\\_trees.bat](#)**  
The import\_trees.bat file is a batch file that imports the schema script files that are generated by the Global Configurable rules maps.
- **[configurable\\_rules.bat](#)**  
The configurable\_rules.bat file is a batch file that can run the entire Global Configurable Rules application. It also includes the commands to import the type tree script files generated.

### import\_trees.bat

The import\_trees.bat file is a batch file that imports the schema script files that are generated by the Global Configurable rules maps.

### configurable\_rules.bat

The configurable\_rules.bat file is a batch file that can run the entire Global Configurable Rules application. It also includes the commands to import the type tree script files generated.

This file installs within the imported project.

## Compliance Check application qualifier file

The hipaa\_x12\_qualifier.dat file is the qualifier file that is used by the Compliance Check application. It contains a list of rules used for performing Type 2, Type 3 and Type 4 WEDI/SNIP validations.

This file installs within the imported project.

## Executable map files

The executable map files install with the imported project.

These maps start by validating the schema script file generated by the Global Configurable Rules application.

The maps then generate each of the HIPAA schemas that are included in the HIPAA EDI component:

- [hipaa\\_x12\\_type\\_2](#)
- [hipaa\\_x12](#)
- [hipaa\\_x12\\_ruleless](#)

## Configurable rules reporting

The Configurable Rules report is generated during configurable rules processing and can be used to check the integrity of the customized mini qualifier file. The report that is created can be found in the data directory under configurable\_rules and is called qualifier\_report.html.

The report contains these sections:

- Validation results
- Modified rules
- Unchanged rules
- Enabled non-configured rules
- **[Validation results](#)**  
The validation map, modifyqualifier.mmc, checks the basic integrity of all records in the configurable\_rules\_mini\_qualifier.dat file. Basic validation is performed on fields and error text fields that exceed 60 characters.
- **[Modified rules](#)**  
Modified rules are listed in this section. The count identified, is the count of the number of modified rules relative to the hipaa\_x12\_qualifier.dat file.
- **[Unchanged rules](#)**  
Some configurable rules have no modifications.

- [Enabled non-configured rules](#)

Rules that are enabled but not configurable are located in this section.

- [Configurable rules assumptions](#)

---

## Validation results

The validation map, modifyqualifier.mmc, checks the basic integrity of all records in the configurable\_rules\_mini\_qualifier.dat file. Basic validation is performed on fields and error text fields that exceed 60 characters.

Some rules are dependent on other rules, and this dependency is verified to ensure that the ON/OFF flags match. If one rule is enabled, but it has a dependency on another rule that is disabled, this is reported and, processing stops. The ON/OFF flags must match. This means that both must be on, or both must be off.

As part of the validation process, a flat file, Errors.out, is generated. This file contains a list of verified dependent rules and can be found in the configurable\_rules\maps directory.

Additional checking for duplicate rules is performed. Both the rule numbers and the entire rule contents are checked for duplication. A warning message is in turn, generated for any rule found in the customized qualifier file that has not been defined for the compliance check application. Rules that fail validation checks are reported in this section, along with the total number of records found in the customized qualifier file, configurable\_rules\_mini\_qualifier.dat. The total number of records that fail the validation steps is also reported.

---

## Modified rules

Modified rules are listed in this section. The count identified, is the count of the number of modified rules relative to the hipaa\_x12\_qualifier.dat file.

The rules are sorted based on version id. The content of each modified rule is identified. The original value for the rule is identified with the notation:  
<<< ORIGINAL VALUE >>>

---

## Unchanged rules

Some configurable rules have no modifications.

The count that is identified, is the count of the number of unchanged rules relative to the hipaa\_x12\_qualifier.dat file. The rules are sorted based on version id. The content of each rule is identified.

---

## Enabled non-configured rules

Rules that are enabled but not configurable are located in this section.

The count identified, is the count of the number of modified rules relative to the hipaa\_x12\_qualifier.dat file. The rules are sorted based on version id. The content of each rule is identified.

---

## Configurable rules assumptions

The following assumptions are part of the first phase of the Configurable Rules application:

- Configurable rules are only available for the HIPAA 5010 version extended level 2, 3, and 4 validations that are found in the mini qualifier file: configurable\_rules\_mini\_qualifier.dat
- Validations for 4050 version transactions cannot be configured.
- Only extended level 2 validations found in the qualifier file are configurable. Standard level 2 validations not found in the qualifier file are not configurable.
- There is no ability to configure any level 6 or 7 validations.
- There are some validations that are dependent on each other. These rules must be either both enabled or both disabled for in the mini qualifier file for Configurable Rules to execute successfully. A full listing of these rules is available on the Configurable Rules report (qualifier\_report.html).
- The only file that is available for users to update is the configurable\_rules\_mini\_qualifier.dat file.  
If this file is not updated correctly, the html error report indicates the nature of the problem and processing stops. The schemas and qualifier files have not been updated.
- The application is only available for execution on Windows.
- The application can be launched from either the Design Server or by using the Command Server. The batch files that are included with the installation can be used to execute the application using the Command Server.
- The Configurable Rules application only needs to be used if you have modified the mini-qualifier file. If you do not modify that file, running the Configurable Rules application is unnecessary.

---

## Basic steps for running the Configurable Rules application

The following are the basic steps required to run the Configurable Rules application. For detailed instructions on how to run the application, see [Instructions for running the Configurable rules application](#).

1. Determine which extended level 2, 3, and 4 rules must be changed from their default on and off states.
2. Back up the qualifier data file and the schemas to a safe location.
3. Modify the mini-qualifier file to change the state of the on and off indicator for each rule needed.
4. Run the configurable\_rules map.
5. Verify the outputs (results).
6. Run the batch files to generate the schemas using the Command Server, or import the schemas using the Design Server.
7. Rename the updated qualifier file.
8. Copy the renamed qualifier file to the main HIPAA\data directory.
9. Rebuild all the maps in the Compliance Check application before executing with the updated components.

Continue with the detailed steps provided here to run the Configurable Rules application.

- [Instructions for running the Configurable rules application](#)

---

## Instructions for running the Configurable rules application

These instructions describe how to run the Configurable Rules application.

1. Determine which extended level 2, 3, and level 4 rules must be changed from their default on and off states. These defaults might not have to be changed, but they must be changed if either of the following conditions occur:
  - If an unwanted rule failure exists on an acknowledgement. Note the rule number and investigate. After investigation, it might be determined that validation might not be required for that rule.
  - If an expected failure is not observed upon acknowledgement. If an investigation determines that the rule is in the mini qualifier file, but has not been enabled.
2. Back up installed files to a safe location. During the process of configuring the rules, a new qualifier data file and a new set of HIPAA EDI schemas are generated. Save these files in a safe location:

```
install_dir\packs_healthcare_vn.n.n\hipaa\configurable_rules\data\
 • configurable_rules_mini_qualifier.dat
install_dir\packs_healthcare_vn.n.n\hipaa\data\
 • hipaa_x12_qualifier.dat
install_dir\packs_healthcare_vn.n.n\hipaa\trees\
 • hipaa_x12.mtt
 • hipaa_x12_type_2.mtt
 • hipaa_x12_ruleless.mtt
```

If the Configurable Rules application is run multiple times, save both of the out-of-the-box pack files, as well as interim versions of the files created.

3. Modify the mini qualifier file to change the on and off states of the rules that are identified in step 1 of this procedure. To change the usage of the rules, you must edit the configurable\_rules\_mini\_qualifier.dat file at this directory location:

```
install_dir\packs_healthcare_vn.n.n\hipaa\configurable_rules\data\
```

The configurable\_rules\_mini\_qualifier.dat file is organized as follows:

- One rule (record) per line, delimited by "^."
- Field 1: Rule number, internal Transformation Extender reference
- Field 2: Transaction version reference, matching the GS08.
- Field 3: Transaction ID
- Field 4: WEDI/SNIP validation level
- Field 5: ON/OFF indicator. 0 (zero) indicates off, and 1 (one) indicates on. This is the only information that can be changed in this file. Save the file after changes are made.
- Field 6: Description of Rule

4. Run the configurable rules map file, configurable\_rules.mms, in the map source. This file is located at this directory location:

```
install_dir\packs\healthcare_vn.n.n\hipaa\configurable_rules\maps\
```

The executable map is configurable\_rules. Run the map as follows:

- a. Open configurable\_rules.mms in the Design Server.
- b. Build all of the maps.
- c. Run the configurable\_rules map.
- d. Verify the results by viewing the qualifier\_report.html file at this directory location:  

```
install_dir\packs\healthcare_vn.n.n\hipaa\configurable_rules\data\
```

5. Verify the results and ensure that the type tree script files listed in this step are created in this directory:

```
install_dir\packs\healthcare_vn.n.n\hipaa\configurable_rules\trees
 • hipaa_x12.mts
 • hipaa_x12_type_2.mts
 • hipaa_x12_ruleless.mts
 • hipaa_x12_master_rule_updated.mts
```

6. If you are running the Command Server, execute the import\_trees.bat file. This file imports the type trees that are required by the Compliance Check application. It is located in the following directory:

```
install_dir\packs\healthcare_vn.n.n\hipaa\configurable_rules\master
```

This file can be executed by using a command prompt or by double-clicking on the file name in the Windows directory. Verify that the files listed in this step are generated in this directory:

```
install_dir\packs\healthcare_vn.n.n\hipaa\configurable_rules\trees
```

- hipaa\_x12.mtt
  - hipaa\_x12\_type\_2.mtt
  - hipaa\_x12\_ruleless.mtt
7. If you are not running the Command Server, the type trees must be imported one at a time using the Design Server. See the Design Server documentation for instructions on importing type trees.
8. Move the type trees generated in Step 7 of this procedure to the following directory:  
`install_dir\packs\healthcare_vn.n.n\hipaa\trees\`
9. Rename the `hipaa_x12_qualifier_updated.dat` file, which is located in the directory to `hipaa_x12_qualifier.dat`. Move the renamed file to:  
`install_dir\packs\healthcare_vn.n.n\hipaa\data`
10. Rebuild all of the executable maps in the `compliance_check.mms` file.
- 

## Type 5 Validation

This support is only available with the HIPAA Data Compliance application.

### Overview

Type 5 Validation can be defined as the process of confirming that HIPAA X12 transaction data content complies with the requirements of the External Code Sources as documented in the Implementation Guides.

Each HIPAA X12 Implementation Guide contains descriptive information about External Code Sources in an appendix. This, typically, is Appendix A of the TR3. This information generally consists of the code source name and identifier, contact information, and an abstract of the related code set contents for each External Code Source referenced in the document.

Specifics about the data content to be validated using External Code Sources are embedded throughout the Implementation Guides, primarily in the Segment Detail portion of the guides. Typically, these External Code Source references are placed at either the data element or qualifier data element level. They include the Code Source identifier followed by the name. In the Pack for Healthcare, the following principles have been applied to distinguish between Type 5 and other validation types in the WEDI/SNIP model.

The validation of a data element against a set of one or more code values that are explicitly listed in the HIPAA Implementation Guide is considered a Type 2 check rather than a Type 5 check. For example, the entry for the BGN05 Time Zone Code element in the 834 Benefit Enrollment and Maintenance document references Code Source 94: International Organization for Standardization (Date and Time). However, the External Code Source reference is followed by an explicit list of values. The validation of this element against the list of permitted values is implemented as a Type 2 check.

The validation of a data element against a set of code values that are referenced in the HIPAA Implementation Guide is considered a Type 5 check. For example, the entry for N402 (State or Province Code) may reference CODE SOURCE 22: States and Provinces. When this reference is made and no specific states or provinces are listed in the implementation guide then this is considered to be a Type 5 check.

Type 5 Validation checks for conformance with published code sets that apply globally to all HIPAA X12 transactions processed. Any partner-specific external code set validation is considered to be beyond the scope of WEDI/SNIP Type 5.

- **Type 5 Validation Methods**

The Pack for Healthcare supports two methods of Type 5 Validation: Code Value and Pattern Match. The choice of validation method is generally determined by the requirements of your organization, the nature of the code list, and access to code list contents.

- **Configuration**

Database requirements

- **Code list identifiers table**

This table lists the unique code list identifiers used by the Pack for Healthcare along with their associated external code source, or sources. The table also provides information about the contents of the code list.

---

## Type 5 Validation Methods

The Pack for Healthcare supports two methods of Type 5 Validation: Code Value and Pattern Match. The choice of validation method is generally determined by the requirements of your organization, the nature of the code list, and access to code list contents.

The `hipaa_X12_qualifier.json` file contains Type 5 rules that can be checked. These rules will have a validation type of T5, with the syntax "ValidationType": "T5." Each Type 5 rule also has additional attributes to indicate what method of Type 5 validation to use, which code table to reference, or which pattern to match.

The Code Value method, with "ValidationSubType": "CV", checks that the data content selected for Type 5 validation is in a particular code list. The Code Table ID, or "CodeTableID" is used to specify the Code List collection that contains the list of all acceptable code items. An example is "CodeTableID": "T5\_022". This method also includes the option to define start and/or end dates for individual code list items. The transaction processing date determines whether the code item falls within the effective date restrictions.

The Pattern Match method, with "ValidationSubType": "PM", checks that the data content is consistent with a particular format or pattern. The Pattern Identifier, or "PatternID", is used to specify the Regular Expression to be used for validation. A Regular Expression, or regex for short, is a codified text string describing a search pattern. Examples of Pattern IDs that are used in the Pack include:

- [0-9A-Z]{2} Exactly two uppercase alphabetic or numeric characters.
- [0-9]{9} Exactly nine numeric characters.
- [A-Z]{3,5} Between three and five uppercase alphabetic characters.

If the Type 5 validation is enabled, all elements that contain an HIPAA National Provider Identifier (NPI) have specific rules to validate:

- The first digit of the NPI must be either 1 or 2.
- The check digit(last digit) is correct, based on the assumed prefix of 80840.

The rules of this element include the PatternID of NPI.

You can calculate the NPI check digit using the Luhn formula, a method for computing the modulus 10 double-add-double check digit:

1. Double the value of alternate digits, start from the rightmost digit.
2. Add a constant of 24 to account for the 80840 prefix present on a card issuer identifier, in addition to the individual digits of products of doubling, and the unaffected digits.
3. Subtract the result from the next higher number that ends in zero.

## Configuration

### Database requirements

The Pack for Healthcare does not come with a packaged database application. To use the Code Value Type 5 validation, you must have a Database server accessible that has MongoDB interface or JDBC support for queries and load.

The maps provided in the External Code List Utilities example, the primary map being load\_all\_code\_sets, can be run to load the code set collections with the documents that represent the list of allowed code values for that code set. There may be code sets that are updated between Pack releases, and the values need to be updated in the database. Updates for individual code values, to add/remove values, or to add a start date or end date can be done by reloading a full updated code list in the JSON format that the utility map uses or using any of the UI tooling supported by your database server.

The Pack for Healthcare does not provide JSON files to load all the code sets listed in the HIPAA Implementation guides. In some cases, due to the size of the code set if all data values were included, such as the ISO Country Code set 5, we do not include the full set. Users can generate the JSON files directly to include only those values they expect to be in the transactions being processed. In some cases, the data for the code set must be separately licensed.

Note on terminology for NoSQL databases, and how code set support is implemented for the Pack for Healthcare: All the code source sets will be loaded into one Database (default used is hipaa\_code\_sets). A collection is defined for each code source. The collection contains multiple documents, with each document containing an individual code source value, with a description and optional start/end dates.

The document format for each code set entry must conform to the JSON structure in code\_set\_template.json, provided in the External Code List Utilities example.

- [Enabling Type 5 Validation and Rules](#)

To enable Type 5 validation, the following, found in the "ValidationLevels" section, must be set in the hipaa\_data\_compliance\_parameter.json file.

## Enabling Type 5 Validation and Rules

To enable Type 5 validation, the following, found in the "ValidationLevels" section, must be set in the hipaa\_data\_compliance\_parameter.json file.

```
"Type5": {"Enabled": true}
```

By default, Type 5 rules are enabled in the hipaa\_x12\_qualifier.json file. To modify these rules, add them to the hipaa\_rules\_to\_change.json file. Use the field **OnOffIndicator: 1** to enable a rule, and **OnOffIndicator: 0** to disable it. For more detailed instructions, refer to the Configurable rules section of this documentation.

## Code list identifiers table

This table lists the unique code list identifiers used by the Pack for Healthcare along with their associated external code source, or sources. The table also provides information about the contents of the code list.

The code lists that default to PM (pattern match) do not have corresponding \*.json load files. These codes are not shipped because of the following reasons:

- Due to licensing/legal requirements, code lists are not redistributed.
- Lists are available, but prohibitively large, and it is expected that customers would maintain either the full list or a subset of the list that is applicable for their implementation.

If a customer is providing their own data for a table that is not shipped, the corresponding rules that are only set as PM can be updated to CV by including the rules in hipaa\_rules\_to\_change.json and updating the ValidationSubType field.

The rules that use the ICD-10 code sets are set to PM by default, and can be switched to CV, but only the full lists of procedure (T5\_896\_P) and diagnosis (T5\_897\_A) codes are included in the pack. There are no specific guidelines that indicate which subset is allowed for T5\_896\_R Principal Procedure Codes or T5\_897\_C Admitting Diagnosis Codes, and business/medical conditions beyond the scope of data validation could dictate which are allowed. Customers can optionally provide their own subset of codes to use for T5\_896\_R and T5\_897\_C, or update the rules using them to reference T5\_896\_P and T5\_897\_A using the hipaa\_rules\_change.json file.

Note: The Pack for Healthcare provides the maps to transform the source code lists to the xml/json format needed for the Type 5 load and these can be referenced as a model for implementations by the customers to create their own code lists for Type 5 checking.

External Code Source	Code List ID	Code List Contents	On/Off Default	Default Method	Code List Import Name
4	T5_004	Federal Reserve Routing Codes or ABA Transit Routing Number	On	PM	
5	T5_005_A	ISO 3166 Part 1 Country Codes	On	PM	
	T5_005_B	ISO 3166 Part 2 Country Subdivision Codes	On	PM	
	T5_005_C	ISO 4217 Currency or Fund Codes	On	PM	
16	T5_016_A	Dun & Bradstreet D-U-N-S Numbers	On	PM	
	T5_016_B	Dun & Bradstreet D-U-N-S+4 Numbers	On	PM	
22	T5_022	US and MX States and CA Provinces	On	CV	T5_022_States_and_Provinces
	T5_022_A	US States and Outlying Areas	On	CV	T5_022_A_US_States

<b>External Code Source</b>	<b>Code List ID</b>	<b>Code List Contents</b>	<b>On/Off Default</b>	<b>Default Method</b>	<b>Code List Import Name</b>
	T5_022_AB	US States and CA Provinces	On	CV	T5_022_AB_US_States_and_CA_Provs
	T5_022_B	Canadian Provinces	On	CV	T5_022_B_CA_Provinces
	T5_022_C	Mexican States	On	CV	T5_022_C_MX_States
41	T5_041	Universal Product Codes	Off	PM	
	T5_041_A	EAN/UCC-8	On	PM	
	T5_041_B	UCC-12	On	PM	
	T5_041_C	EAN/UCC-13	On	PM	
	T5_041_D	GTIN 14-digit Data Structure	On	PM	
51	T5_051	US ZIP, ZIP Plus 4 and CA Postal Codes	On	PM	
	T5_051_A	US ZIP Codes (5 digits)	On	PM	
	T5_051_B	US ZIP Plus 4 Codes (9 digits)	On	PM	
	T5_051_BC	US ZIP+4 (9 digits only) and CA Postal Codes	On	PM	
	T5_051_C	CA Postal Codes	On	PM	
60	T5_060	Depository Financial Institution (DFI) Identification Numbers	Off	PM	
91	T5_091	CPA Canadian Financial Institution Branch and Institution Numbers	On	PM	
94	T5_094	Equivalents to ISO 8601 Time Zone Codes	Off	PM	
102	T5_102	ISO 639 Language Codes	On	PM	
121	T5_121	HIBCC Health Industry Numbers (HIN)	On	PM	
130/133	T5_130_and_133_P	HCPCS Level I & II Procedure/Service Codes	On	PM	
	T5_130_and_133_Q	HCPCS Level I & II Procedure/Service Modifier Codes	On	PM	
	T5_130_and_133_R	HCPCS Level I & II Procedure/Service Codes (Principal)	On	PM	
130	T5_130_P	HCPCS Level II Ancillary Service/Procedure Codes	On	CV	T5_130_P_HCPCS_II_Proc_Codes
	T5_130_Q	HCPCS Level II Ancillary Service/Procedure Modifier Codes	On	CV	T5_130_Q_HCPCS_II_Modifier_Codes
	T5_130_R	HCPCS Level II Ancillary Service/Procedure Codes (Principal)	On	PM	
131	T5_131_A	ICD-9 Diagnosis Codes	On	PM	T5_131_A_ICD9_Diagnosis_Codes
	T5_131_B	ICD-9 Diagnosis Codes (Principal)	On	PM	
	T5_131_C	ICD-9 Diagnosis Codes (Admitting)	On	PM	
	T5_131_D	ICD-9 Diagnosis Codes (Patient Reason for Visit)	On	PM	
	T5_131_E	ICD-9 Diagnosis Codes (E-Codes)	On	PM	T5_131_E_ICD9_Ext_Causes_Codes
	T5_131_P	ICD-9 In-Patient Hospital Procedure Codes	On	PM	T5_131_P_ICD9_Procedure_Codes
	T5_131_R	ICD-9 In-Patient Hospital Procedure Codes (Principal)	On	PM	
132	T5_132_P	NUBC Procedure Codes	Off	PM	
	T5_132_Q	NUBC Procedure Modifier Codes	Off	PM	
	T5_132_R	NUBC Revenue Codes	On	PM	
	T5_132_S	NUBC Condition Codes	On	PM	
	T5_132_T	NUBC Occurrence Codes	On	PM	
	T5_132_U	NUBC Occurrence Span Codes	On	PM	
132/641	T5_132_V	NUBC Value Codes	On	PM	
	T5_132_S_and_641	NUBC Condition Codes and D & B Condition Code List	Off	PM	
133	T5_133_P	CPT-4 Procedure Codes	On	PM	
	T5_133_Q	CPT-4 Procedure Modifier Codes	Off	PM	
135	T5_135_A	ADA Systemized Nomenclature of Dentistry Diagnosis Codes	Off	PM	
	T5_135_B	ADA Systemized Nomenclature of Dentistry Diagnosis Codes (Principal)	Off	PM	
	T5_135_P	ADA CDT-4 Procedure Codes	On	PM	
	T5_135_Q	ADA CDT-4 Procedure Modifier Codes	Off	PM	
	T5_135_T	ADA Universal National Tooth Designation System	On	PM	
	T5_135_U	ADA Code on Dental Procedures and Nomenclature Oral Cavity Codes	On	PM	
139	T5_139	Claim Adjustment Reason Codes (CARC)	On	CV	T5_139_Claim_Adjust_Reason_Codes
229	T5_229	Diagnosis Related Group (DRG) Codes	On	CV	T5_229_Dx_Related_Group_Codes
230	T5_230	NUBC Point of Origin for Admission or Visit (Admission Source Code)	On	PM	
231	T5_231	NUBC Priority of Admission or Visit (Admission Type Code)	On	PM	
235	T5_235	NUBC Claim Frequency Type Codes	On	PM	

<b>External Code Source</b>	<b>Code List ID</b>	<b>Code List Contents</b>	<b>On/Off Default</b>	<b>Default Method</b>	<b>Code List Import Name</b>
236	T5_236	NUBC Uniform Billing Claim Form Bill Type Codes	On	PM	
237	T5_237	CMS Place of Service Codes for Professional Claims	On	CV	T5_237_Place_of_Service_Codes
239	T5_239	NUBC Patient Status Codes	On	PM	
240	T5_240_A	NDC National Drug Codes in 5-4-2 Format	On	PM	
	T5_240_B	NDC National Health Related Item Codes in 4-6 Format	Off	PM	
244	T5_244	Codes identifying the nature of insurance coverage	On	CV	T5_244_Line_of_Business_Codes
245	T5_245	NAIC Insurance Company Identifiers	On	PM	
284	T5_284	NCCI Nature of Injury Codes	Off	PM	
307	T5_307	National Council for Prescription Drug Programs Pharmacy Numbers	On	PM	
327	T5_327	SWIFT BICs (8 or 11 characters)	On	PM	
359	T5_359	CMS Home Health Treatment Codes	Off	PM	
407	T5_407	Nature of Injury Code from Occupational Injury and Illness Manual	Off	PM	
411	T5_411	CMS Claim Payment or Remittance Advice Remark Codes (RARC)	On	CV	T5_411_Remittance_Advice_Codes
457	T5_457	NISO Z39.53 Language Codes	On	PM	
468	T5_468	CMS Ambulatory Patient Classification Codes	Off	PM	
507	T5_507	Health Care Claim Status Category Codes	On	CV	T5_507_Claim_Status_Cat_Codes
	T5_507_A	Health Care Claim Status Category D0 or E Codes	On	CV	T5_507_A_Claim_Stat_Cat_D0_or_E
	T5_507_B	Health Care Claim Status Category R Codes	On	CV	T5_507_B_Claim_Stat_Cat_R_Codes
	T5_507_C	Health Care Claim Status Category Not R Codes	On	CV	T5_507_C_Claim_Stat_Cat_Not_R
508	T5_508	Claim Status Codes	On	CV	T5_508_Claim_Status_Codes
513	T5_513_P	Home Infusion EDI Coalition (HIEC) Product/Service Code	Off	PM	
	T5_513_Q	Home Infusion EDI Coalition (HIEC) Product/Service Code (Modifier)	Off	PM	
530	T5_530	NCPDP Claim Reject/Payment Codes	Off	PM	
537	T5_537	CMS National Provider Identifiers (NPI)	On	PM	
540	T5_540	CMS Health Care Plan Identifiers (HPID)	Off	PM	
576	T5_576_P	Workers Compensation (IAIABC) Procedure/Supply Codes	Off	PM	
	T5_576_Q	Workers Compensation (IAIABC) Procedure/Supply Modifier Codes	Off	PM	
582	T5_582	DMERC (Certificate of Medical Necessity) CMN Form Identifiers	Off	PM	
656	T5_656	ACORD Form Type Codes	Off	PM	
663	T5_663_P	Logical Observation Identifier Names and Codes (LOINC)	On	PM	
	T5_663_Q	Logical Observation Identifier Names and Codes (LOINC) Modifiers	On	PM	
682	T5_682	Health Care Provider Taxonomy Codes	On	CV	T5_682_Provider_Taxonomy_Codes
716	T5_716_P	HIPPS Skilled Nursing Facility Rate Codes	On	CV	T5_716_P_HIPPS_Fac_Rate_Codes
	T5_716_Q	HIPPS Skilled Nursing Facility Rate Modifier Codes	Off	PM	
843	T5_843_P	Advanced Billing Concepts (ABC) Procedure Codes	Off	PM	
	T5_843_Q	Advanced Billing Concepts (ABC) Procedure Modifier Codes	Off	PM	
	T5_843_R	Advanced Billing Concepts (ABC) Procedure Codes (Principal)	Off	PM	
844	T5_844	DOD Dependent Eligibility Category	Off	PM	
859	T5_859	CDC Classification of Race or Ethnicity Codes	Off	PM	
860	T5_860	CDC Race or Ethnicity Collection Codes	Off	PM	
886	T5_886	Health Care Service Review Decision Reason Codes	On	CV	T5_886_Decision_Reason_Codes
896	T5_896_P	ICD-10-PCS In-Patient Hospital Procedure Codes	On	PM	
	T5_896_R	ICD-10-PCS In-patient Hospital Procedure Codes (Principal)	On	PM	
897	T5_897_A	ICD-10-CM Diagnosis Codes	On	PM	
	T5_897_B	ICD-10-CM Diagnosis Codes (Principal)	On	PM	
	T5_897_C	ICD-10-CM Diagnosis Codes (Admitting)	On	PM	
	T5_897_D	ICD-10-CM Diagnosis Codes (Patient Reason for Visit)	On	PM	
	T5_897_E	ICD-10-CM Diagnosis Codes (E-Code)	On	PM	
932	T5_932	UPU Universal Postal Codes	Off	PM	
932/51	T5_932_and_051	US ZIP, ZIP Plus 4, CA and UPU Postal Codes	Off	PM	

External Code Source	Code List ID	Code List Contents	On/Off Default	Default Method	Code List Import Name
	T5_932_and_051_BC	US ZIP+4 (9 digits only), CA and UPU Postal Codes	Off	PM	
DOD1	T5_DOD1	DOD Health Service Region	On	PM	

## Type 6 Validation

This support is only available with the HIPAA Data Compliance application.

### Overview

Type 6 validation can be defined as the process of confirming that HIPAA X12 transaction data for claims complies with a specific line of service requirements. Details about line of service requirements are included in the HIPAA X12 Implementation Guides.

These line of service requirements are generally expressed as situational rules and may be associated with segments or data elements. Line of service categories such as ambulance, chiropractic, and podiatry are given as examples of line of service categories in the Type 6 definition. In the Pack for Healthcare, the following principles have been applied to distinguish line of service requirements from recommendations or other types of situational rules:

- Self-contained requirements - A self-contained requirement is one where all situational criteria can be evaluated using only the information present in the transaction set. A situational rule that includes a condition such as "when the distance of transportation is known" or "when the data is present in the UMO's system" is a recommendation, not a requirement.
- Unambiguous requirements - A requirement is considered unambiguous if all of its criteria can be evaluated using information from identifiable locations.
- Distinct requirements - A distinct line of service requirement is one that is not already enforced by another HIPAA X12 validation, such as a syntax rule enforcing the presence of related elements.

The scope of Type 6 validation is limited to the Health Care Claim (837) and Health Care Services Review (278) transaction sets.

- **Implementation**  
To understand how these Type 6 rules are processed in hipaa\_data\_compliance, consider the following three aspects of Type 6 rules:
- **Configuration**  
Database requirements

## Implementation

To understand how these Type 6 rules are processed in hipaa\_data\_compliance, consider the following three aspects of Type 6 rules:

1. What the line of service is – e.g. ambulance.
2. What segment/element in the transaction determines whether the claim involves this line of service.

The first is the rule that is present in the hipaa\_x12\_qualifier.json rules file.

The second aspect is encoded as a category within the code set name in the hipaa\_x12\_qualifier.json file, such as AMB for an ambulance in the code table ID field "CodeTableID" : "T6\_ST\_MRT\_AMB".

The final aspect is also encoded as a service location, in the code table id, such as "ST" in "CodeTableID" : "T6\_ST\_MRT\_AMB".

Table 1. Type 6 Service Categories

Category code	Description
ANE	Anesthesia
CHR	Chiropractic (spinal manipulation)
DEN	Dental
DME	Durable Medical Equipment
HHC	Home Health Care Services
HPC	Hospice
MBH	Mental and Behavioral Health
MRT	Medically Related Transportation
OXY	Oxygen Therapy
PEN	Parenteral / Enteral Nutrition
POD	Podiatry
RXD	Prescription Drug
SUG	Surgery
VIS	Vision (optometry)

Table 2. Type 6 Service Locations

Transaction Set	Source Type Code ST	Product or Service Identification PS	Form Identification code FC
278	2000E UM03 2000F UM03	2000F SV101-01 + SV101-02 2000F SV202-01 + SV202-02 2000F SV301-01 + SV301-02	

Transaction Set	Source Type Code ST	Product or Service Identification PS	Form Identification code FC
837		2400 SV101-01 + SV101-02 2400 SV202-01 + SV202-02 2400 SV301-01 + SV301-02	2440 LQ01+02

## Configuration

### Database requirements

The code values that indicate a particular line of service are checked with a database query, just as the Type 5 code values are checked.

The Pack for Healthcare does not come with a packaged database application. To use Type 6 validation, you must have a Database server accessible that has MongoDB interface support for queries and load.

The maps provided in the External Code List Utilities example, the primary map being load\_all\_code\_sets, can be run to load the code set collections with the documents that represent the list of allowed code values for the line of service for the Type 6 rules that are implemented in the Pack for Healthcare. If these values need to be updated between Pack releases, they need to be updated in the database. Updates for individual code values, to add/remove values, or to add a start date or end date can be done by reloading a full updated code list in the JSON format that the utilities map uses or by using any of the UI tooling supported by your database server.

If you need to modify the values in the code sets, the key difference in Type 6 tables is that if the value that indicates a particular line of service is represented by two different elements or subelements, the values are concatenated with an underscore in the table.

```
{
 "code" : "HC_A0170"
 "description" : "Transport parking fees/tolls",
 "startDate" : "1982-01-01"
}
```

- [Enabling Type 6 Validation and Rules](#)

To enable Type 6 validation, the following, found in the “ValidationLevels” section, must be set in the hipaa\_data\_compliance\_parameter.json file.

## Enabling Type 6 Validation and Rules

To enable Type 6 validation, the following, found in the “ValidationLevels” section, must be set in the hipaa\_data\_compliance\_parameter.json file.

```
"Type6": {"Enabled": true}
```

By default, Type 5 rules are enabled in the hipaa\_x12\_qualifier.json file. To modify these rules, add them to the hipaa\_rules\_to\_change.json file. Use the field **OnOffIndicator: 1** to enable a rule, and **OnOffIndicator: 0** to disable it. For more detailed instructions, refer to the Configurable rules section of this documentation.

## Type 7 Validation

This support is only available with the HIPAA Data Compliance application.

### Overview

In the original WEDI SNIP compliance testing model, Type 7 validation was limited to those explicit requirements specified in the HIPAA Implementation Guides for a small set of payer entities.

Within the healthcare industry, however, the term Type 7 validation is typically used in a broader sense that encompasses generalized, business-to-business testing based on requirements from additional sources. The Pack for Healthcare supports this expanded interpretation.

In the IBM Transformation Extender Pack for Healthcare, validation requirements are classified as Type 7 if either or both of the following conditions are true:

- The requirement is more restrictive than the standard requirements as documented in the HIPAA X12 Implementation Guides.
- The requirement does not apply globally to all defined business partners.

The scope of the validation performed by Types 1 through 6 is limited to enforcing the ‘standard’ requirements documented in the HIPAA X12 Implementation Guides (TR3s) and applying those requirements globally to all business partners. Type 7 validation extends that scope with an additional layer of checking that allows selective enforcement of business-specific requirements.

- [Implementation](#)
- [Configuration](#)

Database requirements

## Implementation

Type 7 Rules specified in the Qualifier file

Type 7 validation with a custom user exit

The HIPAA data compliance application is not a full trading partner application. Still, the new features allow for basic customization based on the value in the data's functional group Application Receiver ID (GS03).

For each "Institution" in the hipaa\_x12\_type\_7\_value.json input file to hipaa\_data\_compliance, if the Type 7 rule in the hipaa\_x12\_qualifier.json contains this same keyword, AND the GS03 value of the transaction being validated matches the "GS03Value" field in the hipaa\_x12\_type\_7\_value.json input file, then hipaa\_data\_compliance will validate the rest of the rule as defined in the hipaa\_x12\_qualifier.json.

Type 7 validation with a custom user exit

In addition to the Type 7 rules defined in the hipaa\_x12\_qualifier.json file, you can also provide a custom user exit in Java or C++ .

The custom user exit must be designed to use HIPAA data and delimiters in a specific format. The custom user exit must conform to the format of the data that is generated during run time by the hipaa data compliance application, which is only available during run time.

The string of delimiters must follow this format:

- Element Separator (ISA Element separator)
- Repetition Separator (ISA11)
- Component Element Separator (ISA16)
- Segment Terminator (ISA Terminator)

The string of HIPAA EDI starts with the segment identifier followed by the loop ID. For example:

ST\*\*270\*1235\*005010X279A1~

BHT\*\*0022\*13\*10001235\*20060501\*1320~

HL\*2000A\*1\*\*20\*1~

NM1\*2100A\*PR\*2\*ABC COMPANY\*\*\*\*\*PI\*842610001~

HL\*2000B\*2\*1\*21\*1~

NM1\*2100B\*1P\*1\*PENWISE\*MARCUS\*\*\*XX\*0202034209~

REF\*2100B\*JD\*178583840~

REF\*2100B\*EO\*477563928~

SE\*\*22\*1235~

## Configuration

Database requirements

To enable partner-specific code set or line of service checking, you must have the code set values loaded into the database, just as the code values for Type 5 and Type 6 are configured.

The Pack for Healthcare does not come with a packaged database application. To use Type 7 validation for partner-specific code value checking, you must have a Database server accessible that has MongoDB interface support for queries and load.

Refer to the Type 5 validation configuration section for additional details on database configuration and load requirements.

- [Enabling Type 7 Validation and Rules](#)

To enable Type 7 validation, found in the "ValidationLevels" section, must be set in the hipaa\_data\_compliance\_parameter.json file.

## Enabling Type 7 Validation and Rules

To enable Type 7 validation, the following, found in the "ValidationLevels" section, must be set in the hipaa\_data\_compliance\_parameter.json file.

"Type7": {"Enabled": true}

By default, Type 5 rules are enabled in the hipaa\_x12\_qualifier.json file. To modify these rules, add them to the hipaa\_rules\_to\_change.json file. Use the field **OnOffIndicator**: 1 to enable a rule, and **OnOffIndicator**: 0 to disable it. For more detailed instructions, refer to the Configurable rules section of this documentation.

## HIPAA EDI component examples

This documentation describes the HIPAA EDI component examples.

- [ICD-10](#)

The International Statistical Classification of Diseases and Related Health Problems 10th Revision (ICD-10) is a set of codes classified by the World Health Organization (WHO) used by physicians, hospitals, and other healthcare organizations to identify patient diagnosis and medical procedures. In the United States, the codes are used by providers and payers for many uses within health care systems, including patient care decision support and claims adjudication.

- [\*\*278N example\*\*](#)  
The 278N example contains sample mapping with the 278 transaction, "Health Care Services Review Information – Notification". By using sample input data with multiple 278 transactions, the example creates an XML based report for the notifications included in the input.
  - [\*\*HIX example\*\*](#)  
The Health Insurance Exchange (HIX) example demonstrates structures and concepts that surround the HIX processing structures.
  - [\*\*Preview Draft 80X0\*\*](#)  
The ASC X12 Insurance Subcommittee, called X12N, is currently managing a series of public review periods to define new implementation guides for the healthcare insurance industry. These implementation guides will eventually replace the HIPAA-adopted TR3s.
  - [\*\*X12N example\*\*](#)  
This example contains schemas that support X12 standard integrity checking (WEDI SNIP Type 1 Validation) for a set of healthcare industry implementation guides produced by ASC X12's Insurance Subcommittee (X12N). These schemas do not conform to the industry TR3s but to the base X12 standard. They are useful for data transformation purposes; they should not be used for data validation for any WEDI/SNIP level above 1.
  - [\*\*External Code List Utilities example\*\*](#)  
The External Code List Utilities example contains a utility map that loads the code set database with the external code sets that are supported in the pack into a MongoDB NoSQL database.
  - [\*\*Type 7 User Exit example\*\*](#)  
This example contains a Java Program that can be used as a guide to help with the development and requirements of a Java custom user exit for Type 7 validation.
  - [\*\*ClaimToExposure tracking example\*\*](#)  
In the era of COVID-19 many organizations will find it helpful to be able to do a quick scan of the data they already have access to in order to quickly isolate individuals with diagnosis codes of COVID-19 or indicating COVID-19 in order to take quick action.
  - [\*\*X12N personal health records example\*\*](#)  
To support ways for the payer community to leverage existing B2B infrastructure while also beginning to work with HL7 CDA R2 and FHIR (in Json or XML), X12 has a draft implementation guide that describes the use of the AXC X12 Patient Information (275) transaction set for the business purpose of transferring Personal Health Record (PHR) information between different health plans.
- 

## ICD-10

The International Statistical Classification of Diseases and Related Health Problems 10th Revision (ICD-10) is a set of codes classified by the World Health Organization (WHO) used by physicians, hospitals, and other healthcare organizations to identify patient diagnosis and medical procedures. In the United States, the codes are used by providers and payers for many uses within health care systems, including patient care decision support and claims adjudication.

The ICD-10-CM and ICD-10-PCS code sets replace the current ICD-9 diagnosis and procedure codes. The ICD-10 code sets that replace the ICD-9 codesets are required for all HIPAA transactions, including outpatient claims (with dates of service), and inpatient claims (with dates of discharge), on and after October 1, 2013. The change to ICD-10 impacts coding for all HIPAA participants.

- [\*\*ICD-10 example\*\*](#)
- 

## ICD-10 example

In the health care industry, there is a short-term need for 'cross-walking' between the old ICD-9 and new ICD-10 code sets to help the industry migrate systems, applications, and data. Since there is not always a one-to-one code equivalent between the coding systems, software such as Transformation Extender ILOG JRules can be used to apply intelligent rules and logic for appropriate code selections.

- [\*\*Insulating legacy systems\*\*](#)  
In some cases, you, as a HIPAA participant, might want to temporarily insulate your legacy systems from the impact of the ICD-10 changes. In such situations, you must accept HIPAA 5010 transactions with ICD-10 codes and then exchange them for ICD-9 codes that your system understands.
  - [\*\*Locating the ICD-10 example files\*\*](#)  
This topic identifies the directory location of the ICD-10 example files.
  - [\*\*Using the ICD-10 example files\*\*](#)  
This topic describes how to use the ICD-10 example files.
- 

## Insulating legacy systems

In some cases, you, as a HIPAA participant, might want to temporarily insulate your legacy systems from the impact of the ICD-10 changes. In such situations, you must accept HIPAA 5010 transactions with ICD-10 codes and then exchange them for ICD-9 codes that your system understands.

Such a process typically includes parsing the HIPAA EDI data to isolate the ICD-10 codes, communicating the codes to be cross-walked to the rules/cross-walking engine, converting the codes based on pre-defined criteria, using the cross-walked codes in legacy systems, and tracking the changes made to the codes for audit and historical reporting. This example demonstrates a small portion of such a system. The Transformation Extender processes incoming HIPAA EDI data and communicates with a rules engine such as "IBM ILOG JRules," using an XML file, and sends the rules to the engine a structure containing the codes that must be cross-walked.

In this scenario, the rules engine (not included) performs the actual cross walk and returns a structure that contains both the original ICD-10 codes and the replacement ICD-9 codes. The resulting XML structure is then used by the Transformation Extender Platform to replace the codes in the original HIPAA EDI data.

It should be noted that this example is a black box inside of a black box. It does not show the orchestration of the various components. An orchestration tool, such as business Process Management, would call on the various transformation and crosswalking components to coordinate their efforts. Also, the rules engine is treated as a black box. In the example, one set of cross-walked test data is shown. There is no functioning rules engine in the example. This example is not intended to be used as a confirmation of the crosswalking done, as the test case is fictitious.

The process behind this specific example is to map ICD-10 codes from a HIPAA 837 professional claim transaction (837P) into an XML format described by a schema. The XML instance document generated by the Transformation Extender Platform is expected as input into a rules engine. A simulated modified XML instance document that contains both the ICD-10 codes and the equivalent ICD-9 codes is returned to the Transformation Extender, where that information is used to replace the ICD-10 codes in the original claim transaction. This cross-walking usually occurs during the process of a real transformation, for example, a transformation between the HIPAA EDI and a

proprietary format. For the sake of simplicity, the transformation in this example does not reformat the data; it simply includes the new codes. A true cross-walking service would do much more than take one code and map it to another, it could have the intelligence to handle one code in, many codes out, multiple codes in, one code out, or exception cases where human process flows should be initialized.

---

## Locating the ICD-10 example files

This topic identifies the directory location of the ICD-10 example files.

The ICD-10 example is located in the following folder:

*install\_dir\packs\healthcare\_vn.n.n\hipaa\examples\ICD-10\_iLog*

Note: Throughout this documentation, *install\_dir* indicates the location of the Transformation Extender Platform, and *n.n.n* indicates the version number of the Pack for Healthcare.

The Transformation Extender type tree, *icd\_transaction\_report.mtt*, is located in the following folder:

*install\_dir\...\ICD-10\_iLog\trees*

This is a representation of the XML schema *icd\_transaction\_report.xsd* located in the following folder:

*install\_dir\...\ICD-10\_iLog\mapsandschemas*

The type tree is used to create XML output and parse XML input, for example.

---

## Using the ICD-10 example files

This topic describes how to use the ICD-10 example files.

### About this task

Perform the following steps to use the example:

### Procedure

1. Open the map source file, *claim\_to\_ilog.mms*, in the ...\\mapsandschemas folder and then build all maps.
  2. Execute the run map *to\_ilog*. The input file, *837p\_5010a1\_icd10\_example.dat*, can be found in the ...\\mapsandschemas\\data folder. The map execution produces the XML output file *toilLog.xml*, which contains the ICD-10 codes. The output file from this step can be found in ...\\mapsandschemas\\data folder.
  3. Execute the run map *from\_ilog*. The input files *837p\_5010a1\_icd10\_example.dat* (input to step 2) and *fromilLog.xml* (output from step 2, enhanced by rules ending processing) can be found in the folder ...\\mapsandschemas\\data. The file *fromilLog.xml* is an example file created by a rules engine. The map execution produces the HIPAA 837 professional file (*837p\_5010a1\_out.dat*) containing the ICD-9 codes. The output file can be found in the ...\\mapsandschemas\\data folder.
  4. Execution of the run map *val\_edi* can be used to ensure the file *837p\_5010a1\_out.dat* is valid.
- 

## 278N example

The 278N example contains sample mapping with the 278 transaction, "Health Care Services Review Information – Notification". By using sample input data with multiple 278 transactions, the example creates an XML based report for the notifications included in the input.

The coded information for Purpose Code (from BHT02), event category and type, from the UM segment along with patient event trace numbers, are mapped in the output XML file.

The example files install within the imported project.

- [Location of the 278N example files](#)  
This topic identifies the directory location of the 278N example files.
  - [Using the 278N example](#)  
This topic describes how to use the 278N example files.
- 

## Location of the 278N example files

This topic identifies the directory location of the 278N example files.

The 278N example is located in the following folder:

*install\_dir\packs\healthcare\_vn.n.n\hipaa\examples\278N*

Note: Throughout this documentation, *install\_dir* refers to the installation location of the base product, and *n.n.n* indicates the version number of the Pack for Healthcare.

---

## Using the 278N example

This topic describes how to use the 278N example files.

Perform these steps to use the 278N example:

1. Open the map **service\_notification** in the ..\mapsandschemas folder, then build the **service\_notification** map.
2. Execute the **service\_notification** map. The input file, **278n\_5010\_examples.dat**, can be found in the ..\data directory. The output file, **service\_notifications.xml**, also found in the ..\data directory, is produced using **service\_notification.mtt**, based on the **service\_notification.xsd** schema in the ..\mapsandschemas directory.

For convenience, the 278N example contains a schema that only contains the 278 transaction.

For any production use of validation or mapping with the 278 transaction, the schemas in the following directory should be used. The schemas in this directory are supported with the latest fixes and enhancements:

```
install_dir\packs\healthcare_vn.n.n\hipaa\trees
```

## HIX example

The Health Insurance Exchange (HIX) example demonstrates structures and concepts that surround the HIX processing structures.

The HIX 820 is the 005010X306 transaction set, for which full support is included in the Pack for Healthcare. The HIX 834 or HIPAA FFE 834 is the 005010X220A1 plus FFE Companion Guide. The existing HIPAA 834 is supported in the pack, but this example incorporates the FFE (Federally Funded Exchange) Companion guide, which is a set of rules that is above HIPAA and is related to specific use for a specific task. In typical WEDI/SNIP terminology, this is WEDI/SNIP type 7. The 834 support beyond HIPAA is shown in this example.

The example files install within the imported project.

- [Locating the HIX example files](#)  
This topic identifies the directory location of the HIX example files.
- [HIX example executable maps](#)  
There are two executable maps in the HIX example, init\_enroll\_ffe\_to\_qhp and init\_enroll\_conf\_qhp\_to\_ffe.
- [Using the HIX example files](#)  
This topic describes how to use the HIX example files.

## Locating the HIX example files

This topic identifies the directory location of the HIX example files.

The HIX example is located in the following folder:

```
install_dir\packs\healthcare_vn.n.n\hipaa\examples\hix
```

Note: Throughout this documentation, *install\_dir* indicates the location of the base product, and *n.n.n* indicates the version number of the Pack for Healthcare.

## HIX example executable maps

There are two executable maps in the HIX example, init\_enroll\_ffe\_to\_qhp and init\_enroll\_conf\_qhp\_to\_ffe.

- [HIPAA to HIX example map](#)  
This example shows how a HIX 834 is created. The map, init\_enroll\_ffe\_to\_qhp, takes an initial enrollment structure and transforms it into an FFE compliance 834 for transmission to a qualified health plan.
- [HIX to HIX example map](#)  
The init\_enroll\_conf\_qhp\_to\_ffe example map takes the HIX enrollment created by init\_enroll\_ffe\_to\_qhp and generates a HIX Enrollment Confirmation, which is returned for the exchange.

## HIPAA to HIX example map

This example shows how a HIX 834 is created. The map, init\_enroll\_ffe\_to\_qhp, takes an initial enrollment structure and transforms it into an FFE compliance 834 for transmission to a qualified health plan.

There are two inputs to this map.

- Input 1 - HIPAA\_834\_Transmission - HIPAA 834 enrollment message. This is the same HIPAA message that is used to enroll members into private payer plans. Although the HIPAA 834 and the HIX 834 are similar (and based on the same transaction) there are some differences. This example map shows one possible way to fill in the gaps.
- Input 2 - MemberReportingCategories - This input is a proprietary file that is used for lookup and data embellishment. This case uses a set of Member Reporting Categories with 'Maintenance Reason Codes'. The reason codes equate to the reason that this message is created, such as for a new enrollment, or to modify an existing enrollment.

There are two outputs from this map:

- Output 1 - HIX\_834\_Transmission - Temporary card output designed for interim results.
- Output 2 - HIX\_834\_Transmission - Manipulates the output 1 and is the final transformed product.

## HIX to HIX example map

The init\_enroll\_conf\_qhp\_to\_ffe example map takes the HIX enrollment created by init\_enroll\_ffe\_to\_qhp and generates a HIX Enrollment Confirmation, which is returned for the exchange.

## Using the HIX example files

This topic describes how to use the HIX example files.

### About this task

These steps describe how to run the HIX example.

The files described in the following steps are in the maps and dat sub-directories in the following location:

`install_dir\packs\healthcare_vn.n.n\hipaa\examples\hix\`

### Procedure

1. Open the map source file ..\maps\hipaa\_to\_hix.mms.
2. Build all maps.
3. Run the init\_enroll\_ffe\_to\_qhp map. The input files, hipaa\_834\_5010a1\_examples.dat and MemberReporting.txt, are in the ..\data folder. When you run the map, the file init\_enroll\_ffe\_to\_qhp.out is produced. This file contains the initial enrollment. The output file from this step is in the ..\data folder.
4. Open the map source file ..\maps\hix\_to\_hix.mms.
5. Build all maps.
6. Run the init\_enroll\_conf\_qhp\_to\_ffe map. The input file is init\_enroll\_ffe\_to\_qhp.out (output from step 3). The output Initial Enrollment Confirmation is the init\_enroll\_conf\_qhp\_to\_ffe.out file. This file is located in the ..\data folder.

## Preview Draft 80X0

The ASC X12 Insurance Subcommittee, called X12N, is currently managing a series of public review periods to define new implementation guides for the healthcare insurance industry. These implementation guides will eventually replace the HIPAA-adopted TR3s.

Because of the current status of the final versions of the implementation guides, the Pack for Healthcare includes examples that provide a preview based on the currently published but not adopted versions. The example includes a schema to allow for validation in accordance with the Type 2 HIPAA Implementation Guide Requirement Testing for the Published 8020 TR3 for these transactions:

- HP – Health Care Claim Payment/Advice – 835, Version 008020X322
- BE – Benefit Enrollment and Maintenance - 834, Version 008020X333
- HS – Eligibility, Coverage, or Benefit Inquiry – 270, Version 008020X332
- HB – Eligibility, Coverage, or Benefit Information - 271, Version 008020X332

For these transactions, step-up and step-down example maps are provided to demonstrate one way to transform data conforming to the HIPAA adopted version 5010 TR3's to the draft TR3 requirements based on X12 8020.

The step-up maps also demonstrate one possible way to enrich the data with new information that is specified in the 8020 TR3's through JSON input.

See the X12n example for X12 8020 and 8030-based schemas with transactions covered by currently published X12N implementation guides to allow for validation in accordance with WEDI/SNIP Type 1 EDI Standard Integrity Testing.

The example files install within the imported project.

- [Location of Preview Draft 80X0 example files](#)

- [Schemas and JSON metadata](#)

This topic describes the example 80x0 schemas and metadata files, as well as notes on how to use these schemas for validation.

- [Step-up and step-down maps](#)

The step-up and step-down maps provide the basis for mapping simple correspondence in unchanged elements, the changes in positioning of information within the transaction, new or changed segments loops, and other differences, both in the implementation guide changes between the two versions, and the base X12 changes.

- [Preview Draft 8010 example source maps](#)

## Location of Preview Draft 80X0 example files

When running the example in Design Studio, the example files are at `Pack_install_dir\hipaa\examples\preview_draft_80x0`.

When running the example on the Design Server, the example files are installed within the imported `hipaaExamples.zip` project.

The example files install within the imported project.

## Schemas and JSON metadata

This topic describes the example 80x0 schemas and metadata files, as well as notes on how to use these schemas for validation.

The schema hipaa\_x12\_type\_2\_v80x0 provides type 2 validation of these transactions implemented in this example:

:

- Health Care Claim Payment/Advice, or 835
- Benefit Enrollment and Maintenance, or 834
- Eligibility, Coverage, or Benefit inquiry/or information, or 270/271

### Notes for using these schemas for validation:

- Element values – for codes that have a limited set of allowed values that is different from the X12 code list, compliance is enforced through a limited restriction list on the element or with a component rule. The limited restriction list is used if all occurrences of the element in the transaction have the same set of allowed values.
- Valid X12 elements that are not used in certain segments or composites are defined in the schema as components of the segment or composite but have the component rule of **ABSENT(\$)**.
- Segments or composites that are part of the base X12 transactions but are not used per the implementation guide are not included in the schema.
- Where the implementation guides identify a limit for the maximum length of a text data element that is less than the X12 length, a component rule is used.
- Where the implementation guides identify a limit for elements that represent monetary amounts, date, or time, the schema item properties for that element enforce the limitation.
- If a situational rule within the implementation guide is based only on the data contained within a single segment, that rule is enforced with a component rule in the schema.
- Component rules that enforce the relational conditions from the base X12 definition are included in the schema if they are not superseded by any of the Type 2 component rules.

For convenience, a 5010 Type 2 schema is included that contains only the 834 and 835 transactions, as well as the segments and elements required for those transactions. This schema is: `hipaa_x12_type_2_step`

The following JSON files are used as the input metadata for the step-up maps as one way to enrich the data with new information that is specified in the Preview Draft 8020 TR3s.

- `v80x0x322_template.json`
- `v80x0x333_template.json`
- `v80x0x332_template.json`

These files do not contain objects to represent all the loops, segment, and elements in the 80x0 transactions, but only those that are changed or new for 8010 or higher version and are used in mapping rules in the step- up maps. The actual version to be used in mapping, 8020, is specified in the file.

## Step-up and step-down maps

The step-up and step-down maps provide the basis for mapping simple correspondence in unchanged elements, the changes in positioning of information within the transaction, new or changed segments loops, and other differences, both in the implementation guide changes between the two versions, and the base X12 changes.

The step-up and step-down maps use the Type 2 schemas to map the included transactions from the HIPAA 5010 base to the new 80x0 TR3 format.

The step-up maps have an additional input that uses the IBM Integration Platform native JSON support to provide a way to extend the output 80x0 data with new fields supported in the new draft implementation guides.

### Notes for using the step-up and step-down maps:

In addition to loop or segment-specific functional maps and map rules, the following notes apply to the step-up and step-down maps:

- If an element or composite is not used in 80x0, it is mapped with an **=NONE** rule, with a comment indicating that it is not used in the 80x0 TR3.
- Loop names in 80x0 schemas and functional map names in the maps reflect an abbreviation of the name in the 80x0 TR3, so they might be different from the loop name in the 5010 schema.
- Where text elements have component rules in the 80x0 schemas to enforce a maximum length, the **LEFT** function is used in step-down mapping to 5010, if the Type 2 schema enforced the length through type item properties.
- As the schema item properties are set to allow pads, and pad to min content, optional items in the mapping are wrapped in a functional map to prevent spaces to be built on output.
- Data for elements that are present in 5010 data but are no longer used in 80x0 and do not correspond to a different element in 80x0, are discarded in the step-up mapping. Similarly, data for new elements defined for use in the 80x0 TR3 that were not present in the 5010 TR3 are discarded.
- Additional map rules and functional maps are implemented where needed for mapping between the two versions.

## Preview Draft 8010 example source maps

There are six source maps in the Preview Draft 80x0 example.

Each source map contains a single executable map and requires functional maps to perform a step-up from 5010 to 80x0 or step-down from 80x0 to 5010.

The source map, `V8020x322_stepup`, contains the executable map `5010_to_8020x322` with the following inputs from the directory:

`install_dir\packs\healthcare_vn.n.n\hipaa\examples\preview_draft_80X0\data`

- **835\_5010\_stepup\_examples.dat** – contains 15 valid 835 transactions
- **v80x0x322\_examples.json** – contains data for new 8010 fields for transactions with a Transaction Control Number of 1309 and the base “ANY” transaction information for the other transactions in the 5010 input.

The single output card creates the file:

#### **835\_8020\_stepup\_examples.out**

The corresponding step-down map for the 835 is **8020x322\_to\_5010.mmc**, in source map **V80x0x322\_stepdown.mms**.

This map has a single input card, which uses the same file produced by the step-up map, **835\_80x0\_stepup\_examples.out**, producing **835\_5010\_stepdown\_examples.out**.

The source map **V70x0x333\_stepup.mms** contains the executable map **5010\_to\_8020x333** with the following inputs:

- **834\_5010\_stepup\_examples.dat** – contains 20 valid 824 transactions
- **v80x0x333\_examples.json** – contains data for new 8020 fields for transactions with a Transaction Control Number of 1289 and the base “ANY” transaction information for the other transactions in the 5010 input.

The single output card creates the file

#### **834\_8020\_stepup\_examples.out**

The corresponding step-down map for the 834 is **8020x333\_to\_5010.mmc**, in source map **V80x0x333\_stepdown.mms**.

This map has a single input card, which uses the same file produced by the step-up map, **834\_8020\_stepup\_examples.out**, producing **834\_5010\_stepdown\_examples.out**.

---

## X12N example

This example contains schemas that support X12 standard integrity checking (WEDI SNIP Type 1 Validation) for a set of healthcare industry implementation guides produced by ASC X12’s Insurance Subcommittee (X12N). These schemas do not conform to the industry TR3s but to the base X12 standard. They are useful for data transformation purposes; they should not be used for data validation for any WEDI/SNIP level above 1.

The example files install within the imported project.

- [\*\*Location of the X12N example files\*\*](#)  
This topic identifies the directory location of the X12N example files.
- [\*\*X12N schemas\*\*](#)  
This topic provides a brief description of the X12N schemas.
- [\*\*Using the X12N example files\*\*](#)  
There are no maps included in the X12N example. You can select any of the provided schemas to use in validation map, or maps created to transform to, or from, the supported versions and transactions.

---

## Location of the X12N example files

This topic identifies the directory location of the X12N example files.

The Preview Draft 7030 example is located in the following folder:

*install\_dir\packs\healthcare\_vn.n.n\hipaa\examples\x12n*

Note: Throughout this documentation, *install\_dir* indicates the location of the base product, and *n.n.n* indicates the version of the Pack for Healthcare.

---

## X12N schemas

This topic provides a brief description of the X12N schemas.

The schemas support X12 standard integrity checking through the defined items and groups that represent the X12 elements, composites, segments, loops, and transactions contained in the supported implementation guides. In addition, Relational conditions that are defined in the standard are enforced through schema component rules on the segment or composite group components.

This covers any of the relational conditions as defined in the “Relational Conditions” subsection of the X12 Application Control Structure, Definitions and Concepts (section 3.7.3.2.3 in version 7030). These conditions are:

- **P** – Paired or Multiple
- **R** - Required
- **E** – Exclusion
- **C** - Conditional
- **L** – List Conditional

For **P**-type conditions, the **ONERROR** functionality is used to provide additional detail about which component of the set was missing if Data Audit is enabled during the execution of a map using this schema.

## Using the X12N example files

There are no maps included in the X12N example. You can select any of the provided schemas to use in validation map, or maps created to transform to, or from, the supported versions and transactions.

The following table shows the versions and transactions supported by the X12N example schemas:

Version and ID	Schema name	Transaction Set ID	Document Title
003051X	x12n_3050	835	Health Care Claim Payment/Advice
003070X074	x12n_3070	186	Laboratory Results Report for Insurance Underwriting Requirements
003070X105			Paramedical Results Report for Insurance Underwriting Requirements
003070X023		270	Health Care Eligibility/Benefit Inquiry and Information Response
003070X023		271	Health Care Eligibility/Benefit Inquiry and Information Response
003070X069		276	Health Care Claim Status Request and Response
003070X069		277	Health Care Claim Status Request and Response
003070X070			Health Care Payer Unsolicited Claim Status
003070X070A1			Addenda to Health Care Payer Unsolicited Claim Status
003070X071			Health Care Claim Request for Additional Information
003070X004		278	Health Care Services Request for Review and Response
003070X058			Health Care Services Review - Notification
003070X020		834	Benefit Enrollment and Maintenance
003070X052		835	Health Care Claim Payment/Advice
003070X064		837	Health Care Claim: Institutional
003070X065		837	Health Care Claim: Dental
003070X066		837	Health Care Claim: Professional
004010X086	x12n_4010	148	Doctor's First Report of Injury
004010X092		270	Health Care Eligibility/Benefit Inquiry and Information Response
004010X092A1			Addenda to Health Care Eligibility/Benefit Inquiry and Information Response
004010X040		271	Unsolicited Health Care Eligibility/Benefit Roster
004010X092			Health Care Eligibility/Benefit Inquiry and Information Response
004010X093		276	Health Care Claim Status Request and Response
004010X093A1			Addenda to Health Care Claim Status Request and Response
004010X093		277	Health Care Claim Status Request and Response
004010X093A1			Addenda to Health Care Claim Status Request and Response
004010X059		278	Health Care Services Review Inquiry/Response
004010X094			Health Care Claim Services Request for Review and Response
004010X094A1			Addenda to Health Care Claim Services Request for Review and Response
004010X094A1			Addenda to Health Care Claim Services Request for Review and Response
004010X111			Health Care Services Review - Notification
004010X047		811	Consumer Service Provider Insurance Billing
004010X061		820	Payroll Deducted and Other Group Premium Payment for Insurance Products
004010X061A1			Addenda to Payroll Deducted and Other Group Premium Payment for Insurance Products
004010X226			EPN STP 820
004010X256			EPN STP 820 for Straight-Through-Processing
004010X282			EPN STP 820 for Straight-Through-Processing
004010X309			EPN STP 820 for Straight-Through-Processing
004010X047		824	Consumer Service Provider Insurance Billing
004010X161			Implementation Guide and Application Reporting
004010X095		834	Benefit Enrollment and Maintenance
004010X095A1			Addenda to Benefit Enrollment and Maintenance
004010X091		835	Health Care Claim Payment/Advice
004010X091A1			Addenda to Health Care Claim Payment/Advice
004010X067		837	Property and Casualty Medical Bill Report to Regulatory Agencies
004010X096			Health Care Claim: Institutional
004010X096A1			Addenda to Health Care Claim: Institutional
004010X097			Health Care Claim: Dental
004010X097A1			Addenda to Health Care Claim: Dental
004010X098			Health Care Claim: Professional
004010X098A1			Addenda to Health Care Claim: Professional
004040X122	x12n_4040	269	Health Care Benefit Coordination Verification Request and Response
004040X167		277	Health Care Claim Acknowledgement
004050X168	x12n_4050	269	Health Care Benefit Coordination Verification Request and Response
004050X138		270	Health Care Eligibility/Benefit Inquiry and Information Response
004050X138		271	Health Care Eligibility/Benefit Inquiry and Information Response
004050X103		274	Health Care Provider Information
004050X109			Health Care Provider Directory
004050X185			Health Care Provider Inquiry and Information Response
004050X253			Health Care Provider Information

004050X151		275	Additional Information To Support a Health Care Claim or Encounter
004050X139		276	Health Care Claim Status Request and Response
004050X139		277	Health Care Claim Status Request and Response
004050X150			Health Care Claim Request for Additional Information
004050X164			Request for Medical Information and Reports in Support of a Disability or Workers' Compensation Claim
004050X140		278	Health Care Services Request for Review and Response
004050X137		820	Payroll Deducted and Other Group Premium Payment for Insurance Products
004050X166		824	Implementation Guide Reporting
004050X125		834	Benefit Enrollment and Maintenance
004050X124		835	Health Care Claim Payment/Advice
004050X141		837	Health Care Claim: Institutional
004050X142			Health Care Claim: Dental
004050X143			Health Care Claim: Professional
004050X156			Health Care Service Data Reporting
004060X201	x12n_4060	811	State Agency Insurance Reporting
005010X187	x12n_5010	269	Health Care Benefit Coordination Verification Request and Response
005010X279		270	Health Care Eligibility/Benefit Inquiry and Information Response
005010X279A1			Errata for [Addenda to] Health Care Eligibility/Benefit Inquiry and Information Response
005010X279		271	Health Care Eligibility/Benefit Inquiry and Information Response
005010X279A1			Errata for [Addenda to] Health Care Eligibility/Benefit Inquiry and Information Response
005010X210		275	Additional Information To Support a Health Care Claim or Encounter
005010X211			Additional Information To Support a Health Care Services Review
005010X212		276	Health Care Claim Status Request and Response
005010X212		277	Health Care Claim Status Request and Response
005010X213			Health Care Claim Request for Additional Information
005010X214			Health Care Claim Acknowledgment
005010X227			Request for Information in Support of a Disability Claim
005010X228			Health Care Claim Pending Status Information
005010X215		278	Health Care Services Review Inquiry/Response
005010X216			Health Care Services Review - Notification
005010X217			Health Care Services Request for Review and Response
005010X217			Health Care Services Request for Review and Response
005010X218		820	Payroll Deducted and Other Group Premium Payment for Insurance Products
005010X306			Health Insurance Exchange Related Payments
005010X186		824	Application Reporting for Insurance
005010X186A1			Errata for [Addenda to] Application Reporting for Insurance
005010		831	Application Control Totals
005010X220		834	Benefit Enrollment and Maintenance
005010X220A1			Errata for [Addenda to] Health Care Benefit Enrollment and Maintenance
005010X307			Health Insurance Exchange: Enrollment
005010X318			Plan Member Reporting
005010X221		835	Health Care Claim Payment/Advice
005010X221A1			Errata for [Addenda to] Health Care Claim Payment/Advice
005010X222		837	Health Care Claim: Professional
005010X222A1			Errata for [Addenda to] Health Care Claim: Professional
005010X222A2			Second Type 1 Errata for [Addenda to] Benefit Enrollment and Maintenance
005010X223			Health Care Claim: Institutional
005010X223A1			First Type 1 Errata for [Addenda to] Health Care Claim: Institutional
005010X223A2			Second Type 1 Errata for [Addenda to] Health Care Claim: Institutional
005010X223A3			Third Type 1 Errata for [Addenda to] Health Care Claim: Institutional
005010X224			Health Care Claim: Dental
005010X224A1			First Type 1 Errata for [Addenda to] Health Care Claim: Dental
005010X224A2			Second Type 1 Errata for [Addenda to] Health Care Claim: Dental
005010X224A3			Third Type 1 Errata for [Addenda to] Health Care Claim: Dental
005010X225			Health Care Service: Data Reporting
005010X225A1			First Type 1 Errata for [Addenda to] Health Care Service: Data Reporting
005010X225A2			Second Type 1 Errata for [Addenda to] Health Care Service: Data Reporting
005010X291			Health Care Predetermination: Professional
005010X291A1			Errata for [Addenda to] Health Care Predetermination: Professional
005010X292			Health Care Predetermination: Institutional
005010X292A1			Errata for [Addenda to] Health Care Predetermination: Institutional
005010X298			Post Adjudicated Claims Data Reporting: Professional
005010X298A1			First Type 1 Errata for [Addenda to] Post Adjudicated Claims Data Reporting: Professional
005010X299			Post Adjudicated Claims Data Reporting: Institutional
005010X299A1			First Type 1 Errata for [Addenda to] Post Adjudicated Claims Data Reporting: Institutional
005010X300			Post Adjudicated Claims Data Reporting: Dental
005010X300A1			First Type 1 Errata for [Addenda to] Post Adjudicated Claims Data Reporting: Dental
005010X230		997	Functional Acknowledgment for Health Care Insurance

005010X231		999	Implementation Acknowledgment for Health Care Insurance
005010X231A1			Errata for [Addenda to] Implementation Acknowledgment for Health Care Insurance
005040X273	x12n_5040	269	Health Care Benefit Coordination Verification Request and Response
005040X254		275	Additional Payer Health Care Information
005050X274	x12n_5050	275	Personal Health Record Transfer Between Health Plans
006010X288	x12n_6010	820	International ACH Transaction
006020X280	x12n_6020	270	Health Care Eligibility/Benefit Inquiry and Information Response
006020X280		271	Health Care Eligibility/Benefit Inquiry and Information Response
006020X296			Unsolicited Health Care Eligibility/Benefit Roster
006020X206		274	Health Care Provider Information
006020X207			Health Care Provider Directory
006020X297			Provider Electronic Service Information Discovery
006020X275		275	Additional Information To Support a Health Care Claim or Encounter
006020X278			Additional Information To Support a Health Care Services Review
006020X314			Additional Information To Support a Health Care Claim or Encounter
006020X316			Additional Information To Support a Health Care Services Review
006020X267		276	Health Care Claim Status Request and Response
006020X267		277	Health Care Claim Status Request and Response
006020X268			Health Care Claim Request for Additional Information
006020X269			Health Care Claim Acknowledgment
005040X270			Health Care Claim Pending Status Information
006020X313			Health Care Claim Request for Additional Information
006020X264		278	Health Care Services Review Inquiry/Response
006020X265			Health Care Services Review - Notification
006020X266			Health Care Services Request for Review and Response
006020X315			Health Care Services Request for Review and Response
006020X284		820	Payroll Deducted and Other Group Premium Payment for Insurance Products
006020X257		824	Application Reporting for Insurance
006020X304		832	Health Care Fee Schedule
006020X283		834	Benefit Enrollment and Maintenance
006020X258		835	Health Care Claim Payment/Advice
006020X259		837	Health Care Claim: Professional
006020X260			Health Care Claim: Institutional
006020X261			Health Care Claim: Dental
006020X262			Health Care Service: Data Reporting
006020X336			Post-adjudicated Claims Data Reporting: Professional
006020X337			Post-adjudicated Claims Data Reporting: Institutional
006020X338			Post-adjudicated Claims Data Reporting: Dental
006020X305		838	Provider Enrollment for EDI Services
006020X290		999	Implementation Acknowledgment for Health Care Insurance
008010X		270	
008010X344	X12n8010	271	Premium Payment Grace Period Notification (271)*

The following table shows the 7030-8030 versions and transactions supported by the X12N example schemas:

ID	Schema name	Transaction Set ID	Document Title
X109	X12n_8010	274	Health Care Provider Directory
	X12n_8020		
X253	X12n_8010	275	Health Care Provider Information
X274	X12n_8010		Personal Health Record Data Transfer Between Health Plans
	X12n_8020	837	
X298	X12n_8010		Post-adjudicated Claims Data Reporting: Professional
	X12n_8020		
X299	X12n_8010		Post-adjudicated Claims Data Reporting: Institutional
	X12n_8020		
X300	X12n_8010	835	Post-adjudicated Claims Data Reporting: Dental
	X12n_8020		
X305	X12n_8020		Provider Enrollment for EDI Services
X318	X12n_8020		Plan Member Reporting
X321	X12n_8020		The Application Reporting for Insurance
X322	X12n_7030	835	Health Care Claim Payment/Advice
	X12n_8010		
	X12n_8020		
	X12n_8030		
X323	X12n_7030	837271	Health Care Claim: Professional
	X12n_8010		
	X12n_8020		
	X12n_8030		
X324	X12n_7030		Health Care Claim: Institutional
	X12n_8010		

	X12n_8020		
	X12n_8030		
X325	X12n_7030		Health Care Claim: Dental
	X12n_8010		
	X12n_8020		
X326	X12n_7030		Health Care Service: Data Reporting
	X12n_8010		
	X12n_8020		
X327	X12n_7030	278	Health Care Services Review - Inquiry and Response
	X12n_8010		
	X12n_8020		
X328	X12n_7030		Health Care Services Review Notification and Acknowledgment
	X12n_8010		
	X12n_8020		
X329	X12n_7030	276/277	Health Care Claim Status Request and Response
	X12n_8010		
	X12n_8020		
X330	X12n_7030	277	Health Care Claim Acknowledgment
	X12n_8010		
	X12n_8020		
X331	X12n_7030		Health Care Claim Pending Status Information
	X12n_8010		
	X12n_8020		
X332	X12n_8030	270/271	Health Care Eligibility/Benefit Inquiry and Information Response
X333	X12n_7030	834	Benefit Enrollment and Maintenance
	X12n_8010		
	X12n_8020		
X340	X12n_7030	277	Health Care Claim Request for Additional Information
	X12n_8010		
	X12n_8020		
X341	X12n_7030	275	Additional Information to Support a Health Care Claim or Encounter
	X12n_8010		
	X12n_8020		
X342	X12n_7030	278	Health Care Services Review - Request for Review and Response
	X12n_8010		
	X12n_8020		
X343	X12n_7030	275	Additional Information to Support a Health Care Services Review
	X12n_8010		
	X12n_8020		
X344	X12n_8010	271	Premium Payment Grace Period Notification
	X12n_8020		
X346	X12n_7030	834	Health Insurance Exchange: Enrollment
	X12n_8010		
	X12n_8020		

## External Code List Utilities example

The External Code List Utilities example contains a utility map that loads the code set database with the external code sets that are supported in the pack into a MongoDB NoSQL database.

These maps require that a MongoDB Server is accessible. This example also contains additional maps that can convert code set source files into the json format needed to load the database, if updates are needed between Pack for Healthcare releases.

- [Terminology for NoSQL databases.](#)  
Terminology for NoSQL databases, and how code set support is implemented for the Pack for Healthcare.
- [Locating](#)
- [Example components](#)
- [Using the example](#)

## Terminology for NoSQL databases.

Terminology for NoSQL databases, and how code set support is implemented for the Pack for Healthcare.

All the code source sets will be loaded into one Database (the default used is hipaa\_code\_sets). Each code source will have its own collection. The collection contains multiple documents, with each document containing an individual code source value, with a description and optional start and end dates.

## Locating

The Code List example is located in the hipaaExamplesCodeSets.zip project.

## Example components

- [Configuration files](#)

The configuration files are used for the database server location and any user credentials needed to access the database server.

- [Maps to load or delete code set information](#)

These maps load or delete code set information:

- [Maps to convert code set data](#)

These maps all begin with T5\_ (for type 5) or T6\_ (for type 6) and can be used to create the json files necessary to use in the database load maps included in this example.

- [Schemas](#)

- [Data Files](#)

The External Code List Utilities example contains the original command or tab-delimited code source values and JSON format files of these values for the code sets for which we support code value validation rules.

## Configuration files

The configuration files are used for the database server location and any user credentials needed to access the database server.

The following configuration files are used:

- hipaa\_code\_set.mrc
- hipaa\_code\_set.mrn

The server location is defaulted to mongodb://localhost:27017 in this example but should be updated if your server must be accessed through a different URL, or you have the appropriate database access information for JDBC before running the example maps.

## Maps to load or delete code set information

These maps load or delete code set information:

- load\_code\_set\_mongo – loads a single code set collection with documents for each code value in the input json file.
- load\_all\_code\_sets – loads all the external code set collections that are specified in the code\_set\_json\_list.txt input file, but calling load\_code\_set\_mongo.
- delete\_code\_set\_values – deletes all the documents from a single collection. The collection itself has not been removed from the database.
- delete\_all\_code\_set\_values – deletes all the documents from collections specified in the code\_set\_json\_list.txt input file.
- load\_code\_set\_jdbc – loads a single code set collection with documents for each code value in the input json file.
- load\_all\_code\_sets\_jdbc – loads all the external code set collections that are specified in the code\_set\_json\_list.txt input file, but calling load\_code\_set\_jdbc.
- delete\_code\_set\_values\_jdbc – deletes all the documents from a single collection. The collection itself has not been removed from the database.
- delete\_all\_code\_set\_values\_jdbc – deletes all the documents from collections specified in the code\_set\_json\_list.txt input file.

## Maps to convert code set data

These maps all begin with T5\_ (for type 5) or T6\_ (for type 6) and can be used to create the json files necessary to use in the database load maps included in this example.

The Type 5 maps also contain a code source number. The External Code Set or Source number is the number that is used for this code list in the HIPAA Implementation Guides. For example, the map that starts T5\_022, is a map for the code values for Code source 22, States and Provinces. The Type 6 map names contain the codes to indicate the line of service location and line of service category for this set of code list values.

## Schemas

- code\_set\_source – The schema that defines the command or tab-delimited input formats for the code set sources included in the example
- code\_set\_template.json - the JSON template used in creating and parsing code source json data files.

## Data Files

The External Code List Utilities example contains the original command or tab-delimited code source values and JSON format files of these values for the code sets for which we support code value validation rules.

## Using the example

- [Using the load or delete code set maps](#)
- [Using the maps to convert code set data](#)

These maps are run if you need to update a full set of code values due to code source changes after the release of the Pack for Healthcare.

## Using the load or delete code set maps

1. Update the Mongo Server location and/or credentials in configuration variables if defaults are not correct for your design environment.
  - Design Server - Update configuration variable host with correct server location. Update variables Password and userID with correct credentials.
2. Build all maps.
3. Update code\_set\_json\_list.txt if you do not want to load all code set collections, by deleting the names of code source json files that you do not want to load or remove documents from.
4. Execute load\_all\_code\_sets or delete\_all\_code\_set\_values.

## Using the maps to convert code set data

These maps are run if you need to update a full set of code values due to code source changes after the release of the Pack for Healthcare.

- Update the \*.txt file with the updated code source list.
- Build the T5\_5\* map that corresponds to your updated code source.
- Run the T5\_5\* map that corresponds to your updated code source.

## Type 7 User Exit example

This example contains a Java Program that can be used as a guide to help with the development and requirements of a Java custom user exit for Type 7 validation.

- [\*\*Location\*\*](#)  
In the Design Server, the Type 7 User Exit example is contained in the hipaaType7ExampleConfig tar.gz or zip file, depending upon the run time platform.
- [\*\*Example components\*\*](#)  
The Type 7 User Exit example contains these components:
- [\*\*Example details\*\*](#)  
The class\_name for TestJExit is com.ibm.dtx.T7jexit.TestJExit.
- [\*\*Using the Type 7 user exit example\*\*](#)  
This type of custom user exit is executed at run time by HIPAA Data Compliance, based on configuration in the hipaa\_data\_compliance\_parameter.json file.

## Location

In the Design Server, the Type 7 User Exit example is contained in the hipaaType7ExampleConfig tar.gz or zip file, depending upon the run time platform.

This is not a project to be imported into the Design server, but it contains the example components.

## Example components

The Type 7 User Exit example contains these components:

- Type7Exit.jar file created using example source, found in the \jars folder
- Java source files, found in \src\com\ibm\dtx\T7jexit, include
  - Log.java – Java source containing a logging class using coded log methods.
  - Log2.java – Java source containing a logging class using base java logging utilities.
  - TestJExit.java – Java source containing example validation methods.
  - T7ErrReturn.java – Java source containing a class defining the expected XML error return structures and processing methods.
- Sample data file – found in \data folder, hipaa\_5010\_x222a1\_837p\_type7\_sample.dat – that can be used when running the sample to confirm the error is reported

This example user exit does not contain validations related to any published Companion Guide.

## Example details

The class\_name for TestJExit is com.ibm.dtx.T7jexit.TestJExit.

The validation method (function) is Type7\_Checks and performs the following checks:

- Checks that the Subscriber Eligibility Segment EQ and the repeating element EQ01 occur less than 30 times in 005010X279A1 270 transaction sets (HIPAA Allows 99).
- For 837 transactions, checks the DTM and DTP dates for future dates and checks that DTP dates are not prior to DTM dates and also checks to ensure that the number of claims in the transaction set is less than 100.
- For 837 transactions with a GS03 (application receiver code) of STATE, limit the dollar amount in AMT02 to \$500 when AMT01 is F5 in the CLM loop.

The validation method Type7\_Checks receives the following two inputs from the hipaa\_data\_compliance maps:

1. Delims– a string that indicates the delimiters that are used in the transaction set. It is used for parsing the segments and elements. It is passed in the following format:  
 Element Separator (ISA Element separator)  
 Repetition Separator (ISA11)  
 Component Element Separator (ISA16)  
 Segment Terminator (ISA Terminator)
2. hipaaData- a string of EDI HIPAA data generated during the run time of the compliance check application. It contains the transaction set data, including the ST/SE segments, but the loop ID is also included as the first element in each segment. If the parameter ExitPartnerDetail is set to true, then the hipaaData string will have limited ISA and GS information before the transaction set data.

## Using the Type 7 user exit example

This type of custom user exit is executed at run time by HIPAA Data Compliance, based on configuration in the hipaa\_data\_compliance\_parameter.json file.

The ExitName and ExitFunction values configured are specific to this example.

As a prerequisite, in Design Server Execute the setup scripts found in hipaaType7ExampleConfig.tar.gz or zip file, dependent on the run time platform.

Follow these steps to use the example.

1. Update hipaa\_data\_compliance\_parameter.json, ValidationLevels, to enable Type7.

```
"ValidationLevels": {
 "Type7": {
 "Enabled": true,
 "RejectLevel": "Errors",
 "ExitType": "Java",
 "ExitName": "com.ibm.dtx.T7jexit.TestJExit",
 "ExitFunction": "Type7_Checks"
 }
}
```

2. Update hipaa\_data\_compliance map input 2 to use the hipaa\_5010\_x222a1\_837p\_type7\_sample.dat file.
3. Build all maps for hipaa\_data\_compliance.
4. Execute hipaa\_data\_compliance
5. View the resulting 999 acknowledgment, which will show the type 7 error reported.

## ClaimToExposure tracking example

In the era of COVID-19 many organizations will find it helpful to be able to do a quick scan of the data they already have access to in order to quickly isolate individuals with diagnosis codes of COVID-19 or indicating COVID-19 in order to take quick action.

This example shows how HIPAA 837 claims data can be used to provide an early warning report. Based on an input X12 837P transaction, this map will create an X12 standard 148 transaction, which is an EDI Report of Injury, Illness or Incident and contains basic demographic information on the patient and a list of diagnosis codes on the claim. The list of ICD-10 diagnosis codes can be modified based on current CDC guidance and organizational preferences.

- [Location](#)
- [ClaimToExposureTracking example maps](#)  
 There is one example map in this example: ClaimToExposureTracking
- [Claim to exposure tracking map details](#)

## Location

The Claim to Exposure Tracking example is located in the hipaaExamples.zip project.

## ClaimToExposureTracking example maps

There is one example map in this example: ClaimToExposureTracking

Based on an input 837P, X12 837 Professional Claim transactions, used for HIPAA and constrained by an X12N TR3 (or implementation guide) 005010X222A1, this map will create an X12 standard 148 transaction, which is an EDI Report of Injury, Illness or Incident and contains basic demographic information on the patient and a list of diagnosis codes on the claim. This same information is also generated in a simple, non-EDI text report.

## Claim to exposure tracking map details

The following are inputs to the Claim837P\_to\_ExposureReport map:

- Input 1 - HIPAA EDI claims data file
- Input 2 - A list of ICD-10 CM diagnosis codes that you want to scan. You can configure this list to fit your needs.
- Input 3 - a list of three control numbers used to create the EDI 148 transaction. One is for the interchange envelope, one is for the Functional Group envelope, and one is for the Transaction set envelope. In a production environment, these will probably be changed and looked up from a database or another system.
- Input 4 - list of diagnosis codes and their descriptions, the ICD-10 CM data to display on reports.

The outputs are the following:

- Output 1 - EDI Report – a 148 transaction, which is an EDI Report of Injury, Illness, or Incident
- Output 2 - Text Based Report

The X12 148, EDI Report, has one HL (Hierarchical Reporting) loop per claim that qualifies for inclusion on the report. It contains basic demographic data, the name of the patient and the subscriber (if different), the patient's address, and the primary and any secondary diagnoses. This report only includes claims with one diagnosis code, which matches one in input 2, but for a selected claim, all other diagnosis codes are included.

The simple text-based report includes the same basic claim information included on the X12 148 report, but with labels and diagnosis descriptions, so it is more readable for a user not familiar with X12.

Note: This example produces a report file that contains personal health information (as does the input claim itself), so ensure that regulatory policy and company policies are followed when creating and sharing these reports.

## X12N personal health records example

To support ways for the payer community to leverage existing B2B infrastructure while also beginning to work with HL7 CDA R2 and FHIR (in Json or XML), X12 has a draft implementation guide that describes the use of the AXC X12 Patient Information (275) transaction set for the business purpose of transferring Personal Health Record (PHR) information between different health plans.

This example has executable maps to generate an X12 275 transaction with imbedded PHR data in JSON format or extract PHR data from an existing 275 transaction.

- [Example location](#)
- [Example maps](#)

In the X12n Personal Health Record (PHR) Example, two executable maps are used to generate an X12 275 transaction with imbedded PHR data in JSON format or extract PHR data from an existing 275 transaction.

## Example location

The Personal Health Record Example is located in the hipaaExamples.zip project.

## Example maps

In the X12n Personal Health Record (PHR) Example, two executable maps are used to generate an X12 275 transaction with imbedded PHR data in JSON format or extract PHR data from an existing 275 transaction.

This transaction, identified currently as 007060X274, is similar to the 275 clinical attachment but with a simplified loop structure. Also, the current draft version of this implementation guide specifies HL7 CDA as the clinical payload, in this example we are using JSON.

The example contains a sample schema with the draft transaction. This schema is a simple 'ruleless' type schema with no implementation guide based rules except for any required basic structural definition.

There are two executable maps:

- `create_275_with_PHR_data` - generates a 275 with imbedded PHR data
- `extract_PHR_data_from_275` - extracts Personal Health Record data from an existing 275 transaction.

Both maps use the FHIR sample claim in JSON format for the PHR data that will be added/extracted from the 275 BDS segment.

## Performance tuning

The HIPAA X12 data structures are hierarchical in nature, and the semantic and business rules can be complex. Some of the data files can be quite large. Given these factors, it's clear that performance is an important consideration when processing HIPAA data. There are various techniques that can be used to improve the performance of HIPAA data processing within Transformation Extender. These techniques involve separating, combining, validating, and transforming data. Some of these methods are based on the standard IBM performance tuning recommendations.

Other areas for performance improvement require an analysis of site-specific and task-specific requirements and the deployment of particular architectures that optimize performance for those particular requirements.

In particular, the processing associated with HIPAA Compliance Check and reporting tends to be 'expensive' from a data processing perspective. As the HIPAA Compliance Check is highly configurable, it is possible to see large variations of performance that are based on configuration. This documentation discusses some of the processing options that are known to require substantial resources along with suggestions for their use and configuration.

- [\*\*Duplicate processing\*\*](#)

The various processing steps can include overlapping Transformation Extender functionality. Avoid overlapping functionality unless, potentially, it is necessary.

- [\*\*Unnecessary processing\*\*](#)

To increase performance, avoid all unnecessary processing of data.

- [\*\*Over reporting\*\*](#)

To maximize performance, generate only those reports that are needed by the business consumers of this information.

---

## Duplicate processing

The various processing steps can include overlapping Transformation Extender functionality. Avoid overlapping functionality unless, potentially, it is necessary.

For example, if you are developing HIPAA EDI data and then transforming that data into another format, there are multiple validation opportunities.

- Formal validation can be performed using the HIPAA Compliance check.

- Informal validation can be performed by use of the various HIPAA-level schemas that are used in the transformation map.

As an example, consider an organization that is performing HIPAA compliance checks to WEDI/SNIP level 4. If a Transformation Extender map that uses the hipaa\_x12 schema is used to transform the data, validation is performed twice.

It is performed once as part of the Compliance Check with the formal HIPAA acknowledgment reporting. Then validation is performed again to WEDI/SNIP level 4 by use of the schema. In this case, the data is already validated before transformation, and a schema with less validation might be used. To increase performance, use the hipaa\_x12\_ruleless schema in the transformation step.

---

## Unnecessary processing

To increase performance, avoid all unnecessary processing of data.

For example, the WEDI/SNIP type 5 capabilities in the HIPAA Compliance Check module can be set to check the values of external code set values that are stored in a database, against the qualified values in the data. Depending on your business requirements, this higher level checking might not be necessary.

Due to the large amount of resources that are required to perform the type 5 check, do not perform the check unless your intention is to reject such data.

Similarly, if you want to process all data with the correct HIPAA structure (WEDI/SNIP type 2), do not perform the more costly type 4 validation. Type 4 validation includes cross-segment summing and other cross-segment checks. The rules that are involved in these levels of checks require the build-up of multiple pieces of information across more than one segment of data. Therefore, these checks require more resources than the type 2 checks require.

Another way to increase performance is to use Claim Level Rejection processing only when it is necessary to meet the business requirement.

The ability to process a subset of a batch of 837 claims is important in some environments but not in all. If there is a business requirement to process all valid claim loops and report on all invalid claim loops within a single transaction set then Claim Level Rejection is necessary. However, the processing necessary to accomplish that task is complex and care must be taken to use Claim Level Rejection only when the business requirement demands this level of processing.

---

## Over reporting

To maximize performance, generate only those reports that are needed by the business consumers of this information.

For example, a translated acknowledgment report is available in the Transformation Extender Platform. This report shows a graphical representation of the errors in the data, which is shown in the same hierarchical structure. This is an informative report for you to generate if you are a non-EDI fluent user. It is often a good tool to use when the Transformation Extender Platform system is initially configured. In some cases, the technical implementation team might not be familiar with ASC X12 999. However, your business users might find the structured 999 acknowledgment sufficient for both computer-readable and human-readable use. Therefore, you might not need a duplicate version of the same information. In such cases, verify that these reports are needed after the initial implementation of the product.

---

## Acknowledgment errors

These lists show HIPAA Compliance Checker acknowledgment error codes.

- [\*\*AK304 segment level\*\*](#)

- [\*\*IK304 error code\*\*](#)

- [\*\*AK403 element level\*\*](#)

- [\*\*IK403 error code\*\*](#)

- [\*\*AK502 transaction set level\*\*](#)

- [\*\*AK905 functional group level\*\*](#)

- [\*\*TA105 interchange level\*\*](#)

- [\*\*TED01 - error reporting/acknowledgement process\*\*](#)

---

## AK304 segment level

<b>Code</b>	<b>Definition</b>
AK304 = 1	Unrecognized segment ID
AK304 = 2	Unexpected segment
AK304 = 3	Mandatory segment missing
AK304 = 4	Loop Occurs Over Maximum Times
AK304 = 5	Segment Exceeds Maximum Use
AK304 = 6	Segment Not in Defined Transaction Set
AK304 = 7	Segment Not in Proper Sequence
AK304 = 8	Segment Has Data Element Errors

---

## IK304 error code

<b>Code</b>	<b>Definition</b>
IK304 = 1	Unrecognized Segment ID
IK304 = 2	Unexpected Segment
IK304 = 16	Implementation dependent seg missing
IK304 = 4	Loop occurs > max allowed
IK304 = 5	Segment exceeds max use
IK304 = 14	Implementation "not used" seg pres
IK304 = 7	Segment not in sequence
IK304 = 8	Segment has data element errs
IK304 = 17	Implementation loop occurs < min times
IK304 = 18	Implementation segment below min use
IK304 = 19	Impl'n dependent not used seg present

---

## AK403 element level

<b>Code</b>	<b>Definition</b>
AK403 = 1	Required data element missing
AK403 = 2	Conditional required data element missing
AK403 = 3	Too many data elements
AK403 = 4	Data element too short
AK403 = 5	Data element too long
AK403 = 6	Invalid character in data element
AK403 = 7	Invalid code value
AK403 = 8	Invalid date
AK403 = 9	Invalid time
AK403 = 10	Exclusion condition violated
AK403 = 12	Too many repetitions
AK403 = 13	Too many components
AK403=I10	Implementation "Not Used" data element present
AK403=I11	Implementation too few repetitions
AK403=I12	Implementation pattern match failure
AK403=I13	Implementation dependent "Not Used" data element present
AK403=I6	Code value not used in implementation
AK403=I9	Implementation dependent data element missing

---

## IK403 error code

<b>Code</b>	<b>Description</b>
IK403 = 19	Impl'n dependent data element missing
IK403 = 2	Conditional required data element missing
IK403 = 13	Too many components
IK403 = 4	Data element too short
IK403 = 5	Data element too long
IK403 = 6	Invalid character in data element
IK403 = 16	Code value not used in implementation
IK403 = 8	Invalid date
IK403 = 9	Invalid time
IK403 = 10	Exclusion Condition Violated
IK403 = 12	Too many repetitions
IK403 = I10	Implementation "not used" data element present

<b>Code</b>	<b>Description</b>
IK403 = I11	Implementation too few repetitions
IK403 = 12	Implementation Pattern Failure Match
IK403 = 13	Implementation dependent "not used" element present

---

## AK502 transaction set level

<b>Code</b>	<b>Definition</b>
AK502 = 1	Transaction set not supported
AK502 = 2	Transaction set trailer missing
AK502 = 3	Transaction set control number in header and trailer do not match
AK502 = 4	Number of included segments does not match actual count
AK502 = 5	One or more segments in error
AK502 = 6	Missing or invalid transaction set identifier
AK502 = 7	Missing or invalid transaction set control number
AK502 = 8	Authentication key name unknown
AK502 = 9	Encryption key name unknown
AK502 = 10	Requested service (authentication or encryption) not available
AK502 = 11	Unknown security recipient
AK502 = 12	Incorrect message length (encryption only)
AK502 = 13	Message authentication code failed
AK502 = 15	Unknown security originator
AK502 = 16	Syntax error in decrypted text
AK502 = 17	Security not supplied
AK502 = 18	Transaction set not in functional group
AK502 = 19	Invalid transaction set implementation convention reference
AK502 = 23	Transaction set control number not unique within the functional group
AK502 = 24	S3E security end segment missing for S3S security start segment
AK502 = 25	S3S security start segment missing for S3E security end segment
AK502 = 26	S4E security end segment missing for S4S security start segment
AK502 = 27	S4S security start segment missing for S4E security end segment
AK502=I6	Implementation convention not supported

---

## AK905 functional group level

<b>Code</b>	<b>Definition</b>
AK905 = 1	Functional group not supported
AK905 = 2	Functional group version not supported
AK905 = 3	Envelope trailer missing
AK905 = 4	Group control number in the functional group header and trailer do not agree
AK905 = 5	Number of included transaction sets does not match actual count
AK905 = 6	Group control number violates syntax
AK905 = 10	Authentication key name unknown
AK905 = 11	Encryption key name unknown
AK905 = 12	Requested service (authentication or encryption) not available
AK905 = 13	Unknown security requirement
AK905 = 14	Unknown security originator
AK905 = 15	Syntax error in decrypted text
AK905 = 16	Security not supported
AK905 = 17	Incorrect message length (encryption only)
AK905 = 18	Message authentication code failed
AK905=19	Functional Group Control Number not unique within interchange
AK905 = 23	S3E security end segment missing for S3S security start segment
AK905 = 24	S3S security start segment missing for S3E security end segment
AK905 = 25	S4E security end segment missing for S4S security start segment
AK905 = 26	S4S security start segment missing for S4E security end segment

---

## TA105 interchange level

<b>Code</b>	<b>Definition</b>
TA105 = 000	No error
TA105 = 001	The interchange control number in the header and trailer do not match

<b>Code</b>	<b>Definition</b>
TA105 = 002	The standard as noted in the control standards identifier is not supported
TA105 = 003	This version of the controls is not supported
TA105 = 004	The segment terminator is invalid
TA105 = 005	Invalid interchange ID qualifier for sender
TA105 = 006	Invalid interchange sender ID
TA105 = 007	Invalid interchange ID qualifier for receiver
TA105 = 008	Invalid interchange receiver ID
TA105 = 009	Unknown interchange receiver ID
TA105 = 010	Invalid authorization information qualifier value
TA105 = 011	Invalid authorization information value
TA105 = 012	Invalid security information qualifier value
TA105 = 013	Invalid security information value
TA105 = 014	Invalid interchange date value
TA105 = 015	Invalid interchange time value
TA105 = 016	Invalid interchange standards identifier value
TA105 = 017	Invalid interchange version ID value
TA105 = 018	Invalid interchange control number value
TA105 = 019	Invalid acknowledgement requested value
TA105 = 020	Invalid test indicator value
TA105 = 021	Invalid number of included groups value
TA105 = 022	Invalid control structure
TA105 = 023	Improper (premature) end-of-file (transmission)
TA105 = 024	Invalid interchange content (e.g. invalid GS segment)
TA105 = 025	Duplicate interchange control number
TA105 = 026	Invalid data element separator
TA105 = 027	Invalid component element separator
TA105 = 028	Invalid delivery date in deferred delivery request
TA105 = 029	Invalid delivery time in deferred delivery request
TA105 = 030	Invalid deliver time code in deferred delivery request
TA105 = 031	Invalid grade of service code

## TED01 - error reporting/acknowledgement process

<b>Code</b>	<b>Definition</b>
TED01 = 010	Total out of balance
TED01 = 848	Incorrect data
TED01 = OTH	Used for reporting XML errors.
TED01 = 024	Other unlisted reasons.

## HL7 component

The documentation provided here describes the Pack for Healthcare HL7 component.

- [HL7 schemas](#)  
The schemas in the HL7 component, include definitions for the HL7 format.
- [HL7 component schemas](#)  
The HL7 component includes schemas with data definitions for the HL7 industry standard.
- [HL7 validation](#)
- [HL7 component examples](#)  
Examples are provided to demonstrate how to use the HL7 component.

## HL7 schemas

The schemas in the HL7 component, include definitions for the HL7 format.

The HL7 component provides health care and insurance payer organizations with an infrastructure that:

- enables compliance with government and industry mandates
- controls administrative costs
- streamlines business processes
- facilitates accuracy and timeliness of information
- offers a competitive advantage
- conforms to existing systems
- adapts to new technologies as they emerge
- integrates multiple systems and standards

# HL7 component schemas

The HL7 component includes schemas with data definitions for the HL7 industry standard.

This means that data that meets the specification requirements is considered valid, while data that does not meet the requirements is rejected. Other schema attributes include the following:

- compatibility with other IBM product offerings, such as Trading Manager
- flexibility in modifying or merging with other schemas
- ability to support reliable and efficient mapping
- easily maintainable

- **[hl7\\_v2\\_x schemas](#)**

The hl7\_v2\_x schemas provided in the HL7 component describe the data exchange formats associated with HL7.

- **[Using the Pack for HL7 schemas](#)**

The Transformation Extender user interface depicts the HL7 message structure as an expandable, contractible set of hierarchical components. The schema enforces compliance of HL7 transmission data with EDI Standard Integrity Testing rules.

## hl7\_v2\_x schemas

The hl7\_v2\_x schemas provided in the HL7 component describe the data exchange formats associated with HL7.

Note: In the hl7\_v2\_x schemas, x indicates the standard version (for example, hl7\_v2\_3, hl7\_v2\_4, etc.)

Most of the schemas in the pack include a category: **%\_Type\_Tree\_Information**. Double-click on the **%\_Description** or **%\_Revision\_History** items to see a narrative description of the schema or the schema revision history.

Some of the hl7\_v2\_x schemas are analyzed with warnings that indicate some components may not be distinguishable from other components. Such warnings can be ignored.

- **[hl7\\_v2\\_1 schema](#)**

This schema contains the data definitions used in the HL7 Version 2.1 standard.

- **[hl7\\_v2\\_2 schema](#)**

This schema contains the data definitions used in the HL7 Version 2.2 standard.

- **[hl7\\_v2\\_3 schema](#)**

This schema contains the data definitions used in the HL7 Version 2.3 standard.

- **[hl7\\_v2\\_4 schema](#)**

This schema contains the data definitions used in the HL7 Version 2.4 standard.

- **[hl7\\_v2\\_5 schema](#)**

This schema contains the data definitions used in the HL7 Version 2.5 standard.

- **[hl7\\_v2\\_5\\_1 schema](#)**

This schema contains the data definitions used in the HL7 Version 2.5.1 standard.

- **[hl7\\_v2\\_6 schema](#)**

This schema contains the data definitions used in the Health Level 7 (HL7) Version 2.6 standard. These data definitions include file, batch, and message formats. File Protocol HL7 is the top-level group in this schema.

- **[hl7\\_v2\\_7 schema](#)**

## hl7\_v2\_1 schema

This schema contains the data definitions used in the HL7 Version 2.1 standard.

The data definitions include file, batch, and message formats. File Protocol **HL7** is the top-level group in this schema.

## hl7\_v2\_2 schema

This schema contains the data definitions used in the HL7 Version 2.2 standard.

These data definitions include file, batch, and message formats. File Protocol **HL7** is the top-level group in this schema.

## hl7\_v2\_3 schema

This schema contains the data definitions used in the HL7 Version 2.3 standard.

These data definitions include file, batch, and message formats. File Protocol **HL7** is the top-level group in this schema.

The hl7\_v2\_3 schema analyzes with warnings that you can ignore, such as the following:

```
L199 - COMPONENT 10 may not be distinguishable from COMPONENT 14 that may follow in TYPE 'A06
ADT Message HL7' (warning)
```

## hl7\_v2\_4 schema

This schema contains the data definitions used in the HL7 Version 2.4 standard.

These data definitions include file, batch, and message formats. File Protocol **HL7** is the top-level group in this schema.

## hl7\_v2\_5 schema

This schema contains the data definitions used in the HL7 Version 2.5 standard.

These data definitions include file, batch, and message formats. File Protocol **HL7** is the top-level group in this schema.

## hl7\_v2\_5\_1 schema

This schema contains the data definitions used in the HL7 Version 2.5.1 standard.

These data definitions include file, batch, and message formats. File Protocol **HL7** is the top-level group in this schema.

## hl7\_v2\_6 schema

This schema contains the data definitions used in the Health Level 7 (HL7) Version 2.6 standard. These data definitions include file, batch, and message formats. File Protocol **HL7** is the top-level group in this schema.

The hl7\_v2\_6 schema analyzes with warnings that you can ignore, such as the following

L201 - Different data objects of COMPONENT 1 may not be distinguishable in  
TYPE 'Response\_LOOP O38 OPR Loop HL7' (warning)

L199 - COMPONENT 5 may not be distinguishable from COMPONENT 8 that may  
follow in TYPE 'A60 ADT Message HL7' (warning)

## hl7\_v2\_7 schema

This schema contains the data definitions used in the Health Level 7 (HL7) Version 2.7 standard. These data definitions include file, batch, and group formats. File Protocol **HL7** is the top-level group in this schema.

The hl7\_v2\_7 schema analyzes with warnings that you can ignore.

## Using the Pack for HL7 schemas

The Transformation Extender user interface depicts the HL7 message structure as an expandable, contractible set of hierarchical components. The schema enforces compliance of HL7 transmission data with EDI Standard Integrity Testing rules.

The rules check for the following:

- valid segments
- segment order
- maximum segment and loop occurrences
- element attributes (for example, size and data type)
- adherence to HL7 syntax rules as defined in the HL7 standard

Note: The examples used here are for example purposes only. They are not necessarily defined within the HL7 standard.

- [Schema data definitions](#)

The schemas contains the data definitions used in the HL7 standard. These data definitions include file, batch, and message formats. File protocol **HL7** is the top-level group in the schemas.

- [Logical grouping of fields](#)

Fields can be grouped to contain other related fields. For example, the **SendingApplication** HD Composite Field is HL7 data type HD. Grouped fields are delimited using the Comp Separator value.

## Schema data definitions

The schemas contains the data definitions used in the HL7 standard. These data definitions include file, batch, and message formats. File protocol **HL7** is the top-level group in the schemas.

An EDI message contains special characters called *delimiters*. These characters enable parsing of the message. The delimiters are:

• segment terminator	• subcomponent separator
• field separator	• repetition separator
• component separator	• escape character

The segment terminator, defined to be used with HL7 messages, is a carriage return (in ASCII, a hex 0D). The other delimiters are defined in the MSH segment, with the field separator in the fourth character position and the other delimiters occurring in the first field segment after the segment ID.

The following delimiter values specified in the MSH segment are used throughout the entire HL7 message:

MSH ~\&	
	= field separator
^	= comp separator
~	= repetition character
\	= EscapeCharacter separator
&	= subcomponent

The following example shows a segment (ERR) that contains all of the delimiters:

```
ERR|IPR^10^8~IPR||CWE^ERR03|W|CWE^ERR05||||CWE^ERR10|CWE^ERR11|PRN^PH^Add
res^034^444^5556666^Any Text^^^^^CWE&XTN Sub-Element^<cr>
```

- **Message structure**

The structure of the message is defined by message type and trigger event. For example, message type ACK and trigger event is A01. A message type may specify more than one trigger event code, but a trigger event may only be specified for a specific message type. The message type and trigger event are identified in the MSH segment. All components of a message may be defined as required, or optional.

---

## Message structure

The structure of the message is defined by message type and trigger event. For example, message type ACK and trigger event is A01. A message type may specify more than one trigger event code, but a trigger event may only be specified for a specific message type. The message type and trigger event are identified in the MSH segment. All components of a message may be defined as required, or optional.

A segment is a logical grouping of data fields. Segments might be either required or optional in a message. Segments can occur only once in a message, or they may be allowed to repeat. A unique three-character code is used to identify each segment. A segment terminator is used to identify the end of a segment. This segment terminator is the carriage return character for HL7. Segments can repeat, and they might be logically grouped together to form a loop within the message.

---

## Data fields

The data fields represent individual units of data found in the message. These fields are defined and re-used within segments and are given a data type. For example, the **DateTimeOfMessage** DTM Element Field is HL7 data type DTM. The data type identifies the characteristics of the data value and is used to validate the data. Data fields can be either required or optional in a segment. Each field is delimited within the segment using the Field Separator Message control value, which is the fourth character position in the MSH segment.

---

## Logical grouping of fields

Fields can be grouped to contain other related fields. For example, the **SendingApplication** HD Composite Field is HL7 data type HD. Grouped fields are delimited using the Comp Separator value.

When a field definition contains another field that defines a group of fields, the data type of the field within the definition is defined as a "Sub," or subcomponent in the schema. For example, ErrorCodeAndLocation ELD Composite contains **CodeIdentifyingError** CWE Sub. Both the ELD and CWE data types contain groups of related fields and can be used within a segment, or within another field. Subcomponent elements are delimited using the subcomponent separator value.

Fields can repeat and are identified as Repeat Field, such as, **ErrorLocation** ELD Repeat Field. Repeating data types are delimited using the repetition separator value.

---

## HL7 validation

Validating data that is formatted for HL7 2x or EDI poses several challenges. There are many required aspects to the syntax and semantic checking, and to report meaningful information based on flaws in the EDI data requires many checks.

- Is the data enveloped and, optionally, batched correctly?
- Are only the correct loops and segments in the data, and are they in the correct order?
- Within each segment are the fields in the correct order?
- Is the data in the field the correct size?

A recent addition to the HL7 offering is a set of XML schemas that describe the delimited EDI data as if it were XML tagged data. These schemas are available free of charge from HL7 for their members. HL7 member organizations can download these schemas and load them into the Transformation Extender HL7 validation directory structure and leverage them for data validation.

The HL7 2.x EDI format is the Intellectual Property of HL7. You must contact HL7 to use this feature.

- [Running the HL7 validation utility](#)

The HL7 Validation utility leverages the information in the XSD schemas provided by HL7 and applies the structure and data content rules from the schemas to the EDI data.

---

## Running the HL7 validation utility

The HL7 Validation utility leverages the information in the XSD schemas provided by HL7 and applies the structure and data content rules from the schemas to the EDI data.

First, the delimited HL7 2.x EDI data is transformed into its XML tagged equivalent. Then that XML data (called an XML instance document), is validated against the schema. This leverages the power of an XML parser for error reporting. An HTML formatted error report (with a very XML-centric description of errors) might be produced if desired, as well as an HL7 2.x EDI formatted ACK with a positive or negative acknowledgment indicator.

This utility also provides transformation of XML tagged Instance documents to their delimited EDI equivalents for the versions of HL7 2.x EDI that are supported by this implementation.

To run the validation utility against HL7 data:

1. Open the main map source, **hl7main**, in the Design Server.
2. Update Input card 1 to select the HL7 input data to validate. Sample file **hl7\_edi\_example\_V2.7** in the hl7\data directory is used in this card as installed. The input can be **EDI HL7** or **XML HL7**. The correct processing of the data is determined by the map by examining the data input.
3. To modify or review any configurable parameters, edit the file: **hl7parms.xml**.
4. Compile all maps in maps directory:
  - In Design Studio: Use **compilemaps.bat**.
  - In Design Server: Create and build a package with all maps.
5. Execute **hl7main** to run the HL7 Validation utility. The following output files are produced in the hl7\data directory:
  - **StatusReport.html** - Report that summarizes the final status for the validation, Message Type, Message Version, XML Scan Status (if the input is XML HL7 data), XML Validation Status, and XML > EDI or EDI > XML Conversion Status.
  - **HL7Out.xml** - If the input file was EDI HL7, this is the UTF-8 converted XML data.
  - **EDIOut.edi** - If the input file was EDI XML, this is the EDI converted data.
  - **hl7\_edi\_ack.out** - If the input was HL7 EDI, and acknowledgment was requested, an Original Mode Acknowledgment is generated in HL7 EDI format.
  - **hl7\_xml\_ack.out** - If the input is HL7 XML and acknowledgment was requested, an Original Mode Acknowledgment is generated in XML format.

---

## HL7 component examples

Examples are provided to demonstrate how to use the HL7 component.

- [NCPDP-SCRIPT / HL7 mapping example](#)

The example for NCPDP-SCRIPT/ HL7 mapping is based on the *NCPDP-HL7 Electronic Prescribing Coordination Mapping Document*. That document provides mapping requirements for interactions between systems using HL7 and systems using NCPDP Script.

- [FHIR example](#)

The FHIR examples demonstrate the mapping from HL7 Fast Healthcare Interoperability Resources to HL7 V2 format and mapping from a Da Vinci Prior Authorization Support (PAS) bundle to the HIPAA EDI Prior Authorization Request (278) and Clinical Attachment (275) transactions.

- [FHIR Example - PAS Bundle with JSON Schema](#)

The FHIR Mapping example demonstrates mapping between FHIR PAS JSON bundles for mapping between Prior Authorization Request and Responses (278) and a FHIR Da Vinci Prior Authorization Support (PAS) bundle. These crosswalk mappings are expected to be used in the industry to support several US interoperability initiatives and conform to the Implementation Guides published by X12N Insurance.

- [FHIR Validation Example](#)

- [HL7 MLLP Example](#)

The MLLP adapter provides a means of passing data to and from the Transformation Extender using the MLLP protocol over TCP/IP.

---

## NCPDP-SCRIPT / HL7 mapping example

The example for NCPDP-SCRIPT/ HL7 mapping is based on the *NCPDP-HL7 Electronic Prescribing Coordination Mapping Document*. That document provides mapping requirements for interactions between systems using HL7 and systems using NCPDP Script.

The NCPDP-SCRIPT/ HL7 mapping example shows both the HL7 to NCPDP and NCPDP to HL7 directions for a new **NEWRX** prescription message. You can use this example as a starting point for the actual maps needed for this transformation or as a general model to understand such mapping.

- [Location of HL7/NCPDP-SCRIPT mapping files](#)

This documentation identifies the directory location of the HL7/NCPDP-SCRIPT mapping example files.

- [HL7 / NCPDP-SCRIPT mapping schemas](#)

This documentation describes the HL7/NCPDP mapping schemas.

- [NCPDP-SCRIPT / HL7 example maps](#)

This documentation describes the example maps provided in the NCPDP-SCRIPT / HL7 example.

---

## Location of HL7/NCPDP-SCRIPT mapping files

This documentation identifies the directory location of the HL7/NCPDP-SCRIPT mapping example files.

The HL7/NCPDP mapping example is located in the following folder:

<install\_dir>\packs\healthcare\_vn.n.n.n\hl7\examples\hl7\_ncpdp\_script

Where <install\_dir> indicates the installed location of the Transformation Extender Platform base product, and n.n.n.n is the version number of the pack.

## HL7 / NCPDP-SCRIPT mapping schemas

This documentation describes the HL7/NCPDP mapping schemas.

The schemas and JSON metadata files install within the imported project:

- hl7\_v2\_8\_2 – a copy of the hl7 schema for version 2.8.2 of the standard.
- ncpdp\_script\_v10\_6\_201610 – NCPDP schema for script version 10.6.

These schemas are copied to the example location. Other versions of HL7 or NCPDP can be used. All versions provided in the pack are supported in the *NCPDP-HL7 Electronic Prescribing Coordination Mapping Document*.

## NCPDP-SCRIPT / HL7 example maps

This documentation describes the example maps provided in the NCPDP-SCRIPT / HL7 example.

There are two maps provided in the example, one for each direction of transformation:

- ncpdp\_script\_to\_hl7 – maps a NCPDP NEWRX message to an HL7 OMP message, using ncpdp\_input.dat, producing hl7\_output.dat
- hl7\_to\_ncpdp\_script – maps HL7 OMP to NCPDP NEWRX, using hl7\_output.dat as input.

## FHIR example

The FHIR examples demonstrate the mapping from HL7 Fast Healthcare Interoperability Resources to HL7 V2 format and mapping from a Da Vinci Prior Authorization Support (PAS) bundle to the HIPAA EDI Prior Authorization Request (278) and Clinical Attachment (275) transactions.

- [Location of the FHIR example files](#)

This documentation identifies the directory location of the FHIR example files.

- [FHIR example schemas](#)

The schemas and JSON metadata files are included for this example.

- [FHIR example maps](#)

The FHIR example provides maps to enable transformation between HL7 ORU messages in EDI format and a FHIR Observation resource in JSON or XML format.

## Location of the FHIR example files

This documentation identifies the directory location of the FHIR example files.

<install\_dir>\packs\healthcare\_vn.n.n.n\hl7\examples\fhir

Where <install\_dir> indicates the installed location of the Transformation Extender base product, and n.n.n.n is the version number of the pack.

When running the example on Design Server, the example files are installed within the imported hl7Examples.zip project.

## FHIR example schemas

The schemas and JSON metadata files are included for this example.

- hl7\_v2\_8\_2 - A copy of the HL7 schema for version 2.8.2 of the standard
- hipaa\_x12\_ruleless - A copy of the HIPAA schema used for mapping 5010 278 and 275 output
- Redistributed HL7 FHIR XML schemas, FHIR v4.0.0
- JSON template from a HL7 FHIR Observation example

## FHIR example maps

The FHIR example provides maps to enable transformation between HL7 ORU messages in EDI format and a FHIR Observation resource in JSON or XML format.

The observation\_json map performs transformations based upon whether the input is in EDI or JSON format, and produces the corresponding output. Similarly, the observation\_xml map performs transformations based upon whether the input is in EDI, or XML format.

There is an additional input file, codsystem\_lookup.txt, that is used to transform the URLs used for code systems in FHIR, to the CODE name that would be used in HL7 EDI. This lookup file is provided as an example on how transformation can be enriched with additional resources if needed.

It also provides the pas\_fhir\_to\_x12 map to create HIPAA EDI Prior Authorization Request (278) and Clinical Attachment (275) transactions from a FHIR Da Vinci Prior Authorization Support (PAS) bundle. These crosswalk mappings are expected to be used in the industry to support several US interoperability initiatives and conform to the Implementation Guides published by X12N Insurance.

- [Running the example for FHIR JSON format](#)

These steps describe how to run the example for the FHIR JSON format:

- [Running the example in FHIR XML format](#)

These steps describe how to run the example for FHIR XML format:

- [Running the FHIR PAS bundle example map](#)

These steps describe how to run the example for the FHIR PAS bundle:

---

## Running the example for FHIR JSON format

These steps describe how to run the example for the FHIR JSON format:

To convert an Observation Resource to an HL7 ORU message use observation\_example.json for the Input\_Data input in the observation\_json map.

1. Build all maps.
2. Run observation\_json.

HL7\_EDI\_Output is created, producing the file: hl7\_edi\_oru\_output.out

To convert an HL7 ORU message to an Observation resource, use the sample\_hl7\_edi.oru.dat file in the observation\_json map.

1. Build all maps
2. Run observation\_json.

FHIR\_JSON\_Output is created, producing file fhir\_json\_observation\_output.json.

---

## Running the example in FHIR XML formal

These steps describe how to run the example for FHIR XML format:

To convert an Observation Resource to an HL7 ORU message, use observation\_example.xml for the Input\_Data input in the observation\_xml map.

1. Build all maps.
2. Run observation\_xml.

HL7\_EDI\_Output is created, producing the file hl7\_edi\_oru\_output.out.

To convert an HL7 ORU message to an Observation resource, use sample\_hl7\_edi.oru.dat in the observation\_xml map.

1. Build all maps.
2. Run observation\_xml.

FHIR\_XML\_Output is created, producing the file fhir\_xml\_observaton\_output.xml.

---

## Running the FHIR PAS bundle example map

These steps describe how to run the example for the FHIR PAS bundle:

To create an X12 interchange with the 278 Health Care Services Review - Request for Review and Response (005010X217) transaction and corresponding 275 Additional Information to Support a Health Care Services Review (006020X316) transaction from a PAS bundle representing a medical services authorization, use bundle.xml input in pas\_fhir\_to\_x12 map (Source map fhir\_bundle\_to\_x12 if running in Design Studio).

1. Build all maps.
2. Run pas\_fhir\_to\_x12.

The X12\_interchange output is created, producing the file pas\_278\_275\_request.out.

---

## FHIR Example - PAS Bundle with JSON Schema

The FHIR Mapping example demonstrates mapping between FHIR PAS JSON bundles for mapping between Prior Authorization Request and Responses (278) and a FHIR Da Vinci Prior Authorization Support (PAS) bundle. These crosswalk mappings are expected to be used in the industry to support several US interoperability initiatives and conform to the Implementation Guides published by X12N Insurance.

- [Location of the PAS Bundle with JSON example files](#)

---

## Location of the PAS Bundle with JSON example files

For design studio installs, the fhir\_val.mms map is found at:

```
<install_dir>\packs\healthcare_vn.n.n.n\hl7\examples\fhirPASJson
```

Where *<install\_dir>* indicates the installed location of the Transformation Extender base product, and *n.n.n.n* is the version number of the pack.

The PAS\_Request map uses as input a Claim Request Sample Bundle in JSON format and generates the corresponding 5020X217 278 Request in X12 HIPAA EDI format.

The PAS\_Response maps take an 5010X217 278 Response and generates the corresponding ClaimResponse Bundle in JSON format.

Both maps use a modified JSON schema for validation and mapping - unused FHIR resources are removed for clearer mapping.

The mapping is based on the guidelines documented in the X12 Implementation guide for 5010X217.

---

## FHIR Validation Example

The FHIR validation example shows how a simple map or flow can invoke HL7 FHIR validation using the FHIR test server at <https://hapi.fhir.org/baseR5>.

- [Location of the FHIR Validation example files](#)

---

## Location of the FHIR Validation example files

For design studio installs, the fhir\_val.mms map is found at:

```
<install_dir>\packs\healthcare_vn.n.n.n\hl7\examples\fhirValidation
```

Where *<install\_dir>* indicates the installed location of the Transformation Extender base product, and *n.n.n.n* is the version number of the pack

When this map is ran standalone, it invokes the FHIR test server to validate the input observation resource. Two outputs are produced if the test server is accessible, and the input can be validated as an FHIR observation resource. One output (Card 2) is validation\_results.json, which is the parsed Outcome resource returned by the test server. The other output (Card3) is simply the original input passed through, with a flow variable set. However, when running this map standalone, the flow variable is extraneous – it is only in the map for its use within the flow.

The sample flow is only supported on the Design Server and is available as a project hl7fhirValidationExample.

This flow invokes HL7 Validation for a single JSON formatted FHIR Observation resource and returns Validation results as an Operation Outcome resource on one of the flow endpoints. If the validation outcome contains no warnings or errors, the map will set a flow variable *valid* to true. If this variable is set to true, a decision node in the flow will then route the original resource to be converted to XML. Select file observation\_valid to run the flow for valid data and see if the data has been transformed to XML. Select file observation\_invalid to run flow for invalid data.

If the test server is not accessible, the map, fhir\_val, that invokes the FHIR test server to validate the input observation resource will fail. The remaining nodes in the flow will not execute, and no data is returned in the endpoints. View the map log to see any details on the failure.

---

## HL7 MLLP Example

The MLLP adapter provides a means of passing data to and from the Transformation Extender using the MLLP protocol over TCP/IP.

This example demonstrates the use of the MLLP adapter in Transformation Extender maps.

A map running under the Transformation Extender Launcher System, or in a Flow is used to simulate an application that accepts an HL7 message via MLLP, and returns an acknowledgment.

- [Location of the MLLP example files](#)

The source maps and launcher system to be used with the Transformation Extender Flow Server or Launcher runtime is found in:

---

## Location of the MLLP example files

The source maps and launcher system to be used with the Transformation Extender Flow Server or Launcher runtime is found in:

```
<install_dir>\packs\healthcare_vn.n.n.n\hl7\examples\mllp
```

Where *<install\_dir>* indicates the installed location of the Transformation Extender base product, and *n.n.n.n* is the version number of the pack.

This also contains a readme.txt file with more details on how to use the example in either the Launcher or Flow Server.

The sample flow and contained nodes and maps, is supplied for the Design Server with Flow engine, and is available as a project hl7mllpExample.

---

## NCPDP component

The Pack for Healthcare NCPDP component is a solution for National Council of Prescription Drug Program (NCPDP) healthcare-industry participants. The NCPDP transactions are used exclusively in the retail pharmacy sector.

In addition to the NCPDP component, there are two other Pack for Healthcare components:

- The HIPAA EDI component
  - The HL7 component
  - [Objects included in the pack](#)
  - [NCPDP maps](#)
  - [Sample Data](#)

Sample data is provided for use with the schemas and maps in the NCPDP component.
  - [NCPDP component examples](#)

This documentation describes the NCPDP SCRIPT example and the PACDR example.
  - [NCPDP SCRIPT schemas](#)

The Transformation Extender interface shows the NCPDP SCRIPT and XML data structure as both an expandable and contractible set of hierarchical components.
  - [NCPDP validation](#)

The NCPDP component provides validation of the NCPDP Telecom version D0 Request Transactions with Batch 1.2 data standards, with regular updates to support code list changes.
- 

## Objects included in the pack

The data exchange, transformation, and integration requirements of the healthcare industry range from simple to complex and can vary greatly from one organization to another. NCPDP component contains predefined objects. These objects are flexible and allow a wide variety of applications and systems to be developed that meet your specific requirements.

The pack contains schemas, maps, sample data, and example utility modules. These predefined, industry-specific objects are organized in a way that provides the flexibility to create and deploy a wide variety of applications and systems that address NCPDP integration requirements. These objects are constructed to allow consistent behavior and results across all supported platforms and operating systems.

The schemas define the more commonly used healthcare industry data standards. The maps, data, and utility modules embrace typical data validation and transformation scenarios.

- [NCPDP schemas](#)
  - [Sample data](#)
  - [Maps](#)
- 

## NCPDP schemas

The NCPDP schemas describe the retail pharmacy telecommunications and batch data interchange formats as specified by the National Council for Prescription Drug Programs (NCPDP).

The NCPDP schemas are named based on the annual implementation version and emergency code list publications.

## Sample data

The data files included in the NCPDP component contain transaction data exchange formats associated with the National Council for Prescription Drug Program (NCPDP) health care application standard.

## Maps

The NCPDP reporting and transformation maps are included in the NCPDP component.

## NCPDP maps

The NCPDP component contains two types of maps. The NCPDP reporting maps are pass-through maps. These maps are used for data validation for NCPDP Versions 5.1, D.0, and 3.0. The NCPDP SCRIPT maps are transformation and pass-through maps for NCPDP SCRIPT Version 10.6.

## Sample Data

Sample data is provided for use with the schemas and maps in the NCPDP component.

- [Transaction data](#)

The NCPDP component includes sample transaction data for NCPDP
- 

## Transaction data

The NCPDP component includes sample transaction data for NCPDP

- [NCPDP sample data](#)

## NCPDP sample data

NCPDP sample data includes request and response data for four of the retail pharmacy transactions as specified by the National Council for Prescription Drug Programs.

NCPDP sample data files are located in the following directory:

*install\_dir\packs\healthcare\_vn.n.n\ncpdp\data*

In the above path, *install\_dir* is the location of the Transformation Extender base product, and *n.n.n* is the version number of the Pack for Healthcare.

The NCPDP sample data files include the following:

- ncpdp\_medsub\_v3\_0\_billrev\_req\_sample.dat
- ncpdp\_medsub\_v3\_0\_billrev\_res\_acc\_sample.dat
- ncpdp\_medsub\_v3\_0\_bill\_req\_sample.dat
- ncpdp\_medsub\_v3\_0\_bill\_res\_rej\_sample.dat
- ncpdp\_medsub\_v3\_0\_rebill\_req\_sample.dat
- ncpdp\_medsub\_v3\_0\_rebill\_res\_acc\_sample.dat
- ncpdp\_vd\_0\_rebill\_response.dat
- ncpdp\_vd\_0\_eligibility\_response.dat
- ncpdp\_vd\_0\_billing\_reversal\_response\_accept.dat
- ncpdp\_vd\_0\_billing\_response\_reject.dat
- ncpdp\_vd\_0\_rebill\_request.dat
- ncpdp\_vd\_0\_eligibility\_request.dat
- ncpdp\_vd\_0\_billing\_reversal\_request.dat
- ncpdp\_vd\_0\_billing\_request.dat

NCPDP example data files are located in the following directory:

*install\_dir\packs\healthcare\_vn.n.n\ncpdp\examples\SCRIPT\data*

These files include the following:

- ncpdp\_script\_v10\_6\_canres.dat
- ncpdp\_script\_v10\_6\_canrx.dat
- ncpdp\_script\_v10\_6\_census.dat
- ncpdp\_script\_v10\_6\_chgres.dat
- ncpdp\_script\_v10\_6\_error.dat
- ncpdp\_script\_v10\_6\_getmsg.dat
- ncpdp\_script\_v10\_6\_newrx.dat
- ncpdp\_script\_v10\_6\_paschg.dat
- ncpdp\_script\_v10\_6\_refreq.dat
- ncpdp\_script\_v10\_6\_refres.dat
- ncpdp\_script\_v10\_6\_resupp.dat
- ncpdp\_script\_v10\_6\_rxhres.dat
- ncpdp\_script\_v10\_6\_status.dat
- ncpdp\_script\_v10\_6\_verify.dat
- ncpdp\_script\_v10\_6\_rxchg.dat
- ncpdp\_script\_v10\_6\_rxfill.dat
- ncpdp\_script\_v10\_6\_rxhreq.dat

## NCPDP component examples

This documentation describes the NCPDP SCRIPT example and the PACDR example.

- [NCPDP SCRIPT example](#)

The mappings supplied with the NCPDP component show the transformation capabilities of the product, and provide a starting point for transformations.

- [NCPDP SCRIPT](#)

- [NCPDP PACDR example](#)

An example is included in the NCPDP component that validates the NCPDP Post Adjudication Standard Version 4.2.

## NCPDP SCRIPT example

The mappings supplied with the NCPDP component show the transformation capabilities of the product, and provide a starting point for transformations.

The example files install within the imported project.

- [Location of the Script example files](#)

The script transformation maps transform the SCRIPT input data to and from SCRIPT xml. The script pass-through maps validate the SCRIPT and SCRIPT xml input

data. The map source files are located at:

- [Release indicator](#)
- [XML validation](#)
- [Using SCRIPT examples on UNIX platforms](#)
- [Using SCRIPT example files in the z/OS environment](#)

## Location of the Script example files

The script transformation maps transform the SCRIPT input data to and from SCRIPT xml. The script pass-through maps validate the SCRIPT and SCRIPT xml input data. The map source files are located at:

install\_dir\packs\healthcare\_vn.n.n\ncpdp\examples\SCRIPT\mapsandschemas

Where <install\_dir> is the location of the Transformation Extender base product, and n.n.n is the version number of the pack.

If schema validation is desired, you must obtain the schema components required to run these maps directly from NCPDP from:

<http://www.ncpdp.org>

## Release indicator

The SCRIPT UNA Service String Advice contains the delimiters to be used for parsing. These values must be non-printable characters. The SCRIPT default Release Indicator Value is 32 (x'20). This value identifies the Release Indicator as not used.

For inbound processing, the value from the UNA is used if the value is not the default (x'20). If the UNA contains the value (x'20), the default value identified in the SCRIPT schema is used. The schema ncpdp\_script\_v10\_6 contains the default value of <ESC>(x'lb').

For outbound processing, the UNA is created using mapping commands and literal values. The Release Indicator value currently mapped is the default (x'20'). The example map, ncpdp\_xml\_to\_script\_v10\_6.mms, can be modified to identify the Release Indicator. The mapping commands are located in the functional map, UNA\_Service\_String.

The schema, ncpdp\_script\_v10\_6, contains a default value of <ESC>(x'lb'). This default can be modified by changing the schema property, **Type Syntax**, for all schema components that identify the Release Identifier.

## XML validation

Schema components required to run these maps must be obtained directly from NCPDP at:[www.ncpdp.org](http://www.ncpdp.org).

The following errors are related to XML element names that contain a space and were reported during XML validation:

	<DetailedStatusMessage> Error(-1), "XMLParser: Input XML data is invalid." SAXParseException, Error [line: 3264 column: 84] Invalid element name:'DosingBasisUnitofMeasureCode' - declaration ignored
	Error (-1), "XMLParser: Input XML data is invalid." SAXParseException, Error [line: 3294 column: 105] Invalid element name:'CalculatedDoseUnitofMeasureCodeQualifier' - declaration ignored Error (-1), "XMLParser: Input XML data is invalid." SAXParseException, Error [line: 3377 column: 81] Invalid element name:'RouteofAdministrationCode' - declaration ignored
	Error (-1), "XMLParser: Input XML data is invalid." SAXParseException, Error [line: 3465 column: 75] Invalid element name:'TimePeriodBasisText' - declaration ignored
	Error (-1), "XMLParser: Input XML data is invalid." SAXParseException, Error [line: 3598 column: 113] Invalid element name:'MaximumDoseRestrictionVariableUnitsCodeQualifier' - declaration ignored
	Error (-1), "XMLParser: Input XML data is invalid." SAXParseException, Error [line: 3667 column: 106] Invalid element name:'IndicationValueUnitofMeasureCodeQualifier' - declaration ignored
	Error (-1), "XMLParser: Input XML data is invalid."
	SAXParseException, Error [line: 2 column: 38] Unknown element 'Message'
	</DetailedStatusMessage>

## Using SCRIPT examples on UNIX platforms

On UNIX platforms, be aware of file pths and case sensitivity for schemas and DTDs. Modifications might be needed to ensure naming conventions match the case.

## Using SCRIPT example files in the z/OS environment

Be aware of the following when using the NCPDP SCRIPT example files in the z/OS environment:

- When ported to z/OS, every map, schema, and file that is ported to z/OS must have a unique 8-character DDNAME in the JCL.
- After building maps for a z/OS platform, you must rename the resulting \*.mvs compiled map to a unique 8 character name. The compiled map file, which contains maps and schemas, should be uploaded as a PDS member. Both maps and schemas must be uploaded as binary files.
- Schemas and DTDs should be uploaded to a variable blocked dataset. These files can reside with the maps in a separate PDS or sequential dataset. You must, however, be aware of the ramifications of uploading variable-length data into fixed blocked datasets. The records are padded up to the fixed size with NULL (x'00') characters. Null characters are not valid XML.

## NCPDP SCRIPT

The script transformation maps transform the SCRIPT input data to and from SCRIPT xml. The script pass-through maps validate the SCRIPT and SCRIPT xml input data. The map source files are located in the following directory:

*install\_dir\packs\healthcare\_vn.n.n\ncpdp\examples\SCRIPT\mapsandschemas*

In the above path, *install\_dir* is the location of the Transformation Extender base product, and *n.n.n* is the version number of the installed Healthcare Pack.

If schema validation is desired, you must obtain the schema components required to run these maps directly from NCPDP from:

[www.ncpdp.org](http://www.ncpdp.org)

This section describes the following maps:

- [ncpdp\\_script\\_v10\\_6\\_to\\_xml.mms](#)
- [ncpdp\\_xml\\_to\\_script\\_v10\\_6.mms](#)
- [ncpdp\\_script\\_v10\\_6\\_passthru.mms](#)
- [ncpdp\\_xml\\_v10\\_6\\_passthru.mms](#)
- [ncpdp\\_script\\_v10\\_6\\_xml\\_v201707.mms](#)
- [\*\*ncpdp\\_script\\_v10\\_6\\_to\\_xml.mms\*\*](#)
- [\*\*ncpdp\\_xml\\_to\\_script\\_v10\\_6.mms\*\*](#)
- [\*\*ncpdp\\_script\\_v10\\_6\\_passthru.mms\*\*](#)
- [\*\*ncpdp\\_xml\\_v10\\_6\\_passthru.mms\*\*](#)
- [\*\*ncpdp\\_script\\_v10\\_6\\_xml\\_v201707.mms\*\*](#)

### ncpdp\_script\_v10\_6\_to\_xml.mms

The ncpdp\_script\_v10\_6\_to\_xml.mms map source file transforms script format to xml format. The input to the map is a file that contains the NCPDP Script v10.6 Message (or Messages). The map produces script xml V10.6 output. The ncpdp\_script\_v10\_6\_transform is the top-level map. This map performs the following functions:

- Validates NCPDP request or response transmission data against definitions in the ncpdp\_script\_v10\_6 schema.
- Generates a script xml file from the transmission.

### ncpdp\_xml\_to\_script\_v10\_6.mms

The ncpdp\_xml\_to\_script\_v10\_6.mms map source file transforms xml format to script format. The input to the map is a file containing the NCPDP Script xml v10.6 Message (or Messages). The map produces script V10.6 output. The ncpdp\_xml\_v10\_6\_transform is the top-level map. Obtain the schema components required to run this map directly from NCPDP at the following location:

[www.ncpdp.org](http://www.ncpdp.org)

This map performs the following functions:

- Parses NCPDP xml request and response xml transmission data using definitions in the script\_xml\_v10\_6 schema.
- Validates the xml request, or response transmission as well formed. If schema validation is desired, the document verification setting for the input card must be set to **Always** or **Never**. The script\_xml\_10\_6\_1.xsd schema is required for schema validation and must be obtained directly from NCPDP at the website noted above. If schema validation is not required, the document verification setting for the input card must be set to well-formed.
- Generates a script file from the transmission.

### ncpdp\_script\_v10\_6\_passthru.mms

The ncpdp\_script\_v10\_6\_passthru.mms map source file validates ncpdp script transmission data against the Partner Inbound Transmission NCPDP\_SCRIPT and Partner Outbound Transmission NCPDP\_SCRIPT definitions in the ncpdp\_script\_v10\_6 schema.

The ncpdp\_v10\_6\_script\_passthru\_file is the top-level map.

## ncpdp\_xml\_v10\_6\_passthru.mms

The ncpdp\_xml\_v10\_6\_passthru.mms map source file parses ncpdp script xml transmission using the Doc XSD definitions in the script\_xml\_v10\_6 schema and validates ncpdp script xml transmission data as well formed. If schema validation is desired, the document verification setting for the input card must be set to **Always** or **Never**. The script\_xml\_10\_6\_1.xsd schema (not included) is required for schema validation and must be obtained directly from NCPDP at this website:

[www.ncpdp.org](http://www.ncpdp.org)

If schema validation is not required, the document verification setting for the input card must be set to well formed.

The ncpdp\_v10\_6\_xml\_passthru\_file is the top-level map.

## ncpdp\_script\_v10\_6\_xml\_v201707.mms

The ncpdp\_script\_v10\_6\_to\_xml\_v201707.mms map source file transforms script format to xml format. The input to the map is a file that contains the NCPDP Script v10.6 Message (or Messages). The map produces script xml v201707 output from the NCPDP provided v201707 schemas.

The v201707 schema components required to run these maps must be obtained directly from NCPDP at:

<http://www.ncpdp.org>

## NCPDP PACDR example

An example is included in the NCPDP component that validates the NCPDP Post Adjudication Standard Version 4.2.

The example files install within the imported project.

The following components are installed.

## NCPDP SCRIPT schemas

The Transformation Extender interface shows the NCPDP SCRIPT and XML data structure as both an expandable and contractible set of hierarchical components.

- [SCRIPT schema structure](#)
- [Data element type names](#)
- [Structure of XML schemas](#)  
The structure of the XML schemas in the NCPDP component is similar to the structure of the originating XML schema.
- [SCRIPT validation](#)

## SCRIPT schema structure

The structure of the SCRIPT schema is similar to an EDI schema. It includes definitions of objects used in EDI. The root type of the SCRIPT schema is NCPDP\_SCRIPT. The schema has the following categories:

- %\_Type\_Tree\_Information
- Interchange
- SCRIPT\_V10\_6
- Transmission

These categories stem from the NCPDP\_SCRIPT root type. Some categories contain Inbound and Outbound group subtypes. These are identical and have been supplied as a convenience to allow customization that is dependent on the direction of the messages.

The largest object in an EDI schema is a transmission. A transmission can include many interchanges from or to many trading partners.

The following are subtypes of the SCRIPT\_V10\_6 category:

- Control type defines the NCPDP Script control structure objects such as Envelope Segments and Delimiters.
- Msg type defines the messages for the version.
- Vxx category defines the message contents, composites, and elements in this particular version. The segment definitions for each message are defined as subtypes of that message.

## Data element type names

The type names of some data elements are abbreviated versions of the element descriptions. For example, the name of the type that defines the data element is Character encoding, which is coded as **CharEncodingCd**, and Agency Qualifier, which is coded as **AgencyQual'r**. If these, or other names, are not appropriate for your use, rename the type. Every reference to that type is then renamed automatically. You must, however, modify any mapping commands that reference these components.

## Structure of XML schemas

The structure of the XML schemas in the NCPDP component is similar to the structure of the originating XML schema.

The root of the NCPDP XML schema is the Doc XSD group. Most executable mapping will be to or from the Doc component. Under Global, the schema has the Message Element along with the other Global options defined in the schema. The Message Element contains all components to be used for mapping.

## SCRIPT validation

The **ncpdp\_script\_v10\_6** schema describes the NCPDP SCRIPT messages and enveloping structures. SCRIPT V10.6 is backward compatible with SCRIPT V8.1. This schema contains component rules and restriction lists to enforce standard syntax and data integrity, as in the following:

- valid segments
- segment order
- maximum segment and loop occurrences
- element attributes, for example, size and data
- required, or intra-segment situational data elements (element cardinality within segments)

The **ncpdp\_script\_v10\_6** schema does not enforce the following:

- inter-segment situations, for example, If A occurs, then B must be populated
- restriction lists not documented in the SCRIPT implementation guide, or external code list and data dictionary
- conflicting syntax, restriction lists, or rules based on V10.6 and V8.1 implementation guides
- balancing rules, for example, element A equals B plus C.

## NCPDP validation

The NCPDP component provides validation of the NCPDP Telecom version D0 Request Transactions with Batch 1.2 data standards, with regular updates to support code list changes.

The NCPDP validation utility produces a summary output of invalid fields or segments and a partially populated NCPDP response. This initial validation utility reports field-level errors within a segment based on the transaction type and segment-level errors for a transaction. This includes checking for missing required fields/segments, not used fields/segments that are present, and conditionally required values based on the transaction type or other fields in the same segment. It also reports errors for incorrect code values.

- **Running the NCPDP validation utility**

The NCPDP validation utility uses the existing NCPDP Telecom schemas in the pack for initial validation, with the restriction list checking disabled. The base **ncpdp\_vd\_0.mtt** and **ncpdp\_vf6.mtt** (schema if using Design Server) are used for this.

## Running the NCPDP validation utility

The NCPDP validation utility uses the existing NCPDP Telecom schemas in the pack for initial validation, with the restriction list checking disabled. The base **ncpdp\_vd\_0.mtt** and **ncpdp\_vf6.mtt** (schema if using Design Server) are used for this.

## About this task

Code values are checked against the input file **element\_restriction\_list.json**, using the **codeListDate** in the **ncpdp\_parameter.json** file to determine the start date for code list values that may have been added in recent standard updates. This file has values for both F6 and D0 versions, using a new field version at all levels to distinguish any segments, fields, or values that are only used in F6 or D0.

The situational and field format rules are defined in **segment\_rules.json** and **segment\_rules\_f6.json**. In this case both files are needed to support the two different versions, but the initial processing in the validation map will determine the version of the NCPDP input file, and select the correct segment rules input appropriately.

Failures from component rules or cardinality in the schema will be reported by error checking in the NCPDP validation utility through map rules and/or the Java exit that is invoked.

Follow these steps to run the NCPDP validation utility:

## Procedure

1. Update **ncpdp\_parameter.json** with the code list date.  
For example, to enforce the code values from **External Code** lists that correspond to the set published in 2017 for activation in 2018, use the following:
  - a. "codeListDate":"201710"
2. Update **ncpdp\_parameter.json** with the Medicaid option as desired. If validation per the Medicaid Subrogation rules is needed, use true. The default is false.
  - a. "medicaidSub":true
3. Ensure **ncpdp\_jexit.jar** is available in the Java Classpath during validation utility execution by one of the following methods:

- a. Copy from <pack\_install\_dir>\ncpdp\jars to <tx\_install\_dir> directory on Windows or <install\_dir>/libs directory on Unix.
  - b. Add a jar reference in the [External Jar Files] section of the <tx\_install\_dir> dtx.ini file to the location of the jar file in <pack\_install\_dir>\ncpdp\jars.
  - c. If running on the design server, edit the <pack\_install\_dir>\ncpdp\jars\ncpdpsetup.bat or ncpdpsetup.sh to point to your design server install, and execute the batch or shell script.
4. Open the ncpdp\_validation map.
  5. If validating data at the message level, perform the following steps:
    - a. Update Input card 1 of **ncpdp\_message\_validation** to select the NCPDP input data for validation.
    - b. Build all maps.
    - c. Execute ncpdp\_message\_validation.
    - d. Output files produced:
      - data/message\_response.dat contains the Response message or the NCPDP proscribed format.
      - data/message\_validation\_report.xml contains XML formatted report
  6. If validating data at the batch level, perform the following steps:
    - a. Update input card1 of the ncpdp\_batch\_validation map to select NCPDP data for validation.
    - b. Build all maps.
    - c. Execute ncpdp\_batch\_validation.
    - d. Output files produced:
      - data/batch\_response.dat contains the Response message or the NCPDP proscribed format.
      - data/batch\_validation\_report.xml contains XML formatted report

## Pack for Supply Chain EDI overview

The IBM Transformation Extender Pack for Supply Chain EDI is an industry accelerator that provides Business-to-Business enabling functionality to users of the Transformation Extender.

The Pack for Supply Chain EDI supports these standards.

- X12, RAIL, and TT-X12 (Travel and Transportation X12 subset)
- EDIFACT and TT-EDIFACT (Travel and Transportation EDIFACT subset)
- IATA (The International Air Transport Association) API and PNR
- TRADACOMS

The pack provides the following functionality.

- Ability to perform B2B enablement functionality for X12, RAIL, EDIFACT, IATA, and TRADACOMS data:
  - Enable data transformation by use of the IBM® B2Bi legacy mapper.
  - Enable data transformation by use of definitions in Transformation Extender schema format.
- Ability to perform X12, and EDIFACT, compliance validation, reporting, and data valid/invalid splitting.
- [About Electronic Data Interchange \(EDI\)](#)  
Electronic Data Interchange (EDI) is the electronic exchange of routine business information by using an agreed-upon file structure.
- [Example files](#)

## About Electronic Data Interchange (EDI)

Electronic Data Interchange (EDI) is the electronic exchange of routine business information by using an agreed-upon file structure.

The file structures are defined with EDI standards for the electronic documents. These uniform electronic formats are referred to as 'standards'. Standards, which are periodically updated, are referenced with versions that specify releases. For example, X12 Version 4010 is version 004, release 010; Version 3050 is version 003, release 050.

EDIFACT (Electronic Data Interchange For Administration, Commerce, and Transport) transitioned to using the **D** versions starting in the early 1990s. The **D** versioning system signifies that the release is a directory, updated twice a year, in a specific year.

- D.XXA  
Indicates the first release of the year **XX**.  
For example, **D.21A** refers to the first release of the year 2021.
- D.XXB  
Indicates the second release of the year **XX**.  
For example, **D.21B** refers to the second release of the year 2021.

## Example files

Example files are included with each pack component for both compliance checking and data transformation.

### Windows Design Studio Install:

Both EDIFACT and X12 example files are installed when you install the Pack for Supply Chain EDI.

*install\_dir\supplychain\_edi\_vn.n.n\x12\compliance\data*

*install\_dir\supplychain\_edi\_vn.n.n\x12\examples*

*install\_dir\supplychain\_edi\_vn.n.n\edifact\compliance\data*

*install\_dir\supplychain\_edi\_vn.n.n\edifact\examples*

#### **Design Server Install:**

Pack components are provided in the form of projects to be imported into the Design Server. These include Standards for custom mapping, compliance checking, and examples. Standards and compliance checking projects are provided for all pack components. X12 and EDIFACT provide additional examples with data transformation mappings.

---

## Installation and uninstallation

Installation and uninstallation procedures are described in the release notes on the Support website.

<http://www.ibm.com/support/docview.wss?uid=swg27008337>

---

## What is an EDI schema?

An EDI schema includes definitions of objects that are used in EDI.

Using these schemas, you can map directly from a data source or to a data target containing EDI data. You do not need to learn yet another format to map to and from EDI, as you do with many other translator products.

The largest object in an EDI schema is a transmission. A transmission can include many interchanges from many trading partners, or to many trading partners.

- **[ANSI X12 data](#)**

The basic business document in ANSI X12 data is called a transaction set. Transaction sets are enclosed in an envelope that separates one transaction set (ST-SE envelope) from another .

- **[EDIFACT data](#)**

EDIFACT data is similar to ANSI X12 data. The difference is that the business document in EDIFACT data is called a message and a loop in EDIFACT data is called a group.

- **[TRADACOMS data](#)**

The TRADACOM standard is primarily used in England and trading partners throughout Europe. TRADACOMS messages are used for domestic trade within the UK and cover a range of commercial transactions plus reports and master files.

- **[EDI version release schemas](#)**

The data for each EDI version is defined in a separate schema. For example, the ansi3070 schema defines ANSI X12 data version 3070.

- **[Types in EDI schemas](#)**

The type names in EDI schemas are consistent with the terminology used in the EDI standards.

- **[Modifying EDI schemas](#)**

Make your own copy of each EDI schema you are likely to use.

---

## ANSI X12 data

The basic business document in ANSI X12 data is called a transaction set. Transaction sets are enclosed in an envelope that separates one transaction set (ST-SE envelope) from another .

Groups of transaction sets that are functionally related are enclosed in a functional group envelope (GS-GE envelope). The functional group envelope separates one functional group from another. Functional groups that come from the same trading partner are grouped in an interchange envelope (ISA-IEA envelope). A series of interchanges from a variety of trading partners forms a transmission.

- **[Envelopes](#)**

Each envelope in EDI data begins with a particular segment and ends with a particular segment. For example, the transaction set envelope begins with an ST segment and ends with an SE segment.

- **[Transaction sets](#)**

Transaction sets are made up of segments and loops. A segment contains a unit of information, for example, a line item or a patient record.

- **[Segments](#)**

Segments are made up of data elements and composites.

---

## Envelopes

Each envelope in EDI data begins with a particular segment and ends with a particular segment. For example, the transaction set envelope begins with an ST segment and ends with an SE segment.

---

## Transaction sets

Transaction sets are made up of segments and loops. A segment contains a unit of information, for example, a line item or a patient record.

Each segment begins with a unique initiator that distinguishes it from all other segments. A segment also has a terminator that tells when the segment ends.

A loop is a repeating pattern of segments and other loops.

## Segments

Segments are made up of data elements and composites.

A data element is the basic unit of an EDI transaction. Data elements are the items of EDISegments, which are delimited and a delimiter separates their components. The value of this delimiter appears in the interchange envelope header, the ISA segment.

A composite is a group of related data elements.

## EDIFACT data

EDIFACT data is similar to ANSI X12 data. The difference is that the business document in EDIFACT data is called a message and a loop in EDIFACT data is called a group.

The EDIFACT standard includes many composites, whereas their use in ANSI X12 is less extensive.

## TRADACOMS data

The TRADACOM standard is primarily used in England and trading partners throughout Europe. TRADACOMS messages are used for domestic trade within the UK and cover a range of commercial transactions plus reports and master files.

There are twenty-six published files. These messages are updated as required to meet changing business needs. The following is an example of TRADACOMS data:

```
STX=ANA:1+5013546009111:AB TRADING COMPANY LIMITED+5013546002222:SMITHS
LIMITED+920106:175200+1+GANDALF+INVTES+B'
MHD=1+INVFIL:6'
TYP=0700+INVOICES'
SDT=5013546009227:9397706+AB TRADING COMPANY LTD+HILLSBOROUGH
WORKS:LANGSETT ROAD:SHEFFIELD:SOUTH YORKSHIRE:S6 2LW+172482067'
CDT=5013546002693+SMITHS LTD+(9397706):INTWOOD ROAD:CRINGLEFORD ,
NORWICH:NORFOLK.:NR4 6XB'
FIL=1+1+920106'
FDT=920106+920106'
MTR=7'
MHD=2+INVOIC:8'
CLO=5013546002693::533589/651+SMITHS LTD+SMITHS LTD:
127 CRAIGHALL RD:PORT DUNDAS:GLASGOW'
IRF=28138501+920106+920106'
PYT=2.50% - 30 DAYS MONTHLY ACCOUNT '
ODD=1+6516923::911210+03431901:920106'
ILD=1+1+:240100752+++:M2+51:45900:M+81338:M2+3733400+S+17500
+++COOLAG STANDARD ROOFBOARD:*75mm x 750mm x 1200mm++120500+1797559+32500'
STL=1+S+17500+1+37334++++37334+933+36401+6370+43704+42771'
TLR=1+37334++++37334+933+36401+6370+43704+42771'
MTR=9'
MHD=3+INVOIC:8'
CLO=5013546002693::533589/141+SMITHS LTD+SMITHS LTD:COPPS ROSD:
FLEETWOOD:LANCASHIRE:NR4 6XB'
IRF=06138502+920106+920106'
PYT=1+2.50% - 30 DAYS MONTHLY ACCOUNT '
ODD=1+1413315::920102+03803601:920103'
ILD=1+1+:320200026+++:M+28:28000:M+114545:M+3207300+S+17500
+++MASTERBOARD DOOR FACING:6mm x 2135mm x 915mm++147800+931140+22500'
STL=1+S+17500+1+32073+====32073+802+31271+5472+37545+36743'
TLR=1+32073+====32073+802+31271+5472+37545+36743'
MTR=9'
MHD=4+INVOIC:8'
CLO=5013546002693::533589/698+SMITHS LTD+COLLECTION'
IRF=04138503+920106+920106'
PYT=2.50% - 30 DAYS MONTHLY ACCOUNT '
ODD=1+6986217::920106+03852501:920106'
ILD=1+1+5012061000641:100153265+++:M2+12:104880:ZZ+14000:M2+
1468300+S+17500+++FIBREGLASS DRITHERM:75mm x 455mm x
1200mm (16 pp)++14000+0+0'
STL=1+S+17500+1+14683++++14683+367+14316+2505+17188+16821'
TLR=1+14683++++14683+367+14316+2505+17188+16821'
MTR=9'
MHD=5+VATTLR:6'
VRS=1+S+17500+84090+81988+14347+98437+96335'
MTR=3'
MHD=6+INVTLR:5'
TOT=84090+81988+14347+98437+96335+3'
MTR=3'
END=6'
```

## EDI version release schemas

The data for each EDI version is defined in a separate schema. For example, the ansi3070 schema defines ANSI X12 data version 3070.

Each EDI schema is arranged in a similar way, so you can easily define your specific EDI trading relationships.

## Types in EDI schemas

The type names in EDI schemas are consistent with the terminology used in the EDI standards.

The root of each schema is called EDI. Each schema has an ANSI or EDIFACT category, an Interchange category, and a Transmission category.

You can combine versions and standards by selecting one schema and copying or merging other versions or standards to it. You can create a schema, which includes separate definitions for each of your trading partners by customizing a schema for one partner and then copying or merging types with definitions of other partners.

An EDI version schema includes the entire data dictionary for that particular version and the transmission and interchange objects common to all versions.

The type names of data elements are abbreviated versions of the element descriptions. For example, the name of the type that defines the data element Account Type Code is Acc'tTypeCd. If you find these or other names inappropriate for your use, rename the type - every reference to that type is renamed for you automatically. For a list of abbreviations that are used in EDI schemas and their corresponding full name, see "EDI schema abbreviations."

The data objects of a particular standard are located under a category with the version name, for example, V3050 in the ansi3050 schema. The names of particular elements, segments, transaction sets, and messages do not change from schema to schema unless the standard, itself, has changed. This naming convention makes it easier to migrate from one version to the next, if the standard does not change.

## Modifying EDI schemas

Make your own copy of each EDI schema you are likely to use.

### About this task

For example, if you are using ANSI 3050 data:

1. Open the ansi3050 schema.
2. Choose Save As from the File menu.
3. Rename this schema something new, for example, my3050.

You can then modify your own version schema. If you inadvertently delete something you need, or later decide to add more, open the original schema and copy the missing information to your own schema.

You should customize your EDI schema to suit your trading needs. For information on modifying your schema to include only those types you need, see "Create industry subsets".

## Analysis of EDI schemas

Before an EDI version schema is released, the schema is analyzed by using the Type Tree Analyzer.

Some schemas produce analysis errors because of the way data objects are defined in EDI. Most of the analysis errors were resolved by changing the schemas. These changes eliminate the analysis errors, but keep the EDI definitions intact.

A few errors cannot be resolved because resolving them requires knowledge of the user's specific data structure.

This documentation explains changes that were made to the EDI schemas and the analysis errors that occur when you analyze each EDI schemas. It also explains what to do about these errors.

- **Summary of errors**

In some cases, the size of an element code, which is defined as an item restriction, was greater than the maximum allowable size for that item.

- **X12 transaction sets that did not pass analysis**

When the definition of a transaction set is in error, you receive one error for the inbound transaction and one for the outbound one.

## Summary of errors

In some cases, the size of an element code, which is defined as an item restriction, was greater than the maximum allowable size for that item.

In these cases, the restriction was deleted:

ANSI schema	Element Type
ansi2003	AmendmentTypeCd Element
ansi2003	CommodityGeologicalConnectorCd Element
ansi2040	AmendmentTypeCd Element
ansi3030	ActionCd306 Element

Note: The type trees listed in the table are installed in the directory:  
*install\_dir\supplychain\_edi\_vn.n.n\x12\trees*

Some of the transaction sets have ambiguous definitions. These transaction sets are removed from the schemas. If you need to use any of these transaction sets, you must modify them according to how you plan to use them.

Analysis uncovered a few transaction sets whose definitions are ambiguous. These transaction sets were not removed from the schemas. However, if you use any of these transaction sets, they must be modified in accordance with how you use it.

In cases where it was possible to change the definition of a transaction set to make it unambiguous, it was changed. Verify the changes before you use that transaction set. If a solution was not possible, without information on the data structure, the transaction set was left in its ambiguous state.

Note: If you do not use a transaction set that causes an analysis error, delete it from the schema. You can always copy it again from the original EDI schema if you begin to use it.

Transaction sets that cause analysis errors are detailed here:

Schema	Transaction Set	Analysis Error
ansi2003	#830	optional segments not distinguishable
ansi2040	#830	optional segments not distinguishable
ansi3010	#830	optional segments not distinguishable
ansi3020	#110	optional segments not distinguishable
	#838	data objects of same component not distinguishable
	#426	optional segments not distinguishable
ansi3030	#838	data objects of same component not distinguishable
	#861	optional loops not distinguishable
	#304	optional segments not distinguishable
	#404*	blocked loops not distinguishable
	#417*	blocked loops not distinguishable
ansi3060	#304	optional segments not distinguishable
ansi3070	#304	optional segments not distinguishable
ansi4010	#304	optional segments not distinguishable
ansi4020	#304	optional segments not distinguishable
ansi5040	#355	optional loops not distinguishable

\* The schemas were changed so these errors do not occur. Explanations of how the schemas were changed are found in the documentation provided here.

## X12 transaction sets that did not pass analysis

When the definition of a transaction set is in error, you receive one error for the inbound transaction and one for the outbound one.

For example, there are two errors in the ansi2003 schema that are related to the LoopLIN of the #830 transaction set, one for inbound and one for outbound.

The inbound and outbound errors are the same. Only the inbound errors are addressed here.

- **[Delete X12 transaction sets in error if not used](#)**

If you do not use a transaction that causes an analysis error, delete both occurrences of it, that way, when you analyze your schema, you do not get the errors. Deleting causes the analysis to run faster. You can delete the entire transaction set category under the Inbound category and the entire transaction set category under the Outbound category.

- **[ANSI2003 the #830 transaction set](#)**

The same error that occurred in the #830 transaction set in ANSI version 2003 also occurred in ANSI version 2040.

- **[ANSI3010 the #830 transaction](#)**

The same error that occurred in the #830 transaction set in ANSI version 2003 also occurred in ANSI version 3010.

- **[ANSI3020 the #110 transaction](#)**

The #110 transaction set causes two errors.

- **[ANSI3020 the #838 transaction](#)**

The ansi3020 analyze error L201 is different from the L199 error.

- **[ANSI3060, ANSI3070, ANSI4010, and ANSI4020 the #304 transaction](#)**

The L199 analyzer error occurs when the ansi3060, ansi3070, ansi4010, and the ansi4020 schemas are analyzed.

## Delete X12 transaction sets in error if not used

If you do not use a transaction that causes an analysis error, delete both occurrences of it, that way, when you analyze your schema, you do not get the errors. Deleting causes the analysis to run faster. You can delete the entire transaction set category under the Inbound category and the entire transaction set category under the Outbound category.

In addition, delete the Funct'IGroups, under Inbound and Outbound, that contain the deleted transaction. Then, analyze the schema again.

For example, the #838 transaction set in the ansi3030 schema causes an analysis error. If you are using ANSI version 3030, and you are not using the #838, you can delete the entire #838 category for the transaction set under Inbound and under Outbound.

Also, delete the Funct'IGroup type #838, under Inbound and under Outbound.

For information on customizing your own EDI type tree, see the Create industry subsets documentation.

If you do use a transaction set that causes an analysis error, refer to the explanation of the errors that follow. Then, determine how you want to fix it.

## ANSI2003 the #830 transaction set

After you analyze the ansi 2003 schema, the following analyzer message appears:

L199 - COMPONENT 4 is not distinguishable from COMPONENT 7 that may follow in TYPE 'LoopLIN #830 Inbound Partner Set V2003 ANSI EDI' (error)

In the #830 transaction set type, there is a LoopLIN type. Within the LoopLIN type, there is a LoopSLN type. LoopSLN ends with an optional PID Segment. However, there is also a PID Segment later in the component list of the LoopLIN. In between, the PO3 Segment and the CTP Segment are optional. This means that if a PID Segment appears in the data, it may be the one in the LoopSLN or the one in the LoopLIN.

Possible solutions for making the transaction set unambiguous include the following:

- Make at least one occurrence of the PO3 Segment or CTP Segment required, make its component range minimum at least 1.
- Remove one of the PID Segment components.

## ANSI2040 the #830 transaction

The same error that occurred in the #830 transaction set in ANSI version 2003 also occurred in ANSI version 2040.

See ANSI2003 the #830 transaction set documentation for more information.

## ANSI3010 the #830 transaction

The same error that occurred in the #830 transaction set in ANSI version 2003 also occurred in ANSI version 3010.

See the ANSI2003 the #830 transaction set documentation.

## ANSI3020 the #110 transaction

The #110 transaction set causes two errors.

The first analyze error defines indistinguishable components with optional components in between the SL1 Segments in the ansi3020 schema and might read as follows:  
L199 - COMPONENT 14 is not distinguishable from COMPONENT 17 that may follow in TYPE 'Transaction #110 Inbound Partner Set V3020 ANSI EDI' (error).

The 14th and 17th components of the #110 transaction set are the LoopLX and the SL1 Segment.

The problem is an ambiguous SL1 Segment; there is one in the actual transaction set. It might be confused with the one nested in the LoopLX within the LoopL5.

You must modify the definition according to your own specifications. Here is a possible solution for making the transaction set unambiguous:

- Remove one of the S1 Segment components.
- Make either the L4 Segment or L3 Segment components that are required by changing the component range minimum to at least 1.

The second analyze error for the ansi3020 schema #110 transaction set also identifies indistinguishable components:

The 14th and 18th components of the #110 transaction set are the LoopLX and the L10 Segment.

This problem is similar to the first error for this transaction. The L10 Segment in the actual transaction set and the one nested within the LoopLX are indistinguishable because optional components fall in between.

Modify the definition according to your own specifications. Here is a possible solution for making the transaction set unambiguous:

Remove one of the L10 Segment components.

- Make either the L3, L4, or SL1 components outside of LoopL5 and LoopL1 required.

## ANSI3020 the #838 transaction

The ansi3020 analyze error L201 is different from the L199 error.

For example, the following L201 message indicates that in the fourth component you cannot distinguish between one occurrence and the next occurrence: L201 - Different data objects of COMPONENT 4 are not distinguishable in TYPE 'LoopPLA #838 Inbound Partner Set V3020 ANSI EDI' (error). In this case, in the 4th component you cannot distinguish between one occurrence and the next occurrence.

In this case, the LoopN11 in the LoopPLA begins with an N1 Segment. Nested within it is another N1 Segment. The Nested N1 Segment is optional and all of the components that follow it are optional. If a second N1 Segment appears in the data, is it the beginning of another LoopN11? Or, is it the N1 Segment of the LoopTUD?

Note: If you use this transaction set, the best way to fix it is to contact your trading partner and ask how it is to be interpreted.

## ANSI3060, ANSI3070, ANSI4010, and ANSI4020 the #304 transaction

The L199 analyzer error occurs when the ansi3060, ansi3070, ansi4010, and the ansi4020 schemas are analyzed.

The #304 transaction set in ANSI version 3060 also occurs in ANSI version 3070, 4010, and 4020. The LoopPO42 contains an N9 Segment, which is indistinguishable from the N9 following the LoopPO42.

Modify the definition according to your own specifications. Remove one of the N9 Segment components or make one of the optional components between the indistinguishable N9 Segment segments required.

## Mapping purchase orders to invoices

An example map maps an ANSI X12 EDI transmission that contains purchase orders (#850 transaction sets) to an ANSI X12 EDI transmission that contains invoices (#810 transaction sets).

The example illustrates how to use the SDQ segment to split the purchase order data into the appropriate invoices.

The sdq.mms file installs in the following directory:

*install\_dir\supplychain\_edi\_vn.n.n\x12\examples\ansi\sdq*

The sdq file installs with the imported project.

- **By line item**  
An SDQ segment problem may occur when mapping from purchase orders by line item to invoices by store.
- **Purchase order**  
This example purchase order show how the various line items are handled.
- **Invoices to be received**  
The following example shows invoices to be received, one interchange for each store.
- **ByStore map**  
In the functional map ByStore, the #810 transaction set is produced by using another functional map, ByPO.
- **ByPO map**  
In the functional map ByPO, another functional map is called to produce the IT1 Loops, the line items of the invoice.
- **ByLine map**  
In the map ByLine, the quantity invoiced is found in the SDQ Segment of the line item in the PO.

## By line item

An SDQ segment problem may occur when mapping from purchase orders by line item to invoices by store.

You may have an input transmission that may contain multiple interchanges from your trading partner who sends you EDI purchase orders. A purchase order may be organized so that each line item identifies a set of stores that receive some quantity of the merchandise identified in that line item.

Your trading partner wants you to send an interchange for each store. The interchange must include one invoice for each inbound purchase order that references that store. Each invoice includes only the line items relevant to that particular store. You may want to receive invoices, one interchange for each store.

There are two inbound purchase orders that refer to store #0100, so the interchange for that store has two invoices. Each invoice for store #0100 references just those line items of the purchase order that had store #0100 identified in an SDQ Segment that was part of that line item.

The executable map, SDQ, has two inputs and two outputs. The inputs are an EDI transmission containing purchase orders, and a trading partner profile. The outputs are an EDI transmission containing invoices, and the updated trading partner profile.

## Purchase order

This example purchase order show how the various line items are handled.

The following example shows one purchase order:

```
ISA*00* *00* *08*WALLY *01*ME *180820*0500*^^00704*00000002*0*T*:
GS*PO*5012738712*18130529*20180920*0500*2*x*007040
ST*850*0005*007040VICS
BEG*00*SA*0037324913**20180820
REF*DP*32
REF*IA*181305320
ITD*01*7*****30***** N30
TD5*O*ZZ*12345**UNITED PARCEL
A {
 PO1*001*150*EA*3.9*LE*UP*070135150611*CB*003234689*VC*1001
 PO4*25*6*EA
 SDQ*EA*92*0100*12*0131*6*0242*6*0548*12*0639*12*0686*6*0749*6*0802*6*1008*6*1027*30
 SDQ*EA*92*1041*6*1044*6*1294*6*1297*12*1800*6*1802*12
B {
 PO1*005*156*EA*3.25*LE*UP*070135160313*CB*003257128*VC*1006
 PO4*26*6*EA
 SDQ*EA*92*0100*12*0131*6*0639*6*0686*6*0749*6*0802*6*0824*6*0847*6*1008*6*1027*30
 SDQ*EA*92*1041*6*1044*6*1241*6*1294*6*1297*12*1392*12*1800*6*1802*12
```

```

C {
 PO1*003*144*EA*3.25*LE*UP*070135170312*CB*003257144*VC*1004
 PO4*24*6*EA
 SDQ*EA*92*0242*6*0639*12*0686*6*0749*6*0784*12*0824*6*0847*6*1008*6*1027*30*1041*6
 SDQ*EA*92*1044*6*1241*6*1294*6*1297*12*1800*6*1802*12
D {
 PO1*002*108*EA*3.9*LE*UP*070135152318*CB*003257169*VC*1002
 PO4*18*6*EA
 SDQ*EA*92*0158*6*0639*12*0686*6*0749*6*1008*6*1027*30*1044*6*1294*6*1297*12*1800*6
 SDQ*EA*92*1802*12
E {
 PO1*004*174*EA*3.25*LE*UP*070135140315*CB*003257185*VC*1005
 PO4*29*6*EA
 SDQ*EA*92*0158*6*0520*18*0639*12*0686*6*0749*6*0784*6*0802*6*0824*6*0847*6*1008*6
 SDQ*EA*92*1027*30*1044*6*1241*6*1294*6*1297*12*1305*6*1392*12*1800*6*1802*12
 CTT*5
 SE*28*0005

```

#### Reference

##### Remarks

- A Send this line item to the 16 stores that are referenced in the SDQ segments.
- B Send this line item to the 18 stores that are referenced in the SDQ segments.
- C Send this line item to the 16 stores that are referenced in the SDQ segments.
- D Send this line item to the 11 stores that are referenced in the SDQ segments.
- E Send this line item to the 19 stores that are referenced in the SDQ segments.

## Invoices to be received

The following example shows invoices to be received, one interchange for each store.

```

ISA*00*ASCENTIAL *00*00 *01*ME *ZZ*WALLY *180822*1142*^^00704*00000010*1*p*:
ISX*!
GS*IN*ME*WALLY*20180822*11422373*1*X*007040
ST*810*1
BIG*20180822**20180820*0037324138*
REF*DP*32
REF*IA*181305320
N1*ST*0100
IT1*003*12*EA*3.25*LE*UP*070135170312*CB*003257144*VC*1004
PO4*11*6*EA
IT1*004*12*EA*3.25*LE*UP*070135140315*CB*003257185*VC*1005
PO4*19*6*EA
TDS*2400
CTT*2
SE*12*1
ST*810*2
BIG*20180822**20180820*0037324913*
REF*DP*32
REF*IA*181305320
N1*ST*0100
IT1*001*12*EA*3.9*LE*UP*070135150611*CB*003234689*VC*1001
PO4*25*6*EA
IT1*005*12*EA*3.25*LE*UP*070135160313*CB*003257128*VC*1006
PO4*26*6*EA
TDS*2400
CTT*2
SE*12*2
GE*2*1
IEA*1*000000010

```

The store number is indicated by the IDCd element of the N1\*ST\*0100 segment. In the example shown here, the store number is N1\*ST\*0100. The object is to generate an interchange for each store indicated in the data from your trading partner Wally. For example, if there are a total of three different stores in the data from your trading partner, you want to generate three interchanges.

To map each output interchange, the functional map ByStore is used.

The `UNIQUE` and `EXTRACT` functions are used to generate one Interchange for each unique value of the IDCd Element within the SDQ Segment. The second argument of ByStore is the entire input transmission. To map just the data from your trading partner, the `LOOKUP` function is used to look up the information in the trading partner profile. The last argument of ByStore is the `INDEX` of the current interchange being produced, which are used to generate the new ISA control number.

## ByStore map

In the functional map ByStore, the #810 transaction set is produced by using another functional map, ByPO.

Only the #850 transaction sets that have data for the store are sent to the functional map ByPO

## ByPO map

In the functional map ByPO, another functional map is called to produce the IT1 Loops, the line items of the invoice.

## ByLine map

In the map ByLine, the quantity invoiced is found in the SDQ Segment of the line item in the PO.

The SDQ map references the ByStore functional map. ByStore maps the data according to store number, and calls ByPO. ByPO maps the data according to PO, and calls ByLine. Finally, ByLine maps the data according to line item.

## Editbol map source file

The editbol map source file maps an EDI data file of ANSI X12 Motor Carrier Shipment Information (transaction #204) to an application file that contains bill of lading information.

The editbol.mms map source file installs in this directory:

*install\_dir\supplychain\_edi\_vn.n\X12\examples\ansi\editbol*

The editbol map source file installs with the project.

- [Mapping inbound EDI #204 transaction sets](#)

The input schema file is ANSI X12 Standard..

## Mapping inbound EDI #204 transaction sets

The input schema file is ANSI X12 Standard..

The schema file installs with the imported project.

The type\_tree is installed in this directory:

*install\_dir\packs\supplychain\_edi\_vn.n\X12\examples\ansi*

The output type tree file is bol.mtt, and is installed in this directory:

*install\_dir\packs\supplychain\_edi\_vn.n\X12\examples\ansi\editbol*

The input type of the executable map is Partner X12 Inbound Transmission.

The output type is File, which is your file that contains any number of bills of lading. Each bill of lading is composed of one header record and many detail records. Each header record has several fields, including repeating sets of name and address information. These sets are defined in the schema as a single object called NameSet, which is used three times. The detail record object contains five fields. These records are fixed-length records, where every field is mandatory and no delimiters are used. Header records are initiated by 01, and detail records by 02.

The output file is made up of BillOfLading(s). The EachBOL functional map maps each #204 transaction set to the output bill of lading.

- [EachBOL functional map](#)

In the EachBOL functional map, the output is a single bill of lading, which is made up of one header record and a series of detail records.

- [MakeNameSet functional map](#)

- [EachDetail functional map](#)

In the EachDetail map, the components of the output Detail Record are mapped from the LoopLX and the #204 transaction set.

## EachBOL functional map

In the EachBOL functional map, the output is a single bill of lading, which is made up of one header record and a series of detail records.

Each header record type is defined in the Type Designer with the initiator 01, and each detail record was defined as beginning with the initiator 02.

The header record is made up of ten components, three of which contain repeating patterns of name and address information. The rule for each component selects the necessary element from the appropriate segment from the inbound #204, or assigns a literal.

In the case of the repeating name and address section of the header record, **NameSet**, the correct occurrence of this loop must be selected to properly populate the user's file with either **ShipTo**, **BillTo**, or **Consignee** name and address data. This has been accomplished by using the **IF** function to test the **N1 Loop**, then calling a functional map to create each component of that **Name Set**. For example, the rule for **NameSet[1]** tests the **N1** segment for an **EntityIDC Element of CN**.

The rules of the three **NameSet** components reference three different **N1** loops: one for **Consignee**, one for **Ship To**, and one for **Bill To**.

## MakeNameSet functional map

The **MakeNameSet** functional map has the **N1 Loop** as the input.

Again, in the **EachBOL** map, the rule for the **Detail Record(s)** calls another functional map, **EachDetail**. This map is responsible for creating each individual **Detail Record** within a **BillOfLading**. The input arguments for this map are the **Transaction #204** and the **LoopLX**. From these two areas come the segments that make up the **DetailRecord**.

## EachDetail functional map

In the EachDetail map, the components of the output Detail Record are mapped from the LoopLX and the #204 transaction set.

## X12 schemas

The Pack for Supply Chain EDI contains X12 schemas. The X12 schemas are used to validate and ultimately transform data that conforms to the ANSI X12 standard, as described in the ANSI X12 specifications.

- [About the X12 Standard](#)  
The standards are maintained by the Accredited Standards Committee X12 (ASC X12) which is chartered by the American National Standards Institute (ANSI).
- [Installation](#)  
The X12 schemas are available for download from the ESD website. Your user ID and password are required to log in.
- [X12 schema details](#)  
Documentation that describes details about the X12 schemas including the ANSI, Interchange, and Transmission categories is provided here.
- [X12 examples](#)  
Example maps and schemas install with the Pack for Supply Chain EDI or within the Design Server imported project x12Examples.zip.

## About the X12 Standard

The standards are maintained by the Accredited Standards Committee X12 (ASC X12) which is chartered by the American National Standards Institute (ANSI).

Standards information and related data embedded in the [Licensee] Software, Outsourcing Services and Combined Software Solutions are protected by copyright and other the intellectual property of X12 Incorporated, a non-profit organization chartered by the American National Standards Institute to develop and maintain EDI standards and XML schemas. In order to provide and fund these activities in serving American industry, X12 is willing to grant to all end users of the Licensee Software, Outsourcing Services or Combined Software Solutions a license at a royalty of \$180 per year. IBM strongly recommends that you contact X12 at [Licensing Partner: IBM](#) to obtain a license to use and access X12 metadata, represented in, for example, more than 30 versions of more than 300 transaction sets and implementation guides, through X12's online viewer, Glass.

## Installation

The X12 schemas are available for download from the ESD website. Your user ID and password are required to log in.

## X12 schema details

Documentation that describes details about the X12 schemas including the ANSI, Interchange, and Transmission categories is provided here.

The schemas that are referenced in this documentation can be found in the Pack for Supply Chain EDI.

- [Overview](#)  
The root type of each ANSI X12 schema is called EDI.
- [ANSI category](#)  
The ANSI category has three subtypes: Control, Funct'lGroup,, and a V (*version*) category.
- [Interchange category](#)  
Double-click the X12 Inbound Interchange type to view the components of the interchange.
- [Transmission category](#)  
A transmission is made up of interchanges, so a Transmission type is made up of Interchange types.
- [Mapping ANSI data](#)  
When a map for mapping ANSI data is created, create an input or output card to represent the ANSI data.
- [Input card - receiving ANSI data from multiple partners](#)  
When the input consists of only one interchange, select an Interchange type that corresponds to the correct Transmission type.
- [Output card - sending ANSI data to multiple partners](#)  
When the output consists of only one interchange, select an Interchange type that corresponds to the Transmission type.

## Overview

The root type of each ANSI X12 schema is called EDI.

Each schema has an ANSI, Interchange, and Transmission category that stems from the EDI root type. For example, the following ansi4020 schema contains the ANSI X12 version 4020 and the ansi4030 schema contains the ANSI X12 version 4030.

- **ANSI**  
These types are subtypes of the ANSI category.
  - **Interchange**  
The Interchange type contains the data types for an interchange.
  - **Transmission**  
The Transmission type contains the data types for a transmission, made up of one or more interchanges.
- 

## ANSI

These types are subtypes of the ANSI category.

- Control type defines the ANSI control structure objects. The Control category is the same on each ANSI version schema.
  - Funct'lGroup type defines the functional groups for the version and the data objects for that particular version.
  - V category defines the version-specific data objects. For example, V4030 is the name of the category in the 4030-version schema (ansi4030).
- 

## Interchange

The Interchange type contains the data types for an interchange.

The Interchange category contains Inbound and Outbound partitioned group subtypes.

## Transmission

The Transmission type contains the data types for a transmission, made up of one or more interchanges.

The Transmission category contains Inbound and Outbound group subtypes.

The ANSI schemas include both inbound and outbound types, where appropriate. In addition, types that are called Partner define a generic trading partner.

## ANSI category

The ANSI category has three subtypes: Control, Funct'lGroup, and a V (*version*) category.

- **Control subtree**  
The subtypes of the Control category are the same in each ANSI schema.
  - **Funct'lGroup subtree**  
The types under the Funct'lGroup category represent functional groups in the X12 version.
  - **Version-specific subtree**  
The types specific to the ANSI version are found under the ANSI category.
- 

## Control subtree

The subtypes of the Control category are the same in each ANSI schema.

These types are used in ANSI envelope segments. Expand the Control category, under ANSI, to see the control types.

The following are subtypes of Control.

- Delimiter Items that are used to define syntax objects.
- Element Items that are used as envelope segment components.
- ISAPartnerInfo Components of the ISA segment.
- Segment Envelope segments.
- Variant Definitions of envelopes in other standards.
- **Segment group**  
The Segment group type in the Control category represents the envelope segments. Envelope segments are used:
- **ISA segment**  
The ISA segment is the first segment in an ANSI X12 Interchange.
- **Interchange acknowledgment**  
The TA1 segment is used as an interchange acknowledgment. It can be used in an interchange, just before the functional groups begin.
- **Interchange syntax extension**  
The ISX segment was introduced with the X12 version 7040 as an optional control segment. It can be used by the interchange sender to identify a release character and for character encoding.

- **Control elements**  
Elements under the Control category are the items that are used as components of the group types under the Control category.
- **Variant category**  
Envelope segments and elements for UCS (Uniform Communication Standard) are found under the Variant category.
- **Delimiter types**  
The Delimiter types are syntax items. They are used as element delimiters, composite delimiters, and terminator delimiters in the ANSI segments.

---

## Segment group

The Segment group type in the Control category represents the envelope segments. Envelope segments are used:

- To surround functional groups within an ANSI X12 interchange, the ISA, and IEA segments in X12.
- For interchange acknowledgments, the TA1 segment.
- To specify interchange delivery and handling requirements, the ISB, and ISE segments.

Each Segment envelope type begins with a set of letters that identifies the segment, for example IEA. These same sets of letters (IEA) are also used in the Properties window of the Type Designer to define the value of the initiator.

---

## ISA segment

The ISA segment is the first segment in an ANSI X12 Interchange.

The ISA segment contains data that identifies trading partners. This segment also specifies the delimiters and terminator within the interchange. The segment does not identify which X12 version data is contained in the interchange. This version information is in the GS segment, which is discussed in the [Funct'lGroup subtree](#) section.

The type ISA has an Inbound and an Outbound subtype, each of which has a Partner subtype. The Inbound ISA type is used as a component of the Inbound Interchange type. The Outbound ISA type is used as a component of the Outbound Interchange type.

The second component of ISA is the type ISAPartnerInfo, which has the following components:

- Auth'nInfoQual'r Element
- Auth'nInfo Element
- SecurityInfoQual'r Element
- SecurityInfo Element
- Sender InterchangeIDQual'r Element
- InterchangeSenderID Element
- Receiver InterchangeIDQual'r Element
- InterchangeRcv'rID Element
- InterchangeDate Element

This is an example of ISA segment data:

```
ISA*00*TSI *01*92511930 *01*ME *12*BRADLEY
*970815*1732*U*00201*000000050*0*T*
```

This ISA example wraps around on the page, but there is no CL/LF in the ISA segment.

In the data of an ISA segment, the first thing after the initiator ISA is the element delimiter, so the first component of the ISA type is Element Delimiter. The value of the delimiter can be any of a number of different values. In input data, after the delimiter value is found as the first component of the ISA. It is set as the delimiter value for the rest of segments in the entire interchange. Likewise, the Composite Delimiter is the third component of the ISA, and the terminator is defined as the last component of the ISA.

When mapping from EDI data, the syntax values can be mapped from the components of the ISA in the input. When you map to EDI data, you can set the syntax values by assigning values to the components of the ISA.

Note: The default value for the delimiter is an asterisk \*, and the default value for the terminator is a carriage return/linefeed <CR><LF>. If you want to test inbound data that does not contain the interchange envelope, the data must contain the default delimiter and terminator values, or the data is invalid. If the delimiter and terminator are different from the defaults, change the defaults in your schemas.

The IEA segment is the last segment in an interchange. The IEA segment has the same interchange control number that is in the ISA. The IEA segment also has a count of the functional groups within the interchange.

---

## Interchange acknowledgment

The TA1 segment is used as an interchange acknowledgment. It can be used in an interchange, just before the functional groups begin.

Typically used only by network providers, an interchange containing TA1s consists of the TA1s surrounded by the ISA and IEA segments; it does not typically contain any functional groups.

---

## Interchange syntax extension

The ISX segment was introduced with the X12 version 7040 as an optional control segment. It can be used by the interchange sender to identify a release character and for character encoding.

If the ISX appears in the data, the values that occur in the ISX set the active syntax values. If there is no ISX, the release character is assumed to be an exclamation point.

## Control elements

Elements under the Control category are the items that are used as components of the group types under the Control category.

For example, elements that appear in the ISA, ISB, and TA1 segments are defined here. Expand the Element category to view an alphabetical list of the elements.

Most of the definitions of the Element subtypes conform to the interchange standard, not the version standard. The main difference is the interpretation of the `NO` notation that is used in X12. The `NO` notation in the interchange standard refers to an unsigned integer. In the version standard, the `NO` notation refers to an integer, possibly preceded by a negative sign.

For example, the `InterchangeCtrl#` and `#InclFunct'lGroups` elements are part of the interchange standard and are defined in the schema as an unsigned integer. The `GroupCtrl#` Element, however, appears as part of a version standard, and is therefore defined as an ANSI implicit decimal with zero decimal places.

## Variant category

Envelope segments and elements for UCS (Uniform Communication Standard) are found under the Variant category.

The BG segment appears at the beginning of a UCS interchange, and the EG segment appears at the end. Elements that appear in the BG and EG segments are subtypes of the Element type.

## Delimiter types

The Delimiter types are syntax items. They are used as element delimiters, composite delimiters, and terminator delimiters in the ANSI segments.

The values of these syntax objects might be different in different interchanges. The Element Delimiter appears as the first component of the ISA. The Composite Delimiter appears as the third component of the ISA, following the ISAPartnerInfo data fields. The Terminator appears as the last component of the ISA. For X12 versions 7040 and later, the **Release Character** appears in the first component of the ISX segment and defaults to exclamation point if the ISX is not used. For information about delimiters that are specified as syntax items, see the Type Designer documentation.

After the values of the syntax objects are set by the ISA, they are used for the rest of the segments in that interchange. For example, the Element Delimiter value of the IEA segment is specified as the item Element Delimiter. The delimiter value that is found in the ISA is the same value in the IEA.

The following item window displays the Include restrictions list for the Element Delimiter type. They are the possible values of the delimiter between elements.

Note: If a trading partner uses a value that is not in the restriction list, you can add the new value to the list.

## Funct'lGroup subtree

The types under the Funct'lGroup category represent functional groups in the X12 version.

To allow for partner-specific definitions, the functional group types are organized under the category Partner, representing functional groups that are exchanged with a generic trading partner. Under Partner, the types are then organized by Inbound and Outbound. Subtypes of Inbound and Outbound are identical. Inbound and Outbound both have a subtype that is named F, followed by the X12 version. For example, F4030 is the name of the functional group type in the ansi4030 schema.

The first segment of a functional group is the GS segment. The GS segment includes information about trading partners, the X12 version of the data, and the types of transaction sets that are contained within the functional group.

Expand the Funct'lGroup category to see all the functional groups in the X12 version. Each functional group type is named according to the kinds of transaction sets that are contained within it. There are two possibilities for the type name of a functional group: name the type the transaction set number or name the type after the value of the element.

- If only one transaction set can appear within the functional group, its type name is the transaction set number. For example, in the functional group that contains purchase orders, the transaction set #850, can contain only purchase orders. It cannot contain any other transaction set. Therefore, the name of the functional group type is #850.

The following group window displays the components of the functional group #850 in the ansi4030 schema. Notice that the second component is a series of #850 transaction sets.

- If there can be more than one transaction set within a functional group, the type name of the functional group is the value of the FunctionalIDCd Element. For example, the CF functional group can contain both #844 and #849 transaction sets. The type name of the functional group, then, is the value of the FunctionalIDCd Element, which is CF. Each functional group has a unique value for the FunctionalIDCd Element.

Note: There is one exception to the naming convention. One functional group has a FunctionalIDCd Element whose value is IN. This functional group can contain Invoices, the #810, and Operating Expense Statements, the #819. Because IN is a reserved word in the Design Studio, the type name cannot be IN, so its name is IE.

As more industries join the EDI community, the list of functional groups gets longer. When you make your own schema from an ANSI schema, delete the functional groups that you have no interest in. For example, if you create your own schema from the ansi4030 schema, delete the functional group types under F4030, under Inbound, that you are not currently receiving. Then, do the same for the F4030 group under Outbound; delete the functional groups that you are not sending out. Remember you can always add them back later.

## Version-specific subtree

The types specific to the ANSI version are found under the ANSI category.

The name of the category where the types are organized begins with a V and is followed by the ANSI version. For example, the name of the type in the ansi4030 schema is V4030.

- **Elements**

The item subtypes of the Element category are the ANSI data elements.

- **XComposites and MComposites**

A composite is a set of related data elements.

- **Segments**

A segment is a logical unit of information, like a data record.

- **Sets - transactions sets, loops, and blocks**

The transaction sets and the loops for the ANSI version are under the Set category.

- **Transaction sets are organized under categories**

Each transaction set appears under a category with the same name.

- **Loops and blocks**

The loops and blocks within a transaction set appear under the same category where the transaction set definition itself is located.

## Elements

The item subtypes of the Element category are the ANSI data elements.

The names of these types are abbreviations of the element descriptions in the Data Interchange Standards Association (DISA) documentation of the ASC X12 standards.

To view an element's full description, perform the following steps:

1. Select the type in the schema.
2. Right-click and select Properties from the menu.

The description of the X12 data element is in the Value column of the Description property.

There are over 1000 element types in an ANSI schema. Although they are arranged alphabetically, you might want to use the **Find** command on the Edit menu to locate a particular element.

The following list summarizes how an ANSI schema corresponds to the ANSI EDI standards:

- Element codes are specified as item restrictions in an ANSI schema.
- Different elements with the same X12 description are named by appending the X12 data element reference number to the end of the type name.
- Some element types have subtypes. Element subtypes were added if the same element was used more than once as a component of the same segment. For example, if a segment has two Date Element components, one might be an Arrival Date Element and the other a Departure Date Element.

## XComposites and MComposites

A composite is a set of related data elements.

The ANSI standards incorporated the first composite in version 3030. In an ANSI X12 schema, the X12 composite data structures are found under the XComposite category, where the X stands for X12.

In addition, other sets of elements that ANSI does not define as composites are defined as composites. These are located under the MComposite category. MComposites were formed to identify repetitive patterns of data elements and to group elements that are logically related.

Defining the composites in this manner simplifies type definitions and mapping rules. In addition, it is easier to identify which elements are required when others are present (or absent, as the case might be). For example, the third component of the ACT segment is MComposite:

The ANSI X12 documentation states the following rule about two elements of the ACT segment:

**P0304 - If either ACT03 or ACT04 is present, then the other is required**

(where ACT03 is the IDCdQual'r Element and ACT04 is the IDCd Element).

If the data elements were defined as components of the ACT segment, a complex component rule, equivalent to the ANSI rule above, would be necessary. Instead, the two elements were defined as components of an IDCd MComposite:

The IDCd MComposite is then a component of the ACT segment, and is optional with a range of (0:1). This means that if an IDCd MComposite is in the data, both of its components must exist. In addition, if there is no IDCd MComposite in the data, neither component is there. Defining the data in this way meets the requirements of the ANSI standards and simplifies the type definition.

Another difference between XComposite types and MComposite types is their delimiters. The delimiter of an XComposite type is the Composite Delimiter, which appears as data in the ISA. The delimiter of an MComposite is the Element Delimiter.

## Segments

A segment is a logical unit of information, like a data record.

The Segment type is partitioned by initiator values. Each segment begins with a unique 2 or 3 character initiator, which identifies the segment. For example, the ACT segment begins with the initiator ACT.

Because an Element Delimiter must follow the segment initiator value and precede the first data element, each segment is defined as prefix delimited by the Element Delimiter and has a terminator of Terminator Delimiter.

Components of a segment might be Element types, MComposite types, and XComposite types. The segments are defined to comply with the rules specified in the X12 documentation.

## Sets - transactions sets, loops, and blocks

The transaction sets and the loops for the ANSI version are under the Set category.

To allow for partner-specific definitions, the transaction set types are organized under the category Partner. Partner subtypes are then organized by Inbound and Outbound; these categories are identical.

## Transaction sets are organized under categories

Each transaction set appears under a category with the same name.

The name of the category is the same as the name of the FunctlGroup type that contains that particular transaction set. For example, the #102 transaction set is a subtype of the #102 category, just as the FunctlGroup category that contains the #102 transaction set is called #102.

The #844 transaction set is contained in a functional group that can have both #844 and #849 transactions. The functional group type name is CF, so the actual transaction set #844 is found under the category named CF.

If there is only one transaction set under a category, the transaction set type name is simply Transaction. If there is more than one transaction set under a category, the type Transaction is partitioned and the separate transactions are subtypes of it. For example, the actual #844 transaction set has the name #844 and is a subtype of Transaction.

The ST Segment is the first component of each transaction, and is used as an identifier, the value of the TSIDC Element in the ST Segment tells what transaction it is, in this case, #844, or #849.

If a transaction set cannot be found by its transaction set number, use the **Find** command. For example, if you did not know that the #844 transaction set type was under the CF category, you might find it by choosing Find from the Edit menu, and entering #844 in the Find What field.

## Loops and blocks

The loops and blocks within a transaction set appear under the same category where the transaction set definition itself is located.

Loops are groups of two or more semantically related segments, though in an actual X12 transmission, a loop can appear as only the single loop that starts a segment. Loops can also contain other loops as components.

X12 defines two types of loops: unbounded and bounded. In unbounded loops, the first data segment in the loop appears once and only once in each loop occurrence (marking the start of an occurrence). Bounded loops are similar, but have no restriction on which data segment begins a loop occurrence and require a loop start (LS) segment before the first loop occurrence and a loop end (LE) segment after the last loop occurrence.

The type name of a loop is Loop followed by the identifier of the first segment in that loop and, if defined for the transaction set, the numeric Loop ID value that is specified by X12. For example, X12 version 4010 defines two different loops in transaction set #124, which begin with segment LM, loop IDs 3200 and 3320. These loops are named LoopLM3200 and LoopLM3320 respectively. If no numeric Loop ID values are specified for a transaction set and the same segment ID begins more than one loop, the sequential position of each loop within the transaction set is used to create unique loop names.

For example, version 4010, transaction set #151 has three different loops, which begin with segment PBI, named LoopPBI1, LoopPBI2, and LoopPBI3. Then there is more than one transaction set under a category, loop and block definitions for a specific transaction set are found under a category whose name begins with the functional group ID code, and is followed by the transaction set number. For example, the loops in transaction set #844 are subtypes of category CF844.

All bounded loop definitions for a transaction set appear under a group named Block. All bounded loops begin with an LS Segment, followed by the loop definition for the loop data segments and then an LE

## Interchange category

Double-click the X12 Inbound Interchange type to view the components of the interchange.

The following table lists and describes the components of the X12 Inbound Interchange.

Component	Description
ISA Segment	The first component of an interchange is the envelope header. The ISA segment identifies who sent the interchange or who you are sending one to. It also contains the objects that are used within the interchange for segment terminators, segment delimiters, and composite delimiters.
ISB Segment	Allows the sender to request more services from the service request handlers.
ISE Segment	Allows the sender to request more services from the service request handlers.

Component	Description
TA1 Segment	Optional and is typically used by network providers. TA1 is an interchange acknowledgment.
ISX Segment	Optional and is used to override the default release character and character encoding.
Funct'lGroup	Represents the functional groups that are sent or received.
IEA Segment	The interchange envelope trailer.

Note: The ISB and ISE segments are found only in ANSI versions 3050 and later. For example, the ISB and ISE segments are not shown as components of an interchange in the ansi3040 schema.

The X12 Outbound Interchange type is similar to the structure of the X12 Inbound Interchange type. The only difference is that Inbound is replaced with Outbound.

- [Restart attribute on functional group](#)  
The Funct'lGroup component of an interchange has the restart attribute.
- [Component rule on IEA segment](#)  
This documentation describes the component rule for the IEA segment.

## Restart attribute on functional group

The Funct'lGroup component of an interchange has the restart attribute.

If you receive functional groups from different departments within a trading partner, one department's good data will not be lost if another department's data is bad. You can remove this restart attribute if you would rather go directly to the next interchange if bad data occurs within any Inbound Funct'lGroup.

## Component rule on IEA segment

This documentation describes the component rule for the IEA segment.

The IEA Segment component of an interchange has the following component rule:

```
InterchangeCtrl# Element:$ =
InterchangeCtrl# Element:::Partner Inbound ISA Segment Control
ANSI & #InclFunct'lGroups Element:$-COUNTABS (Inbound Partner
Funct'lGroup ANSI)
```

The component rule states that the InterchangeCtrl# Element of the IEA Segment must be the same as the InterchangeCtrl# Element of the ISA Segment. If an Inbound Partner Funct'lGroup is invalid and a restart occurs, the component rule keeps different interchanges from being mixed up with one another.

## Transmission category

A transmission is made up of interchanges, so a Transmission type is made up of Interchange types.

An Inbound Transmission type is made up of Inbound Interchange types. For example, an X12 Inbound Transmission type is made up of X12 Inbound Interchange types and a Partner X12 Inbound Transmission type is made up of Partner X12 Inbound Interchange types.

The type Partner represents a generic trading partner. If you are trading with different partners who use different subsets of the X12 standards, you might want to rename the Partner type. You can then add new types to define the additional trading partners.

The components of Outbound Transmission types are similar to the components of the Inbound types.

- [Restart attribute on an interchange](#)  
The Interchange component of a Transmission type has the restart attribute.

## Restart attribute on an interchange

The Interchange component of a Transmission type has the restart attribute.

On input data, the restart attribute is intended to filter network noise and to detect invalid interchanges. If you receive an invalid interchange from one partner, the restart allows you to process good data in other interchanges in the same transmission.

When you receive network messages and you know where they appear in an inbound communication file, you might want to include them as a component of an Inbound Transmission type. If you do not include them as data, they are rejected as invalid interchanges if they are not recognized as part of a transmission. However, the processing time takes longer than if you included these network messages as data.

For EDI output data, restart is useful for auditing data. You can audit the output data to report when outbound interchanges within a transmission are invalid.

## Mapping ANSI data

When a map for mapping ANSI data is created, create an input or output card to represent the ANSI data.

When you specify the TypeName for the input card, you must specify a type under the Transmission category.

Select Inbound Transmission, or one of its subtypes, when your input is ANSI data. When you send ANSI data, select Outbound Transmission, or one of its subtypes.

## Input card - receiving ANSI data from multiple partners

When the input consists of only one interchange, select an Interchange type that corresponds to the correct Transmission type.

Select the Transmission type as shown in this table:

If you receive	Select this type
Multiple EDI standards	Inbound Transmission Note: Represents data from both ANSI and UCS
Only ANSI X12 data, and different trading partners in your schema are defined.	X12 Inbound Transmission
Only UCS data, and different trading partners in your schema are defined.	UCS Inbound Transmission
Only ANSI X12 data, and different trading partners in your schema are not defined.	Partner X12 Inbound Transmission
Only UCS data, and different trading partners in your schema are not defined.	Partner UCS Inbound Transmission

If you are familiar with Trading Partner PC, an Inbound Transmission (or one of its subtypes) defines the mail\_in\_new file. When you are mapping to EDI data, select an Outbound Transmission type for the output card. The type to choose depends on what you are sending.

## Output card - sending ANSI data to multiple partners

When the output consists of only one interchange, select an Interchange type that corresponds to the Transmission type.

This is shown in the following table:

If you are sending	Select this type
Multiple EDI standards.	Outbound Transmission Note: Represents data from both ANSI and UCS.
Only ANSI X12 data, and you define different trading partners in your schema	X12 Outbound Transmission
Only UCS data, and you define different trading partners in your schema	UCS Outbound Transmission
Only ANSI X12 data, and you have not defined different trading partners in your schema	Partner X12 Outbound Transmission
Only UCS data, and you have not defined different trading partners in your schema	Partner UCS Outbound Transmission

If you are familiar with Trading Partner PC, an Outbound Transmission type (or one of its subtypes) defines the mail\_out.new file.

## X12 examples

Example maps and schemas install with the Pack for Supply Chain EDI or within the Design Server imported project x12Examples.zip.

The X12 map and schema example files install in subdirectories under the following folder:

*install\_dir\supplychain\_edi\_vn.n.n\x12\examples*

The examples and example files are defined in the following table:

Example and associated files	Description
examples/ansi/asntoedi/ Example files: <ul style="list-style-type: none"><li>• asn856.txt - input record oriented data</li><li>• asn865.out - expected results X12 7040 856 transaction</li><li>• shipnotice - input schema</li><li>• sendasn - map source containing executable map sendasn</li></ul>	Creates an outbound EDI Advance Ship Notice (#856) with its nested hierarchical levels.
examples/ansi/editobol/ Example files: <ul style="list-style-type: none"><li>• Editobol.txt - input X12 7040 204 transaction</li><li>• bolout.txt - expected results record oriented data</li><li>• bol - input schema</li><li>• editobol - map source containing executable map bolading</li></ul>	Creates an outbound application record oriented file based on an inbound EDI Motor Carrier Shipment Information (#204).

Example and associated files	Description
examples/ansi/hlLoop/ Example files: <ul style="list-style-type: none"> <li>• asn856Test.input - input X12 7040 204 transaction</li> <li>• asn856Test.out - expected results record oriented data</li> <li>• ansi8020_856 - input schema</li> <li>• receiveasn - map source containing executable map receiveasn</li> </ul>	Creates an outbound application record oriented file based on an inbound EDI Advance Ship Notice (#856) with its nested hierarchical levels.
examples/ansi/sdq/ Example files: <ul style="list-style-type: none"> <li>• partner.txt - input partner file</li> <li>• wal850.new - input X12 850 (purchase orders).</li> <li>• Storeinfo.out - output temp file holding store information</li> <li>• wally.out - expected results of X12 810 invoices</li> <li>• profile - input schema for partner file</li> <li>• storeinfo - input schema for storeinfo file</li> <li>• sdq - map source containing executable map sqd</li> </ul>	Creates an outbound EDI invoice (#810) from an inbound EDI Order (#850). The turn-around mapping produces an interchange for each store number found in the inbound 850 SQD segment.
examples/ansi/xmlpo/ Example files: <ul style="list-style-type: none"> <li>• x12_850.in - input X12 7040 850 (purchase order)</li> <li>• orders.xml - input XML orders</li> <li>• xml_po_output.xml - expected output EDI to XML map</li> <li>• x12_po.out - expected output XML to EDI map</li> <li>• po_file - XML schema</li> <li>• xmlpo - map source that contains executable maps x12toxml and xmltox12</li> </ul>	Creates an outbound EDI 850 (purchase order) message, using XML input and an outbound XML Order using EDI 850 (purchase order) message input.

- [asntoedi example](#)

Instructions for running the asntoedi example.

- [editobol example](#)

Instructions for running the editobol example.

- [hlLoop example](#)

The hlLoop example is an example of using a customized ITX standard for the X12 856 Ship Notice/Manifest or Advanced Ship Notice - ASN. The ITX standard has been customized to preserve the hierarchy during runtime parsing of the input data when a hierarchy is defined for the output for easier mapping.

- [sqd example](#)

Instructions for running the sqd example.

- [xmlpo example](#)

Instructions for running the xmlpo example.

## asntoedi example

Instructions for running the asntoedi example.

Review the example notes prior to running the example.

- [asntoedi example notes](#)

Points to be aware of when using the asntoedi example.

- [How to run the asntoedi example](#)

Run instructions for the asntoedi example.

## asntoedi example notes

Points to be aware of when using the asntoedi example.

The asntoedi example output is generated from input that contains fixed formatted header and detail records.

The LAST index generates the correct HierarchicalID Element, which must increment by one for each HL Segment in the hierarchy being built, as shown here:

```
ST*856*1
BSN*00*00000084*940111*1427
HL*1**S <- HierarchicalID Element of this shipment = 1,
 no parent
```

```

TD1*BOX*1
HL*2*1*I <- HierarchicalID Element of this item = 2,
 parent shipment is 1

HL*3*1*I <- HierarchicalID Element of this item = 3,
 parent shipment is 1

HL*4*1*I <- HierarchicalID Element of this item = 4,
 parent shipment is 1

HL*5**S <- HierarchicalID Element of this shipment = 5,
 no parent

TD1*BOX*1
HL*6*5*I <- HierarchicalID Element of this item = 6,
 parent shipment is 5

HL*7**S <- HierarchicalID Element of this shipment = 7,
 no parent

TD1*BOX*1
HL*8*7*I <- HierarchicalID Element of this item = 8,
 parent shipment is 7

HL*9*7*I <- HierarchicalID Element of this item = 9,
 parent shipment is 7

HL*10*7*I <- HierarchicalID Element of this item = 10,
 parent shipment is 7

CTT*7
SE*17*1

```

## How to run the asntoedi example

Run instructions for the asntoedi example.

### About this task

Follow these steps:

### Procedure

1. Open the x12\_examples project.
2. Navigate to schemas and open the X12\_7040 subset schema.
3. Analyze the schema. The schema should analyze with no errors.
4. Open the shipnotice schema and analyze the schema. The schema should analyze with no errors.
5. Navigate to maps and open the sendasn map.
6. Build the map and then run the map.
7. View the output. Go to the map Diagram-Result, and hover the cursor over Target, then select View Data.
8. To download the file, go to Files and download the file, asn856.out.

## editobol example

Instructions for running the editobol example.

Review the example notes prior to running the example.

- [editobol example notes](#)  
Points to be aware of when using the editobol example
- [How to run the editobol example](#)  
Run instructions for the editobol example.

## editobol example notes

Points to be aware of when using the editobol example

- The input type of the executable map is Partner X12 Inbound Transmission. The output type is File, which is your file that contains any number of bills of lading. Each bill of lading is composed of one header record and many detail records. Each header record has several fields, including repeating sets of name and address information. These sets have been defined in the schema as a single object called NameSet, which is used three times. The detail record object contains five fields. These are fixed length records, where every field is mandatory and no delimiters are used. Header records are initiated by 01, and detail records by 02.
- The output file is made up of BillOfLading(s). The EachBOL functional map maps each #204 transaction set to the output bill of lading.

## How to run the editobol example

Run instructions for the editobol example.

## About this task

---

Follow these steps:

## Procedure

---

1. Open the x12\_examples project.
  2. Navigate to schemas and open the ansi7040 standard schema.
  3. Analyze the schema. The schema should analyze with no errors.
  4. Open the bol schema and analyze the schema. The schema should analyze with no errors.
  5. Navigate to maps and open the bolanding map.
  6. Build the map and then run the map.
  7. View the output. Go to the map Diagram-Result, and hover the cursor over Target, then select View Data.
  8. To download the file, go to Files and download the file,.bolout.txt.
- 

## hlLoop example

The hlLoop example is an example of using a customized ITX standard for the X12 856 Ship Notice/Manifest or Advanced Ship Notice - ASN. The ITX standard has been customized to preserve the hierarchy during runtime parsing of the input data when a hierarchy is defined for the output for easier mapping.

Map source and executable: receiveasn.mms

Customized standard: ansi8020\_856.mtt

Input X12 856: asn856Test.input

Output X12 856: asn856Test.out

The REF segments in the output file identify, the functional map executions in the map source/executable along with the description from the input file of the loop levels. For example: REF\*ZZ\*forEachItem\*Item 1 & Pack 2 & S1Order 1

identifies functional map forEachItem execution

Item 1 with Pack 2 parent

Pack 2 parent is Order1

Order 1 parent is S1 (Shipment 1).

- [\*\*Understanding Hierarchical Loops\*\*](#)

Hierarchical loops defined in EDI transactions or messages are considered the most complex structures in the EDI processing environment. Hierarchical loops define different levels of data, which can be used in any sequence, and skipped if the information is not needed. This allows EDI messages to align the loop to the data.

- [\*\*Hierarchical Design Rules\*\*](#)

- [\*\*Example of Record Oriented Data\*\*](#)

- [\*\*Example Diagram of a Hierarchy\*\*](#)

- [\*\*Customizing ITX Standard for Hierarchical Loop level\*\*](#)

The instructions below reference the X12 856 Advanced Ship Notice transactions but can be used for similar hierarchical structures.

- [\*\*Mapping HL levels to the output\*\*](#)

At the point where you want to begin the shipping level, example command to be used. This will pass the Shipment HL Loop and all HL child loops to the functional map forEachShipment().

---

## Understanding Hierarchical Loops

Hierarchical loops defined in EDI transactions or messages are considered the most complex structures in the EDI processing environment. Hierarchical loops define different levels of data, which can be used in any sequence, and skipped if the information is not needed. This allows EDI messages to align the loop to the data.

A Hierarchical loop is simply a special kind of nested loop where the nesting is not physically defined in the EDI transaction, but rather in the data content of the hierarchy. Successive occurrences of a hierarchical show a parent - child relationship.

The X12 856 transaction set is used to describe a physical shipment as well as information relating to the shipment, such as order information, product description, physical characteristics, type of packaging, marking, carrier information, and configuration of goods within the transportation equipment.

The X12 856 HL segment is used to define the start of a hierarchical looping structure. The HL segment must be created for each node of the application data hierarchy.

---

## Hierarchical Design Rules

1. Define the hierarchical paths.
  - a. Create a diagram of the hierarchy.
  - b. Number the nodes top to bottom and left to right starting with the left most node.
    - Primary path is top to bottom.
    - Secondary path is left to right.
2. Identify the base nodes.
  - Base node is a level without a parent.

- Multiple hierarchical paths may indicate multiple base nodes.
  - Identify the hierarchical levels.
3. Identify the levels.
- Shipment = S
  - Order = O
  - Packs = P
  - Items = I
4. Identify the relationships.
- What are the parent/child relationships?
  - Is the parent/child relationship defined in the data?
  - Is the application dependent on the hierarchy?
    - May need to enforce hierarchy.
  - Is the mapping for the child dependent on the parent?
    - May need to enforce hierarchy.

If the hierarchy does not need to be enforced, then the HL loop could be mapped as a flat (not nested) loop. The ITX standard can be used as defined.

If the hierarchy needs to be enforced (application data definition contains nesting), then the HL loop may need to be customized to preserve the hierarchy and allow mapping to preserve the hierarchy.

## Example of Record Oriented Data

```
H00 DIUTESTPTR 01003020VICS 940830170700000170900062921
H01 001313131313 9407141416 0001
SH011 S
SH02CTN25000072
OR012 1 O
OR02887378
OR03B C FOREWAY 9 DUNSNUMBR0897
OR04BYHLLOOP, INC. PA013 2 P
PA013 0000000600000000 000000000054000LB0000000000000000
IT014 3 I
IT02 UP022222640304
IT03 0000000006EA00000000000000000000006EA
SH011 S
SH02CTN25000072
```

## Example Diagram of a Hierarchy



## Customizing ITX Standard for Hierarchical Loop level

The instructions below reference the X12 856 Advanced Ship Notice transactions but can be used for similar hierarchical structures.

### Procedure

1. Make a copy of the type tree to be customized. You can also trim/prune the type tree to remove components not used. There is a documented process for accomplishing this task.
2. Open the copy of the standard type tree to be used.
3. Navigate to ANSI V8020>Set>Partner>Inbound>856 transaction and expand.
4. Copy LoopHL to each HL Loop level needed in the hierarchy.  
For example, copy LoopHL to LoopHLShip, then again for LoopHLOrder, and again for LoopHLPack, and LoopHLPackItem.
5. Open LoopHLPack and add LoopHLItem (set range to min 0, max s). Open LoopHLOrder and add LoopHLPack (set range the same on all). Open LoopHLShip and add LoopHLOrder.
6. Open the transaction and edit the component LoopHL as follows:
  - a. Modify to LoopHLShip. Or drag and drop Loop HLShip and set range to be the same as LoopHL.
  - b. Remove the LoopHL component.
7. Enter the following mapping rule on LoopHLShip:

```
HierarchicalLevelCd Element:HL Segment:$ = "S"
& (ABSENT(HierarchicalParentIDNum Element:HL Segment:$)
 | MEMBER(HierarchicalParentIDNum Element:HL Segment:$, {"0", " "}))
```

8. Open LoopHLShip and move to the bottom and enter the following mapping rule on LoopHLOder:

```
HierarchicalLevelCd Element:HL Segment:$ = "O"
& HierarchicalParentIDNum Element:HL Segment:$ = HierarchicalIDNum Element:HL Segment
```

9. Open LoopHLPack and LoopHLItem and enter the following mapping rule:

```
HierarchicalLevelCd Element:HL Segment:$ = "P"
& HierarchicalParentIDNum Element:HL Segment:$ = HierarchicalIDNum Element:HL Segment
HierarchicalLevelCd Element:HL Segment:$ = "I"
& HierarchicalParentIDNum Element:HL Segment:$ = HierarchicalIDNum Element:HL Segment
```

10. Save and analyze the type tree.

## Results

---

The input should parse into each HL Loop depending on the HL Loop level code:

```
(HierarchicalLevelCd Element:HL Segment:$) and its parent id (HierarchicalParentIDNum Element:HL Segment:$ = HierarchicalIDNum Element:HL Segment)
```

## Mapping HL levels to the output

---

At the point where you want to begin the shipping level, example command to be used. This will pass the Shipment HL Loop and all HL child loops to the functional map `forEachShipment()`.

```
=forEachShipment(LoopHLShip:Input)
```

And inside the `forEachShipment()` map you can map the next level for Order and so on.

## sqd example

---

Instructions for running the sqd example.

Review the example notes prior to running the example.

- [sqd example notes](#)  
Points to be aware of when using the sqd example.
- [How to run the sqd example](#)  
Run instructions for the sqd example.

## sqd example notes

---

Points to be aware of when using the sqd example.

This example shows the following:

- The use of multiple series functions in the same mapping rule.
- Mapping element, composite and terminator syntax items used in X12, rather than using defaults.

This example shows how to re-organize input X12 EDI data consisting of multiple purchase orders (#850 transaction sets) to create multiple output X12 invoices (#810 transaction sets).

The input data is organized into #850 transaction sets. Within each #850 is a series of line items (LoopPO1). Nested within each line item is a series of SDQ segments. Each SDQ segment can contain up to 10 store identification codes. So, line item #1 could be associated with hundreds of stores, line item #2 could be associated with some of the same stores, and some different stores. The output must be re-organized to send one invoice for each store identified in the input data. Each invoice must have only the line items associated with the store specified in the invoice.

## How to run the sqd example

---

Run instructions for the sqd example.

## About this task

---

Follow these steps:

## Procedure

---

1. Open the x12\_examples project.
2. Navigate to schemas and open the ansi7040 schema.
3. Analyze the schema. The schema should analyze with no errors.
4. Open the schema profile and analyze the schema. The schema should analyze with no errors.
5. Open the schema storeinfo and analyze the schema. The schema should analyze with no errors.
6. Navigate to maps and open the storeinfo map.
7. Build the map and then run the map.
8. View the output. Go to the map Diagram-Result, and hover the cursor over Target, then select View Data.

9. To download the file, go to Files and download the file, wally.out..

## xmlpo example

Instructions for running the xmlpo example.

Review the example notes prior to running the example.

- [xmlpo example notes](#)

Points to be aware of when using the xplo example.

- [How to run the xmlpo example](#)

Run instructions for the xmlpo example.

## xmlpo example notes

Points to be aware of when using the xplo example.

- The schema for the input card for the x12toxml map and for the output card for the xmltox12 map is the ansi7040 schema.
- A trimmed schema can be used for improved performance.
- The native format of the orders.xsd schema is used directly in the mapping.
- The x12toxml map will only process a single interchange. One approach to using it with a transmission containing multiple interchanges would be to use a pre-process that split each interchange in the transmission into a file of its own. A Message Broker flow could be used that calls a map to split the file and then invokes the x12toxml map to transform each of these interchange files into an XML format.

## How to run the xmlpo example

Run instructions for the xmlpo example.

## About this task

Follow these steps:

## Procedure

1. Open the x12\_examples project.
2. Navigate to schemas and open the ansi7040 schema.
3. Analyze the schema. The schema should analyze with no errors.
4. Navigate to the maps and open the xmlto12 map.
5. Build the map, then run the map.
6. View the output. Go to the map Diagram-Result, and hover the cursor over Target, then select View Data.
7. To download the file, go to Files and download the file,.x12\_po.out..
8. Repeat steps 6, 7, 8 and 9 of this procedure.

## EDIFACT schemas

The Pack for Supply Chain EDI contains EDIFACT standard schemas and example schemas.

EDIFACT schemas are used to validate and ultimately transform, data that conforms to the United Nation's (UN) EDIFACT standard, as described in the UN EDIFACT specifications.

- [About the EDIFACT standard](#)

The EDIFACT (Electronic Data Interchange For Administration, Commerce, and Transport) standard is primarily used in Europe.

- [Installation](#)

The EDIFACT schemas install with the Pack for Supply Chain EDI.

- [EDIFACT schema details](#)

Standard schemas and example schemas are contained in the Pack for Supply Chain EDI.

- [EDIFACT examples](#)

Example maps and schemas install with the Pack for Supply Chain EDI or within the Design Server imported project edifactExamples.zip and edifactExampleBlockChain.zip.

## About the EDIFACT standard

The EDIFACT (Electronic Data Interchange For Administration, Commerce, and Transport) standard is primarily used in Europe.

The standard is maintained by the UN and the specifications can be obtained from the United Nations Electronic Commission for Europe (UNECE) website:

[www.unece.org/trade/untdid/welcome.htm](http://www.unece.org/trade/untdid/welcome.htm)

This website contains links for the industry-specific, sub-standards.

ITX supports all of the modern **D** EDIFACT standard versions. Versions prior to that time are provided as parsing examples only. EDIFACT transitioned to using the **D** versions starting in the early 1990s. The **D** versioning system signifies that the release is a directory, updated twice a year, in a specific year.

#### Timeline of change:

- Before the Early 1990s  
EDIFACT versions were labeled with year identifiers such as **UN/EDIFACT 1988** or **UN/EDIFACT 1990**.
- Early 1990s onward  
EDIFACT started using the **D** versioning system. For example, **D.93A** refers to the directory released in the first half of 1993, while **D.93B** refers to the directory released in the second half of 1993.

#### Reason for the change:

The following are the reasons for the transition to the **D** versioning system:

- To standardize the release cycles  
The **D** versions provide a clear and systematic way to indicate the release cycle and year, making it easier to track updates and changes.
- To ensure consistency and predictability  
With biannual releases, users of EDIFACT standards can anticipate and prepare for updates more predictably, facilitating better planning and implementation.
- To improve the documentation and reference  
The **D** system simplifies the documentation and referencing, as each version is explicitly tied to a specific time period, reducing ambiguity.

#### How it works:

- D.XXA  
Indicates the first release of the year **XX**.

For example, **D.21A** refers to the first release of the year 2021.

- D.XXB  
Indicates the second release of the year **XX**.

For example, **D.21B** refers to the second release of the year 2021.

By using the **D** versioning system, EDIFACT ensures that the standards are regularly updated to keep pace with evolving business needs and technological advancements, while providing a clear and structured way to manage these updates.

---

## Installation

The EDIFACT schemas install with the Pack for Supply Chain EDI.

---

## EDIFACT schema details

Standard schemas and example schemas are contained in the Pack for Supply Chain EDI.

- [Overview](#)  
The root type of each EDIFACT schema is EDI.
- [EDIFACT category details](#)  
The EDIFACT category has four subtypes.
- [Interchange category](#)  
Double-click the Partner Inbound Interchange type to view the components of the interchange.
- [Transmission category](#)  
Transmission type is made up of Interchange types.
- [Mapping EDIFACT data](#)  
When creating a map for EDIFACT data, create an input or output card to represent the EDIFACT data.
- [Input card - EDIFACT data from multiple trading partners](#)  
Select an Interchange type that corresponds to a Transmission type.
- [Output card - sending EDIFACT data to multiple partners](#)  
When the output consists of only one interchange, select an Interchange type corresponding to the Transmission type

---

## Overview

The root type of each EDIFACT schema is EDI.

Each schema has EDIFACT, Interchange, and Transmission categories that stem from the EDI root type. For example, the edifd20a schema contains the EDIFACT version D, release 20A.

EDIFACT schemas are organized in the same way that ANSI X12 schemas are organized. If you trade by using both ANSI X12 and EDIFACT, you can easily design a schema to make it a multi-standard schema.

- [EDIFACT category](#)  
The EDIFACT category contains four subtypes.
  - [Interchange](#)  
The Interchange type contains the data types for an interchange.
  - [Transmission](#)  
The Transmission type contains the data types for a transmission. A data object is made up of one or more interchanges.
- 

## EDIFACT category

The EDIFACT category contains four subtypes.

The following are subtypes of the EDIFACT category:

- Control type defines the EDIFACT control structure objects. The Control category is the same on each EDIFACT version schema.
- Group type defines the functional groups for the version.
- Msg type defines the messages for the version.
- V category defines the message contents, segments, composites, and elements in that particular version.

The version-specific data objects can be found under the category whose name begins with a V followed by the version number. For example, VD20A is the name of the category in the edifd20a version schema.

For more information on these subtypes, see the ["EDIFACT Category"](#) section.

---

## Interchange

The Interchange type contains the data types for an interchange.

The Interchange category contains Inbound and Outbound partitioned group subtypes.

---

## Transmission

The Transmission type contains the data types for a transmission. A data object is made up of one or more interchanges.

The Transmission category contains Inbound and Outbound group subtypes.

Note: The EDIFACT schemas include both inbound and outbound types, where appropriate. In addition, types called Partner define a generic trading partner.

---

## EDIFACT category details

The EDIFACT category has four subtypes.

The subtypes are listed here: Control, Group, Msg, and a V (*version*) category.

- [Control subtree](#)  
The subtypes of the Control category are the same in each EDIFACT schema. These types are used in EDIFACT envelope segments.
  - [Group subtypes](#)  
The subtypes of the Group category represent functional groups in the given EDIFACT version.
  - [Msg subtype](#)  
The subtypes of the Msg category represent EDIFACT messages.
  - [Version specific subtype](#)  
The version specific category (for example, v93\_2) is a subtype of the EDIFACT category. The name of this category is determined by the EDIFACT version.
- 

## Control subtree

The subtypes of the Control category are the same in each EDIFACT schema. These types are used in EDIFACT envelope segments.

Expand the Control category to view the sub-schemas.

- [Envelope types](#)  
This documentation describes the ways in which envelope types are used.
- [UNA service string](#)  
The UNA type is an optional envelope control. It is used to specify syntax objects that appear within the EDIFACT interchange.
- [EDIFACT interchange envelope UNB and UNZ](#)  
An EDIFACT interchange begins with a UNB. It contains version release information, syntax information, and partner information. An interchange ends with a UNZ.
- [Functional group envelope UNG and UNE](#)  
A functional group begins with a UNG and ends with a UNE. Functional group envelopes in EDIFACT are optional.
- [Message envelopes - UNH and UNT](#)  
A message begins with a UNH and ends with a UNT.
- [Element category](#)

- **Delimiter category**  
The Delimiter types are syntax items.
- **Composite category**  
A composite is a group of related elements.

## Envelope types

This documentation describes the ways in which envelope types are used.

Envelope types are used for these reasons:

- To specify the syntax objects in the interchange that follows (the UNA).
- To surround messages or functional groups of an interchange (the UNB and UNZ).
- To surround a set of messages of the same type (the UNG and UNE).
- To surround the body of a message (the UNH and UNT).

Each Envelope type begins with a set of letters that identifies the segment, for example UNA. These same sets of letters (UNA) are also used in the Properties window of the Type Designer to define the value of the initiator.

## UNA service string

The UNA type is an optional envelope control. It is used to specify syntax objects that appear within the EDIFACT interchange.

The following are the components of the UNA type:

Component	Description
Composite Delimiter	A delimiter between components of a composite.
Element Delimiter	A delimiter between elements in an envelope or segment.
Decimal Delimiter	A separator between the integer and fractional part of EDIFACT decimal numbers.
ReleaseCharacter	A character used a release character.
Reserved Delimiter	Reserved for future use.
Terminator Delimiter	A terminator of an envelope or segment.

The Composite, Element, ReleaseCharacter and Terminator Delimiter types are all specified as syntax items. If a UNA appears in the data, the values that occur in the UNA set the active syntax values. If there is no UNA, the release character is assumed to be a question mark ? and the delimiter values are set to the values in the UNB.

## EDIFACT interchange envelope UNB and UNZ

An EDIFACT interchange begins with a UNB. It contains version release information, syntax information, and partner information. An interchange ends with a UNZ.

A UNB is defined as a prefix delimited type with the initiator UNB. Its delimiter is defined as Element Delimiter. The first component of the UNB type specifies the character set for text items, the Composite delimiter, and the version release.

Note: The default for the UNB delimiter is + and its default terminator is an apostrophe (') followed by a CR/LF. The default for the composite delimiter is set in the SyntaxID Composite type to a colon (:). You can change these defaults in the schemas, if necessary.

## Functional group envelope UNG and UNE

A functional group begins with a UNG and ends with a UNE. Functional group envelopes in EDIFACT are optional.

When functional group envelopes are used, the Functional Group Ref# Elements on both the UNG and the UNE must match. It is common not to use them because the version information is also in the message envelope.

## Message envelopes - UNH and UNT

A message begins with a UNH and ends with a UNT.

## Element category

The item subtypes of the **Element** category are used as components of **Envelope** types. For example, trading partner identification items are defined under the **Element** category.

## Delimiter category

The Delimiter types are syntax items.

They are used as composite delimiters, decimal separators, element delimiters, release characters, and terminators in the EDIFACT segments.

The values of these syntax objects can be different in different interchanges. Their values are determined in the UNA; each syntax object is a component of the UNA type. However, if the UNA is not present, the syntax values are found in the UNB. For example, the UNB type is defined as delimited, and the value of its delimiter is specified as the syntax item Element Delimiter. The value of the Find property is Yes. This value indicates that, in input data, the value in the UNB is found and used as the value for the delimiter for the rest of the segments in the interchange.

Double-click a subtype of Delimiter (for example, Element) to view its restrictions. The restrictions are the values that you can use for that syntax object.

## Composite category

A composite is a group of related elements.

The Composite group subtypes are used as components of Envelope types. A composite has its own delimiter, which is typically a colon :.

## Group subtypes

The subtypes of the Group category represent functional groups in the given EDIFACT version.

The type name of a functional group is F followed by the EDIFACT version. For example, the functional group type in the edifd20a schema is FD20A.

A functional group is made up of messages. A functional group begins with a UNG and ends with a UNE, which are optional.

## Msg subtype

The subtypes of the Msg category represent EDIFACT messages.

The sub-types of Fversion (for example, FD20A) are all the messages in the EDIFACT version. The name of each message is the value of the MsgType Element in the UNH.

A message begins with a UNH and ends with a UNT.

The middle component is the type that represents the actual contents of the message. These types are found under the version-specific category, for example, the VD20A category in the edid20a schemas.

Each message has a unique value for the MsgType Element. The value is specified by the component rule on the UNH component. For example, the component rule states that the MsgType Element of the UNH in the BAPLTE message is BAPLTE. Each message has a similar component rule.

## Version specific subtype

The version specific category (for example, v93\_2) is a subtype of the EDIFACT category. The name of this category is determined by the EDIFACT version.

- [Composite category](#)  
A composite is a set of related data elements.
- [Viewing the Element category's full description](#)  
The item subtypes of the Element category are the EDIFACT data elements.
- [Segment category](#)  
A segment is a logical unit of information, like a data record. The Segment type is partitioned by initiator value. Each segment begins with a unique initiator, which identifies the segment. For example, the BUS segment begins with the initiator BUS.
- [Set category](#)
- [Groups of segments](#)  
In EDIFACT, a repeating group of segments is called a group.

## Composite category

A composite is a set of related data elements.

Each composite is defined as infix delimited and its delimiter is the syntax item Composite Delimiter. For an interchange, the value of the Composite Delimiter is set in the UNA or the UNB. Typically, the composite delimiter is a colon :.

## Viewing the Element category's full description

The item subtypes of the Element category are the EDIFACT data elements.

## About this task

To see the full description of an element, view the type properties in the Map Designer.

To view an element's full description, perform these steps:

## Procedure

---

1. Select the type in the schema.
2. Right-click and select Properties from the menu. Or, click the Properties icon on the tool bar.  
The description is in the Value column of the Description property.

## Results

---

There are many element types in an EDIFACT schema. Although they are arranged alphabetically, you might want to use the **Find** command on the Edit menu to locate a particular element.

The following list summarizes how an EDIFACT schema corresponds to the EDIFACT documentation of the EDI standards:

- Element codes are specified as restrictions in an EDIFACT schema.
- Different elements with the same EDIFACT description are named by appending the reference number designator to the end of the subtype name.
- Some elements have subtypes of their own. These elements are not found as data elements in the EDIFACT documentation. Element subtypes were added if the same element was used more than once as a component of the same Segment. For example, if a Segment has two Date Element components, one might be an Arrival Date Element and the other a Departure Date Element.

---

## Segment category

A segment is a logical unit of information, like a data record. The Segment type is partitioned by initiator value. Each segment begins with a unique initiator, which identifies the segment. For example, the BUS segment begins with the initiator BUS.

Because an Element Delimiter must follow the segment initiator value and precede the first data element, each segment is defined as prefix delimited by the Element Delimiter, typically the plus sign + and has a terminator of Terminator Delimiter, typically the apostrophe ('). In addition, the release character is defined as Release Character Delimiter, typically the question mark (?).

Components of a segment can be Element types and Composite types. The segments were defined to comply with the rules specified in the EDIFACT documentation.

Note: For readability, the default Segment terminator is defined as an apostrophe ('), followed by a CR/LF. You can change the terminator as needed.

---

## Set category

Types are organized as subtypes of the Partner category: Inbound and Outbound. The subtypes of the Inbound and Outbound categories are identical.

Types defining message contents are organized under categories. Each type that defines the contents of a certain message appears under a certain category. The name of the category is the message name, which matches the type name under the Msg category. For example, the type that defines the contents of the BAPLTE message is found under the BAPLTE category.

The name of the type that defines the contents of a message is Message.

---

## Groups of segments

In EDIFACT, a repeating group of segments is called a group.

A BAPLTE group type can also contain other group types. The word Group is typically appended with a number, such as 1 (for Group1), to distinguish it from other groups. For example, the BAPLTE message type contains Group1, Group2, and Group3.

---

## Interchange category

Double-click the Partner Inbound Interchange type to view the components of the interchange.

The first component of the interchange, UNA, is sometimes called a "service string" in EDIFACT. It is optional. It specifies the syntax objects, such as delimiters, that appear in the rest of the interchange.

The second and fourth components collectively define the interchange envelope. The UNB is the envelope header. It specifies the trading partner information and syntactical information when the UNA is missing. The UNZ is the envelope trailer.

The third component defines the functional groups within the interchange. In EDIFACT data, the functional group envelopes are not required. Most often, EDIFACT data contains only a message envelope, which is inside the Inbound Partner Group.

The layout of components in the outbound interchange is similar to the layout in the inbound.

- [Restart attribute on functional group](#)

The Group component of an interchange has the restart attribute.

- [Component rule on UNZ segment](#)

The UNZ segment has a component rule that assures that the group of one interchange is not mixed up with the group of another interchange if a restart occurs.

## Restart attribute on functional group

The Group component of an interchange has the restart attribute.

When you receive different functional groups from different departments within a trading partner, one department's good data is not lost if another department's data is bad. You can remove this restart attribute if you would rather go directly to the next interchange if bad data occurs within any Inbound Group.

## Component rule on UNZ segment

The UNZ segment has a component rule that assures that the group of one interchange is not mixed up with the group of another interchange if a restart occurs.

The component rule is shown here:

```
InterchangeControlRef Element:$ = InterchangeControlRef Element:Partner Inbound UNB
Envelope Control EDIFACT Data
```

The rule states that the InterchangeControlRef Element of the UNZ must be equal to the InterchangeControlRef Element of the UNB. If an Inbound Partner Group is invalid and a restart occurs, the component rule keeps different interchanges from getting mixed up with one another.

## Transmission category

Transmission type is made up of Interchange types.

An Inbound Transmission type is made up of Inbound Interchange types. For example, an EDIFACT Inbound Transmission type is made up of EDIFACT Inbound Interchange types and a Partner EDIFACT Inbound Transmission type is made up of Partner EDIFACT Inbound Interchange types.

The Partner type represents a generic trading partner. If you are trading with different partners who use different subsets of the EDIFACT standards, you might want to rename the Partner type and add then add new types to define the additional trading partners.

The components of Outbound Transmission types are similar to those of Inbound.

The following group windows display the components of the Inbound and Outbound subtypes of Transmission. The Partner Inbound Transmission EDI group window illustrates that it is made up of Partner Inbound Interchange types and that the Partner Outbound Transmission EDI group window illustrates that it is made up of Partner Outbound Interchange types.

- [Restart attribute on an interchange](#)

The Interchange component of a Transmission type has the restart attribute. On input data, the restart attribute is intended to filter network noise and to detect invalid interchanges.

## Restart attribute on an interchange

The Interchange component of a Transmission type has the restart attribute. On input data, the restart attribute is intended to filter network noise and to detect invalid interchanges.

When you receive an invalid interchange from one partner, the restart attribute allows processing of valid data in other interchanges in the same transmission.

When you receive network messages and you know where they appear in an inbound communication file, you may want to include them as a component of an Inbound Transmission type. If you do not include them as data, they are not recognized as being part of a transmission and are rejected as invalid interchanges. However, the processing time is longer than if you included these network messages as data.

For EDI output data, the restart attribute is useful for auditing data. You can audit the output data to report when outbound interchanges within a transmission are invalid.

The following schema, EdifISO9735-4 conforms to ISO 9735 part 4.

## Mapping EDIFACT data

When creating a map for EDIFACT data, create an input or output card to represent the EDIFACT data.

When specifying the type of the card, select a subtype from the Transmission category.

When your input is EDIFACT data, select Inbound Transmission, or one of its subtypes. When sending EDIFACT data, select Outbound Transmission, or one of its subtypes.

## Input card - EDIFACT data from multiple trading partners

Select an Interchange type that corresponds to a Transmission type.

When the input consists of only one interchange, select an Interchange type that corresponds to the Transmission type, as shown in this table.

When You Receive	Select This Type
Multiple EDI standards	Inbound Transmission
Only EDIFACT data, and you have not defined different trading partners in your schema	Partner Inbound Transmission

When you map to EDI data, select an Outbound Transmission type for the output card. The type to choose depends on what is being sent.

## Output card - sending EDIFACT data to multiple partners

When the output consists of only one interchange, select an Interchange type corresponding to the Transmission type

The Interchange control type that corresponds to the Transmission type is shown in this table:

If You Are Sending	Select This Type
Multiple EDI standards	Outbound Transmission
Only EDIFACT data, and you have not defined different trading partners in your schema	Partner Outbound Transmission

## EDIFACT examples

Example maps and schemas install with the Pack for Supply Chain EDI or within the Design Server imported project edifactExamples.zip and edifactExampleBlockChain.zip.

The example map and schema example files install in subdirectories under the following folder:

*install\_dir\supplychain\_edi\_vn.n.n\edifact\examples*

The examples and example files are defined in the following table:

Example and associated files	Description
examples/arrival Example files: <ul style="list-style-type: none"><li>• arrival - map source containing arrival_notice executable map</li><li>• simonin - input schema</li><li>• child.txt</li><li>• master.txt</li><li>• arrival.out - expected output</li></ul>	Creates an outbound EDI Arrival notice message (IFTMAN), used for the exchange of cargo and transport related information. A lookup is performed in the master.txt input file using the track number field located in the child.txt input file.
examples/edf2xml Example files: <ul style="list-style-type: none"><li>• ordersedi_in.txt - input EDIFACT D17A purchase orders</li><li>• orders_in.xml - input XML purchase orders</li><li>• orders_out.xml - results EDI to XML executable map</li><li>• edifact_out.txt - results XML to EDI executable map</li><li>• orders - XML schema that defines the XML purchase order</li><li>• xmlpo - map source containing executable maps edifacttoxml and xmltoedifact</li></ul>	Creates an outbound EDI ORDER message, using XML input and an outbound XML Order using EDI ORDER message input.
examples/ediBlockChain Example files: <ul style="list-style-type: none"><li>• Multiple maps, schemas, files</li></ul>	Example of using traditional EDI with blockchain.

Example and associated files	Description
examples/packages Example files: <ul style="list-style-type: none"> <li>• Input_binPackage.txt - input binary data to attach</li> <li>• Input_ediPackage.txt - input edifact message to attach</li> <li>• Input_xmlPackage.xml - input non-binary data to attach</li> <li>• Output_edipkgAttach.txt - output from map edipkgAttach</li> <li>• Input_edipkgDetach1.txt - input for map edipkgDetach1</li> <li>• Input_edipkgDetach2.txt - input for map edipkgDetach2</li> <li>• packageTests - map source containing executable maps edipkgAttach, edipkgDetach1, edipkgDetach2</li> </ul>	Example using EDIFACT service segment definitions for packaging and security taking three different inputs to create EDIFACT packages.
examples/security Example files: <ul style="list-style-type: none"> <li>• Input_binPackage.txt - input binary data to attach</li> <li>• Input_ediPackage.txt - input edifact message to attach</li> <li>• Input_xmlPackage.xml - input non-binary data to attach</li> <li>• Output_edisecurity.txt - output from map edisecOut</li> <li>• Output_security.txt - output for map edisecIn</li> <li>• securityTests - map source containing executable maps edisecIn, edisecOut</li> </ul>	Example using EDIFACT service segment definitions for packaging and security taking three different inputs to create EDIFACT packages with security.
examples/vermas Example files: <ul style="list-style-type: none"> <li>• vermas_input.json - input JSON file representing vermas data</li> <li>• vermas_d17a.txt - results JSON to vermas executable map</li> <li>• vermas_template - JSON schema template for vermas message</li> <li>• jsonvermas - map source containing executable map json2vermas</li> </ul>	Creates an outbound EDI Verified gross mass message (VERMAS) message, which provides VGM information on transport equipment such as a seagoing container. The executable map issuing JSON as input.

- [arrival example](#)

Instructions for running the arrival example.

- [edf2xml example](#)

Instructions for running the edf2xml example.

- [ediblockchain example](#)

The ediblockchain example is an example of using traditional EDI with blockchain. The example is using EDIFACT EDI documents, but the block chain concept is not limited to EDIFACT or EDI documents.

- [Packages example](#)

The edfiso9735-41 type tree contains service segment definitions for packaging and security along with any updates to the CONTRL message.

- [Security example](#)

The edfiso9735-41 type tree contains service segment definitions for packaging and security along with any updates to the CONTRL message.

- [vermas example](#)

Instructions for running the vermas example.

## arrival example

Instructions for running the arrival example.

Review the example notes before you run the example.

- [arrival example notes](#)

Points to be aware of when using the arrival example.

- [How to run the arrival example](#)

Run instructions for the arrival example.

## arrival example notes

Points to be aware of when using the arrival example.

- In this map there are two input files which make up arrival notices. The master file contains headers. The child file contains sets of details, which correspond to the headers. These files are defined in the simonin schema.
- The output file is an EDIFACT EDI Interchange. It is defined in the edifd17a schema.
- In the map rule for the output IFTMAN, you will see the functional map toIFTMAN. This map generates one IFTMAN per functional map oIFTMAN. This map generates one IFTMAN per group of child records, and its corresponding master record.
- Within the map IFTMAN, the ChildToLoopGID functional map is referenced.
- Expected output can be found in the file, arrival.out.

---

## How to run the arrival example

Run instructions for the arrival example.

### About this task

Follow these instructions:

### Procedure

1. Open the edifact\_examples project.
2. Navigate to schemas and open the edfd17a standard schema.
3. Analyze the schema. The schema should analyze with no errors.
4. Open the simonin schema and analyze the schema. The schema should analyze with no errors.
5. Navigate to maps and open the arrival\_notice map.
6. Build the map and then run the map.
7. View the output. Go to the map Diagram-Result, and hover the cursor over Target, then select View Data.
8. To download the file, go to Files and download the file, arrival.out.

---

## edf2xml example

Instructions for running the edf2xml example.

Review the example notes before you run the example.

- [edf2xml example notes](#)  
Points to be aware of when using the edf2xml example.
- [How to run the edf2xml example](#)  
Instructions for running the edf2xml example.

---

## edf2xml example notes

Points to be aware of when using the edf2xml example.

- The schema for the input card for the edifacttoxml map and for the output card for the xmltoedifact map is the edifd17a schema. A trimmed schema can be used for improved performance.
- The native format of the orders.xsd schema is used directly in the mapping.
- The edifacttoxml map will only process a single interchange. One approach to using it with a transmission containing multiple interchanges would be to use a pre-process that split each interchange in the transmission into a file of its own. A Message Broker flow could be used that calls a map to split the file and then invokes the edifacttoxml map to transform each of these interchange files into an XML format.

---

## How to run the edf2xml example

Instructions for running the edf2xml example.

### About this task

Follow these instructions:

### Procedure

1. Open the edifact\_examples project.
2. Navigate to schemas and open the edfd17a standard schema.
3. Analyze the schema. The schema should analyze with no errors.
4. Navigate to maps and open the xmltoedifact map.
5. Build the map and then run the map.
6. View the output. Go to the map Diagram-Result, and hover the cursor over Target, then select View Data.
7. To download the file, go to Files and download the file, edifact\_out.txt.
8. Repeat steps 6, 7, 8 and 9 of this procedure.

## ediblockchain example

The ediblockchain example is an example of using traditional EDI with blockchain. The example is using EDIFACT EDI documents, but the block chain concept is not limited to EDIFACT or EDI documents.

With traditional EDI, participants are using multiple forms of data with their business processes for example:

- Legacy in-house applications
- ERP systems like SAP
- Public EDI standards like EDIFACT, X12, RAIL, HIPAA
- XML and JSON data
- Spreadsheets, email, and fax

Participants do business with their partners and handle these different forms of data multiple ways using transformation tools (ITX), vans, peer-to-peer connections, and portals.

They have some very complex maintenance and configuration with all these forms of data and they have tried to keep audit trails and business partner relationships along with maintenance of all this configuration and routing using as few tools as possible. Some Participants have moved some of the complex processing to a Service and so their transformations across all the forms of data is in one location.

This traditional EDI is 30+ years old and with these business processes, there are still gaps. It is difficult to share data with non-edi participants. Some participants are not known or not trusted. They want to expand their business relationships, but the configuration can be a complicated task. Some participants have manual data entry to handle email, fax and non-program readable data prone to errors. The audit trail is compromised or incomplete with some business processes required to keep the information for a designated period.

- **Understanding the ediBlockChain example**

The example identifies three EDI participants and three non-EDI participants. Each participant adds blocks to the order change by issuing requests to a block chain service. Execution begins with the Buyer having a json order which initiates EDI processes with the supplier. Next the supplier initiates EDI processes with the carrier and buyer. Last, the carrier books trucks for delivery to the buyer.

- **The structure of the example:**

- **What the ediblockchain folder contains**

The ediblockchain folder contains a script to compile all the maps in the example, compileall.bat

- **Quick execution steps:**

## Understanding the ediBlockChain example

The example identifies three EDI participants and three non-EDI participants. Each participant adds blocks to the order change by issuing requests to a block chain service. Execution begins with the Buyer having a json order which initiates EDI processes with the supplier. Next the supplier initiates EDI processes with the carrier and buyer. Last, the carrier books trucks for delivery to the buyer.

Each participant folder in the example contains maps for execution with a top-level map beginning with "a". The top-level maps contain the specific calls to the block chain service and shows document flows and execution order on the last output card with the name (mapControl).

The buyer creates an EDI ORDER, creates block chain order, adds participants for access to the chain and edi reference information as a block in the chain. The EDI ORDER contains block chain information in the RFF segment:

RFF+CBB:orderChain^12857654123450000134012345500004'

The supplier receives the EDI ORDER, creates an EDI ORDRESP, supplies the items in the order, initiates shipping via EDI IFTMBF (Carrier Booking), adds participants for access to the chain and edi reference information as a block in the chain. The EDI ORDRESP and EDI IFTMBF contains block chain information in the RFF segment:

RFF+CBB:orderChain^12857654123450000134012345500004'

The carrier receives the EDI IFTMBF, creates EDI IFTMBC (Carrier Booking Confirmation), adds participants for access to the chain and edi reference information as a block in the chain. The carrier adds non-EDI participants as Trucks and each truck will add temperature readings on the shipped items to the block chain.

Each participant will use a predefined control block interface and document definitions and call the block chain service map. The control block identifies the type of call being made and contains information for the block chain service map to process the request and add a block to the chain.

## The structure of the example:

```
ediblockchain
... blockchain
..... config
..... journal
..... maps
..... templates
... buyer
... carrier
```

... edifiles  
... supplier

---

## What the ediblockchain folder contains

The ediblockchain folder contains a script to compile all the maps in the example, compileall.bat

The blockchain folder contains the config folder, journal, maps, and templates. The config folder contains a configuration file which may be expanded in the future. The journal folder contains the json templates for the block chain information file and actual block chain file.

The journal folder also contains some control block definitions for the block chain service maps. The maps folder contains the map source for the block chain requests and processing. The templates folder contains some json document and control block definitions for the participants to use to make requests and pass documents to the block chain service. Viewing the templates will help understand the mapping and block chain service requests.

The edifiles folder contains the EDIFACT standard being used and will contain all the EDI documents communicated between each participant.

The buyer, carrier, and supplier folders identify an EDI participant and contain maps and data to create edi documents and make block chain requests. The map source in each participant folder contains a top-level map (aBuyerNewOrder, aSupplierNewOrder, aCarrierNewOrder). The last output card controls the block chain service requests, and triggers for the next participant.

The buyer folder is the main execution folder. The map source file is BuyerOrderMaps.mms and the top-level map aBuyerNewOrder will start the new order and trigger the other participants processing.

---

## Quick execution steps:

### Procedure

1. Compile all the maps in the example.
  - Using Design Studio, execute the script in folder .\ediblockchain\compileall.bat.
  - Using Design Server, the ediblockchain example is in the project edifactExampleBlockChain.zip. After importing, create a package for the project and build the package.
2. Execute the folder .\ediblockchain\buyer.
  - Using Design Studio, open the BuyerOrderMaps.mms and run the top level map aBuyerNewOrder
  - Using Design Server, open the aBuyerNewOrder map and run.

This executes all the block chain participants.

### Results

Execution flows from the buyer with a purchase order, to the supplier who communicates with the purchaser and the carrier for shipping, and the carrier who identifies the trucks used for delivery. Each block contains information relevant to the partner or participant.

Using Design Studio, folder .\ediblockchain\blockchain\journal contains example results:

- orderChain.dat contains the chain data.
- orderChainInfo.dat is the partner information and keys to chain data.

Using Design Server:

- Execute map aViewOrderChain and view the output to see the chain data.
- Execute map aViewOrderChainInfo and view the output to see the partner information and keys to the chain data.

## Packages example

The edfiso9735-41 type tree contains service segment definitions for packaging and security along with any updates to the CONTRL message.

The EDIFACT documentation identifies the UNO, Object, UNP as a package. The UNO/UNP should follow EDIFACT syntax with a segment terminator following the segment information. The documentation states that the data presented as an object shall not be governed by EDI syntax rules. This means the object cannot be validated as a typical EDI message. Although it is possible the object is encrypted data and when decrypted could be validated using the usual EDI syntax and the Supply Chain EDI compliance checking.

The documentation does not identify if the Object attached in the package should be terminated with a segment terminator or any other terminator.

Terminating the Object attachment would allow the UNO/UNP segments to be parsed and processed in the TX map using the edfiso9735-41 type tree and allow the Object to be detached without the UNO/UNP service segments. The edfiso9735-41 type tree defines the OBJECT group to allow this processing for both inbound and outbound for the Package Category.

If the Object is not terminated with a segment terminator, the UNO/UNP becomes part of the Object to be detached and must be processed outside the TX map used to detach the package which requires a user defined type tree and separate map. The edfiso9735-41 type tree defines the PACKAGE group to allow this processing for both inbound and outbound for the Package Category.

The input definition used for the example packages is defined as UTF-8 to ensure binary data is processed preserving the original data.

The output definition for the EDIFACT package object is defined as UTF-8.

- [What the map source of packages example contains](#)

## What the map source of packages example contains

The map Source packagetests.mms contains three executable maps:

- edipkgAttach - An example of taking three different inputs to create EDIFACT packages.
  - Three inputs:
    - Input\_xmlPackage.xml: Input 1 is xml
    - Input\_ediPackage.txt: Input 2 is EDIFACT EDI
    - Input\_binPackage.txt: Input 3 is binary
  - Output\_edipkgAttach.txt: Output is EDIFACT EDI containing packages.
- edipkgDetach1 - An example using one EDIFACT input containing multiple edi and xml packages to create raw data output. The output attempts to identify the UNO/Object/UNP parsing and mapping with and without a segment terminator for the Object.
  - Input\_edipkgDetach1.txt: Input 1 is EDIFACT EDI and multiple packages
  - Output\_edipkgDetach1.txt: Output is EDIFACT EDI containing packages
- edipkgDetach2 - An example using one EDIFACT input containing multiple edi, xml, binary packages to create raw data output. The input file was created with map execution of the edipkgAttach map and renamed. The output attempts to identify the UNO/Object/UNP parsing and mapping with and without a segment terminator for the Object.
  - Input\_edipkgDetach2.txt: Input 1 is EDIFACT EDI and multiple packages
  - Output\_edipkgDetach2.txt: Output is EDIFACT EDI containing packages

## Security example

The edfiso9735-41 type tree contains service segment definitions for packaging and security along with any updates to the CONTRL message.

The security segments provide EDIFACT structures with the security service of confidentiality for example encryption, compression, filter, digital signatures, etc. The message body or package is not governed by EDI message syntax rules and cannot be validated as a typical EDI message. Although it is possible to detach and decrypt the data which could be validated using the usual EDI syntax and the Supply Chain EDI compliance checking.

The input definition used for the example security also uses packages and is defined as UTF-8 to ensure binary data is processed preserving the original data.

The output definition for the EDIFACT package object is defined as UTF-8.

- [What the map source of security example contains](#)

## What the map source of security example contains

Map Source of securitytests.mms contains two executable maps:

- edisecOut - An example of taking three different inputs to create EDIFACT packages with security segments.
  - Three inputs:
    - Input\_xmlPackage.xml: Input 1 is xml
    - Input\_ediPackage.txt: Input 2 is EDIFACT EDI
    - Input\_binPackage.txt: Input 3 is binary
  - Output\_edisecurity.txt: Output is EDIFACT EDI containing packages and security segments.
- edisecIn - An example using one EDIFACT input containing multiple edi and xml packages with security segments to create raw data output. The output attempts to identify the UNO/Object/UNP parsing and mapping with and without a segment terminator for the Object.
  - Input\_ediSecurity.txt: Input 1 is EDIFACT EDI and multiple packages and security segments.
  - Output\_security.txt: Output is EDIFACT EDI containing packages.

## vermas example

Instructions for running the vermas example.

- [vermas example notes](#)  
Points to be aware of when using the vermas example.
- [How to run the vermas example](#)  
Run instructions for the vermas example.

## vermas example notes

Points to be aware of when using the vermas example.

- The vermas example creates an outbound EDI Verified gross mass message (VERMAS) message.
- The VERMAS message provides VGM information on transport equipment such as a seagoing container.
- The executable map uses a JSON document as input.
- The expected results can be found in vermas\_d17a.txt file.

# How to run the vermas example

Run instructions for the vermas example.

## About this task

Follow these instructions:

## Procedure

1. Open the edifact\_examples project.
2. Navigate to schemas and open the edfd17a standard schema.
3. Analyze the schema. The schema should analyze with no errors.
4. Navigate to maps and open the json2vermas map.
5. Build the map and then run the map.
6. View the output. Go to the map Diagram-Result, and hover the cursor over Target, then select View Data.
7. To download the file, go to Files and download the file, vermas\_d17a.txt.

## Compliance check Overview

The EDI Standard organizations publish formats for X12, RAIL, IATA, and EDIFACT documents so that they can be correctly translated by computers. Strict compliance with these standards is essential if EDI files are to be processed properly. Supply Chain EDI Compliance Check framework enforces these EDI standards by validating X12, RAIL, IATA, and EDIFACT EDI documents before translation occurs for your applications. Validation of newly created EDI documents ensures that the data you create conforms and does not create issues for your trading partners. As the trend in the industry is to consider fines for non-compliance, verifying that the data you create is compliant, and is of key importance.

Validation ensures more accurate EDI translation and a more robust automated process by detecting and rejecting EDI files with anomalies that might break the application and translation programs and interrupt production.

If there are errors present in the EDI data, the EDI Compliance Check framework produces acknowledgments that are ready for re-enveloping and transmittal back to the sending trading partner.

## Validation

Validation is performed at all levels of the EDIFACT, IATA, X12, or RAIL EDI message: the interchange, group and transaction envelopes, each segment, and each of the element components in each transaction.

The main maps for the EDI Compliance Checker are named ccx12 for X12, x12u for X12U, ccttx12 for X12 travel & transportation subset, ccrail for RAIL, ccapi for IATA-API, ccpnr for IATA-PNR, ccttedf for EDIFACT travel & transportation subset, and ccedf for EDIFACT. Each EDI Compliance Check map uses a framework that processes data through these steps.

- Collects and stores metadata information about the configuration and input data.
- Validates the Interchange envelope and optionally produces a TA1 acknowledgment for X12 and RAIL data.
- Validates the group and transaction data; collects error location and type information.
- Produces the acknowledgment for the interchange. If errors are found, an error log is produced for each transaction, group, or interchange error.
- Produces a JSON report based on interchange and group acknowledgment information.
- [X12 Validation considerations](#)  
The X12 EDI Compliance Check framework performs optional extended processing to be considered.
- [Validation overview](#)  
The EDI Compliance Check framework performs validation as follows:
- [Validation checklist](#)  
Validation involves ensuring the EDI data is syntactically correct as well as the message structure.

## X12 Validation considerations

The X12 EDI Compliance Check framework performs optional extended processing to be considered.

The Interchange Syntax Extension or ISX control segment was introduced with X12 version 7040 with release character delimiter and character encoding. X12 version 8010 was expanded to include override X12 version and Industry Identifier.

X12 members are advised to have trading partner agreements in place before using the ISX segment. When the data stream includes the ISX control segment, X12 compliance checking MUST be configured. At least one ISXcontrol setting must be present to continue compliance processing. These settings can be found in the x12cfg.xsd and XML configuration files for ccx12 execution or JSON configuration file for x12u execution.

```
<ISXcontrol>
 <IgnoreISX>F</IgnoreISX>
```

Any ISX control segment containing a character encoding value must use the new component x12u for compliance checking, or use ccx12 component with the XML configuration setting to `<IgnoreISX>T</IgnoreISX>`.

X12 validation processing for the release character must use ISA version 7040 or higher. Processing for the character encoding and code list override must use ISA version 8010 or higher. The ISX segment must be present in the input data stream.

The ISX control configuration settings identify processing control for the ISX segment.

```
<ISXcontrol>
 <IgnoreISX>F</IgnoreISX>
 <ReleaseCharacter>
 <DefaultReleaseCharacter></DefaultReleaseCharacter>
 <OverrideReleaseCharacter></OverrideReleaseCharacter>
 </ReleaseCharacter>
 <CharacterEncoding>
 <AllowEncoding>F</AllowEncoding>
 <DefaultSourceEncoding></DefaultSourceEncoding>
 <OverrideSourceEncoding></OverrideSourceEncoding>
 <DefaultTargetEncoding></DefaultTargetEncoding>
 <OverrideTargetEncoding></OverrideTargetEncoding>
 </CharacterEncoding>
 <CodeListVersionOverride>
 <AllowCodeListOverride>F</AllowCodeListOverride>
 <DefaultCodeListVersion></DefaultCodeListVersion>
 <OverrideCodeListVersion></OverrideCodeListVersion>
 </CodeListVersionOverride>
</ISXcontrol>
```

Default and override values for the release character, character encoding, and code list version override can be provided. Default values for Character Encoding (ISX02 element) and Code List Version Override (ISX03 element) can be used when these ISX elements do not contain the value. The exception being the Release Character Default, which will modify the value found in the ISX01 element. Override values can be used to override the values in the ISX. The values in the ISX will be used when no default or override values are configured.

The standards in the X12U component are defined with the UTF-8 universal character set and sized as character vs bytes. The standards provide validation and application mapping to preserve data values. For application maps it is recommended the target encoding configuration use UTF-8.

X12 standard defines valid values for the ISX character encoding element which can be expanded in future releases. Values that represent character sets can be implemented using the X12U component. For testing purposes, UTF-8 and UTF-16 TX codes have been added for X12 versions earlier than 00804. The supported character sets can be found in the x12codes.txt file.

```
ISX02=00803,1|US-ASCII,2|ibm-037,4|Latin1,5|Latin2,6|Cyrillic,7|Greek8,8|Latin3,9|Latin4,10|Arabic,11Hebrew,12|Latin5,13|Latin-9,T4|UTF-8,T5|UTF-16LE,T6|UTF-16BE
```

For example: ISX02 for ISA version 00803, value 1 represents the TX character set US-ASCII and ISX02 value 7 represents TX character set Greek8. The ISX02 values T4, T5, and T6 can be used for testing purposes. Example data in UTF-8 and UTF-16LE have also been included with the X12U component.

A set of maps are included in the X12U component in the map source file x12conv.mms to perform the character decoding and encoding as defined in the ISX and compliance check configuration using the X12U Standard schemas as the target or source for processing.

- **Character Encoding Processing**

The Supply Chain EDI component X12U must be used for character encoding processing. The ISX02 value identifies the character encoding beginning with the group envelope (GS/GE) for the rest of the interchange.

## Character Encoding Processing

The Supply Chain EDI component X12U must be used for character encoding processing. The ISX02 value identifies the character encoding beginning with the group envelope (GS/GE) for the rest of the interchange.

When the data stream is encoded with the ISX encoding value specified, the output or target data stream from the X12U processing will be encoded with the source encoding by default. The configuration settings are provided to override the target encoding. The TA1 does not contain the GS/GE structure and will not be encoded.

It is assumed that the ISX character encoding identification provides encoding at both element and package levels.

The Unicode Standard defines Universal character sets enabling data to be transmitted in single or multi-byte encodings to prevent the loss of data. The universal character sets such as UTF-16 and UTF-32 would indicate Package level encoding. With Package level encoding, the data stream beginning with the GS to the end of the interchange including the segment identification and code list values would be encoded. The segment identification and code list values as defined in the X12 standard would be preserved in single or multi-byte format.

The language specific character sets enable data to be transmitted in a specific language and applies to all the transaction text elements beyond the ISX. The language specific or non-universal character sets would indicate Element level encoding. With Element level encoding, the data stream beginning with the GS to the end of the interchange excluding segment identification and code list values would be encoded. The segment identification and code list values should remain as defined in the X12 standard. Only the text elements should contain the value in the character set as defined in the ISX character encoding identification.

## Validation overview

The EDI Compliance Check framework performs validation as follows:

- Envelope validation - The interchange, group, and transaction envelopes are checked for structure and configurable partner content restrictions. Segment, group and transaction counts are also validated.
  - An interchange that fails the envelope validation cannot continue in the process.
  - If the group envelope validation fails, the contents of the group cannot continue but other groups in the interchange continue.
  - If any transaction fails the envelope validation, only that transaction envelope is rejected, and all other transactions continue to be validated.

- Schema validation - the data in each transaction is compared to the pattern defined by the appropriate version of the standard EDI schema. If the data matches the pattern in the schema, it passes validation and no further processing is necessary.
- Segment structure validation - if data fails the schema validation step, it is passed through more validation. The transaction is checked for segment structure errors by using an external library to validate the order of the segments. This step checks for errors such as missing mandatory segments, or loops, too many repetitions of a segment or loop, or unrecognized segments.
- Segment detail validation - if no segment structure errors are found, the data is passed to a segment map to check for data element errors within the segments. This uses the audit log information to collect information about the location and reason for the error. Examples of errors that are detected by this step include the missing data elements, invalid data element values, and data elements that are too long or too short.
- The EDI Compliance Check framework collects all error information into a detailed error log and creates the acknowledgment. The error log and acknowledgment can report multiple errors if multiple errors are detected. However, there are some situations in which only the first error is reported. For example, if an error is found in the envelope, then detailed validation on the contained transactions is not performed. There are also some cases such as missing loops, or segments, where more error messages are suppressed because they would be misleading as they are the result of recovery attempts from the initial error. An optional JSON report component may also be created with configuration options for Accepted, Full, Never, and Rejected along with Interchange, Group, and Transaction header and trailer images from the inbound document.

Note: Incoming acknowledgments in the interchange are not validated or acknowledged.

## Validation checklist

Validation involves ensuring the EDI data is syntactically correct as well as the message structure.

The following is a list on syntax checks performed:

- Length - Both minimum and maximum lengths are checked.
- Data type - Data types are checked to ensure that numerical values, text values and date values are correctly represented.
- Qualifier restrictions - Qualifier restriction values are checked against restriction lists.
- Empty data elements - Empty data elements are checked to make certain there are no missing mandatory elements. For X12 and RAIL data, data elements that are marked as mandatory must have content if the segment is present. In the case of a composite data element, at least one component data element must have content if the composite is present.
- Relational conditions may exist between one or more data elements based on the presence or absence of those data elements. These conditions are verified.
- Unexpected elements/segments/loops - Loops and segments are checked to make sure that they occur in the data only where expected and do not occur where they are not expected. This error type can also be reported as a "missing mandatory" error and include the loop or segment that was expected at that position.
- Repetition - The number of times a loop, segment, or element is found in the data against the range that is specified by the standard.
- Duplicate control number checking (optional) - Optional validation is provided to check for duplicate control numbers within the group and transaction envelopes within the interchange. The compliance check maps work with one interchange at a time, so do not keep information about previous interchanges, and therefore, do not check for duplicate interchange control numbers. However, if duplicate interchange control numbers are checked outside of the compliance check maps, an input flag can be passed in, and an appropriate acknowledgment, reporting the duplicate interchange can be generated by the compliance check maps.

## Run the EDI compliance check maps

This section describes the basic steps to run the EDI compliance check maps by using the sample configuration and input data that is provided. These examples show how to run the maps using Design Studio and Design Server.

It is expected that Design Studio transformation extender projects have been defined and reference the Supply Chain EDI pack or Design Server compliance projects for EDIFACT, IATA, X12, or RAIL have been imported. Please review the ITX documentation for more information on defining and importing projects and review the IBM Transformation Extender Pack for Supply Chain EDI for download information.

Each compliance map, as shipped, executes with sample valid input data, which is also shipped. The expected output is valid data located in the respective output folders along with accepted outbound acknowledgments. Execution with sample invalid input data, also shipped, will result in output in the respective reject folders along with an error log and rejected outbound acknowledgments.

- [Run the X12/RAIL compliance check maps](#)

This section describes how to use the sample configuration and input data to run the X12 or RAIL compliance check maps using the Design Studio and Design Server. The X12 travel & transportation subset contains similar naming conventions.

- [Run the EDIFACT/IATA compliance check maps](#)

This section describes how to use the sample configuration and input data to run the EDIFACT compliance check maps using the Design Server.

## Run the X12/RAIL compliance check maps

This section describes how to use the sample configuration and input data to run the X12 or RAIL compliance check maps using the Design Studio and Design Server. The X12 travel & transportation subset contains similar naming conventions.

Compliance check maps are in three map source files: ccx12.mms, ccx12u.mms, x12segment.mms (contains a run map for each x12 version supported), and x12ttval.mms (contains a run map for each x12 version supported). For RAIL the source files are: ccrail.mms, railsegment.mms, and railttval.mms. Additional maps used for decoding and encoding are located in map source file x12conv.mms.

The sections below provide information related to X12 compliance checking but are applicable to both X12 and RAIL. X12 and RAIL processing is very much the same with slight differences.

There are several ways to build the maps including individual map selections using the Design Studio and Design Server, the Command Server, Deployment and Packages, Deployment and Build, etc.

- [To compile all the maps using Design Studio](#)
- [To compile all the maps using Design Server](#)

- [To compile select maps using Design Studio](#)
- [To compile select maps using Design Server](#)
- [Where is my output?](#)

The ccx12/rail maps ship with output destinations using FileOutputStream. This xml element in the configuration file (x12defaultconfig.xml or raildefaultconfig.xml), defines the path for output files.

- [Rest Container](#)

To design and compile ITX maps to be run by the Rest container, ITX Design Studio, or ITX Design Server must be used. The compiled maps then must be saved to the volume, or a host directory, bound to the /data/maps location in the container file system.

---

## To compile all the maps using Design Studio

### Procedure

1. Open Windows explorer.
2. Navigate to <supply chain pack install>  
  \x12\compliance\mapsandschemas
3. Execute the script compileallx12.bat
4. The script may be modified to compile for specific target platform using the -P option. For example:  
  mcompile ccx12.mms -A -O \*.mmc -P AIX

---

## To compile all the maps using Design Server

### Procedure

1. Open the Design Server.
2. Using the Deploy workspace select Packages.
3. Add a new package and select x12Compliance project.
4. Under the Maps section, check the checkbox beside Name.  
  This will select all the x12 compliance check maps.
5. Under the Files section, check the checkbox beside Name.  
  This will select all the x12 files.
6. Click the Add button.  
  This will create the package.
7. Click the Build icon in the top right corner.  
  Note: This will take a few minutes as there are many x12 standards supported. Once the maps are built there is no need to re-build the package unless changes have been applied.
8. Return to the Design Server Home page.
9. Using the Design workspace select Project x12compliance and open the ccx12 map.

---

## To compile select maps using Design Studio

### Procedure

1. Open the Design Studio and select or create a workspace.
2. Create or use an existing Extender project that contains the supply chain edi pack install.
3. Navigate to <supply chain pack install>  
  \x12\compliance\mapsandschemas
4. Open any of the map source files ccx12.mms, x12segment.mms or x12ttval.mms
5. Select the map to build and click the Build Map button at the top.

---

## To compile select maps using Design Server

### Procedure

1. Open the Design Server.
2. Using the Design workspace select x12Compliance project.
3. Search and select the ccx12 map.
4. In the Navigator, find and build the following maps: ccx12, putdatax, x12audit, ttv5010, seg5010. Selection can also be accomplished using Packages.
5. Return to the ccx12 map and open.
6. Select Run and you should see a successful run display.

---

## Where is my output?

The ccx12/rail maps ship with output destinations using FileOutputStream. This xml element in the configuration file (x12defaultconfig.xml or raildefaultconfig.xml), defines the path for output files.

The ccx12u maps use a JSON version of the xml configuration file (x12fullconfig.json.dat). A conversion utility to convert existing XML configuration files to JSON is located in x12\_data\_utilities.mms map source file. Verify the output is valid before JSON implementation.

## Design Studio

---

### File Output

```
<?xml version="1.0" encoding="UTF-8"?>
<Configuration>
 <VersionInfo>X12 10.1.0.0</VersionInfo>
 <RuntimeInfo>
 <FileOutput>
 <FileDirectory>./data/output</FileDirectory>
 </FileOutput>
```

The output file names are generated with a datetime stamp. This creates unique files with each execution of compliance. The unique files setting is defined on input card input card 2 as an echo literal value.

The default location defined in the configuration file is:

```
<supply chain pack install>\x12\compliance\data\output
```

The sub folders underneath contain the interchange and functional acknowledgment files, x12 valid and rejected data, and error log file.

## Design Server

---

### Card Output

```
<?xml version="1.0" encoding="UTF-8"?>
<Configuration>
 <VersionInfo>X12 10.1.0.0</VersionInfo>
 <RuntimeInfo>
 <CardOutput/>
```

When projects are imported using Design Server, example inputs and outputs are provided in the project under the Files tab. The output cards are defined as SINK or work files and can be viewed but not downloaded. Additional configurations are needed to download the output files created during Design Server execution. Please review the Configuration and Debug sections for more details.

One way to generate/download output from running the compliance example using Design Server is to define output destination files.

1. Open the x12Compliance project and the ccx12 map. Output cards are located under Structure.
2. Select the following example files as output cards. Custom files may also be created using the Files Add functions.
  - Output card ValidData  
Select Folder >> compliance/data/output/x12/cc\_x12.in
  - Output card InvalidData  
Select Folder >> compliance/data/output/x12/reject/cc\_x12.reject
  - Output card ErrorLog  
Select Folder >> compliance/data/output/x12/reject/cc\_x12.log
  - Output card AckOut  
Select Folder >> compliance/data/output/fa/outbound/cc\_ack.out
  - Output card TA1Out  
Select Folder >> compliance/data/output/ta1/outbound/cc\_ta1.out
3. Modify settings on each output card identified.  
Select connection >> file, click Next button.
4. Select the file using the dropdown list and click Next.
5. Click OK.
6. Save the ccx12 map.
7. Build the map.
8. Run the map.
9. Under Files, select the files created above and view or download.

To view the files created in the compliance/data/output/x12/reject folder, ccx12 compliance map must execute with invalid input.

1. Under Structure, select input card 1, x12data.
2. Expand Action Properties and select x12\_invalid.txt.
3. Save the map.
4. Build the map.
5. Run the map.
6. Repeat the steps for downloading files.

---

## Rest Container

To design and compile ITX maps to be run by the Rest container, ITX Design Studio, or ITX Design Server must be used. The compiled maps then must be saved to the volume, or a host directory, bound to the /data/maps location in the container file system.

Configuration files are located (relative to the map location) with the default output file location for the container file system defined. The map ccx12 input card ConfigInfo should be modified to use the x12rsconfig.xml file or the x12fullconfig.xml file. After modification of the map, deploy the package to the container file system.

The FileDirectory defined in the x12 configuration file should point to the container file system and mount location.

With container mount location /home/tx-rest/data and package name x12Compliance.

/home/tx-rest/data/maps/x12Compliance/compliance/data/x12rsconfig.xml

```
<FileOutput>
 <!-- This section will send output to files. Reference file x12rsconfig.sh -->
 <FileDirectory>/data/data/x12Compliance</FileDirectory>
</FileOutput>
```

The compliance check output folder structure must also be defined for the container file system and mount location /data/data.

A script file is located (relative to the map location) with the container package deployment to the rest container depending on the server definition in Design Server for deployment.

With container mount location /home/tx-rest/data

/home/tx-rest/data/maps/x12Compliance/compliance/data/x12rsconfig.sh

```
mkdir -p /home//tx-rest/data/x12Compliance/x12/reject
```

```
mkdir -p /home//tx-rest/data/x12Compliance/fa/inbound
```

```
mkdir -p /home//tx-rest/data/x12Compliance/fa/outbound
```

```
mkdir -p /home//tx-rest/data/x12Compliance/ta1/inbound
```

```
mkdir -p /home//tx-rest/data/x12Compliance/ta1/outbound
```

After deploying the package and modifications restart the container.

Please review the container documentation for more information.

---

## Run the EDIFACT/IATA compliance check maps

This section describes how to use the sample configuration and input data to run the EDIFACT compliance check maps using the Design Server.

Compliance check maps are in three map source files: ccedf.mms, edfsegment.mms (contains a run map for each EDIFACT version supported), and edftval.mms (contains a run map for each EDIFACT version supported). For EDIFACT subset for travel and transportation, the map source files are ccctedf.mms, ttedfsegment.mms (contains a run map for each EDIFACT version supported), and ttedftval.mms (contains a run map for each EDIFACT version supported). For IATA-API the map source files are ccapi.mms, apisegment.mms, and apitval.mms. For IATA-PNR the map source files are ccpnr.mms, pnsegment.mms, and prntval.mms.

There are several ways to build the maps including individual map selections using the Design Server, the Command Server, Deployment and Packages, Deployment and Build, etc.

- [To compile all the maps Design Studio](#)
- [To compile all the maps Design Server](#)
- [To compile select maps Design Studio](#)
- [To compile select maps Design Server](#)
- [Where is my output?](#)

The ccedf map ships with output destinations using FileOutput. This xml element in the configuration file (edfdefaultconfig.xml), defines the path for output files.

- [Rest Container](#)

To design and compile ITX maps to be run by the Rest container, ITX Design Studio, or ITX Design Server must be used. The compiled maps then must be saved to the volume, or a host directory, bound to the /data/maps location in the container file system.

---

## To compile all the maps Design Studio

### Procedure

1. Open Windows explorer.
2. Navigate to <supply chain pack install>  
 \edifact\compliance\mapsandschemas
3. Execute the script compilealldf.bat
4. The script may be modified to compile for specific target platform using the -P option. For example:  
 mcompile ccedf.mms -A -O \*.mmc -P AIX

---

## To compile all the maps Design Server

### Procedure

1. Open the Design Server.
  2. Using the Deploy workspace select Packages.
  3. Add a new package and select edifactCompliance project.
  4. Under the Maps section, check the checkbox beside Name.  
This will select all the edifact compliance check maps.
  5. Under the Files section, check the checkbox beside Name.  
This will select all the edifact files.
  6. Click the Add button.  
This will create the package.
  7. Click the Build icon in the top right corner.  
Note: This will take a few minutes as there are many edifact standards supported. Once the maps are built there is no need to re-build the package unless changes have been applied.
  8. Return to the Design Server Home page.
  9. Using the Design workspace select Project edifactcompliance and open the ccedf map.
- 

## To compile select maps Design Studio

### Procedure

1. Open the Design Studio and select or create a workspace.
  2. Create or use an existing Extender project that contains the supply chain edi pack install.
  3. Navigate to <supply chain pack install>\edifact\compliance\mapsandschemas
  4. Open any of the map source files ccedf.mms, edfsegment.mms or edftval.mms
  5. Select the map to build and click the Build Map button at the top.
- 

## To compile select maps Design Server

### Procedure

1. Open the Design Server.
  2. Using the Design workspace select edifactCompliance project.
  3. Search and select the ccedf map.
  4. In the Navigator, find and build the following maps: ccedf, putdatae, edfaudit, ttvd04a, segd04a. Selection can also be accomplished using Packages.
  5. Return to the ccedf map and open.
  6. Select Run and you should see a successful run display.
- 

## Where is my output?

The ccedf map ships with output destinations using FileOutputStream. This xml element in the configuration file (edfdefaultconfig.xml), defines the path for output files.

### Design Studio

#### File Output

```
<?xml version="1.0" encoding="UTF-8"?>
<Configuration>
 <VersionInfo>EDIFACT 10.1.0.0</VersionInfo>
 <RuntimeInfo>
 <FileOutput>
 <FileDirectory>../data/output</FileDirectory>
 </FileOutput>
 </RuntimeInfo>
 </Configuration>
```

The output file names are generated with a datetime stamp. This creates unique files with each execution of compliance. The unique files setting is defined on input card input card 2 as an echo literal value.

The default location defined in the configuration file is:

```
<supply chain pack install>\edifact\compliance\data\output
```

The sub folders underneath contain the interchange and functional acknowledgment files, EDIFACT valid and rejected data, and error log file.

### Design Server

#### Card Output

```
<?xml version="1.0" encoding="UTF-8"?>
<Configuration>
 <VersionInfo>EDIFACT 10.1.0.0</VersionInfo>
 <RuntimeInfo>
 <CardOutput/>
 </RuntimeInfo>
 </Configuration>
```

When projects are imported using Design Server, example inputs and outputs are provided in the project under the Files tab. The output cards are defined as SINK or work files and can be viewed but not downloaded. Additional configurations are needed to download the output files created during Design Server execution. Please review the Configuration and Debug sections for more details.

One way to generate/download output from running the compliance example using Design Server is to define output destination files.

1. Open the edifactCompliance project and the ccddf map. Output cards are located under Structure.
2. Select the following example files as output cards. Custom files may also be created using the Files Add functions.
  - Output card ValidData  
Select Folder >> compliance/data/output/edifact/cc\_edf.in
  - Output card InvalidData  
Select Folder >> compliance/data/output/edifact/reject/cc\_edf.reject
  - Output card ErrorLog  
Select Folder >> compliance/data/output/edifact/reject/cc\_edf.log
  - Output card AckOut  
Select Folder >> compliance/data/output/fa/outbound/cc\_ack.out
3. Modify settings on each output card identified.  
Select connection >> file, click Next button.
4. Select the file using the dropdown list and click Next.
5. Click OK.
6. Save the ccddf map.
7. Build the map.
8. Run the map.
9. Under Files, select the files created above and view or download.

To view the files created in the compliance/data/output/edifact/reject folder, ccddf compliance map must execute with invalid input.

1. Under Structure, select input card 1, edfdata.
2. Expand Action Properties and select edf\_invalid.txt.
3. Save the map.
4. Build the map.
5. Run the map.
6. Repeat the steps for downloading files.

---

## Rest Container

To design and compile ITX maps to be run by the Rest container, ITX Design Studio, or ITX Design Server must be used. The compiled maps then must be saved to the volume, or a host directory, bound to the /data/maps location in the container file system.

Configuration files are located (relative to the map location) with the default output file location for the container file system defined. The map ccddf input card ConfigInfo should be modified to use the edfrsconfig.xml file or the edffullconfig.xml file. After modification of the map, deploy the package to the container file system.

The FileDirectory defined in the EDIFACT configuration file should point to the container file system and mount location.

With container mount location /home/tx-rest/data and package name edifactCompliance.

/home/tx-rest/data/maps/edifactCompliance/compliance/data/edfrsconfig.xml

```
<FileOutput>
 <!-- This section will send output to files. Reference file edfrsconfig.sh -->
 <FileDirectory>/data/data/edifactCompliance</FileDirectory>
</FileOutput>
```

The compliance check output folder structure must also be defined for the container file system and mount location /data/data.

A script file is located (relative to the map location) with the container package deployment to the rest container depending on the server definition in Design Server for deployment.

With container mount location /home/tx-rest/data

/home/tx-rest/data/maps/edifactCompliance/compliance/data/edfrsconfig.sh

```
mkdir -p /home//tx-rest/data/edifactCompliance/edifact/reject
mkdir -p /home//tx-rest/data/edifactCompliance/fa/inbound
mkdir -p /home//tx-rest/data/edifactCompliance/fa/outbound
```

After deploying the package and modifications restart the container.

Please review the container documentation for more information.

---

## Configuration

In addition to the X12, RAIL, or EDIFACT interchange, the EDI compliance check maps require extra input cards.

These additional input cards define the settings that are listed here.

- Configuration settings that describe how the data is to be processed.
- Document-specific settings, such as adding a unique ID to identify the input interchange.
- Static table data that is used as part of the validation. This data is included with the compliance check maps.

These settings control whether certain types of validation are performed on the data, and the allowed values. They also control how, and if, acknowledgments are generated, and where data output is written.

- **Input card details**

The primary compliance check maps, cc12, ccrail, and ccedf take the input cards that are described in the following sections.

- **System information file**

The system information file, edicc.ini, defines certain system-specific information.

- **Configuration information details**

Configuration information is in XML format, and specified by the third input card. The configuration information is similar for X12, X12U, RAIL, IATA, and EDIFACT, but differs slightly because of standards-specific information.

---

## Input card details

The primary compliance check maps, cc12, ccrail, and ccedf take the input cards that are described in the following sections.

- **Card 1: EDI data**

This is the input X12, RAIL, or EDIFACT interchange that is to be validated. It must be a single X12, RAIL, or EDIFACT interchange. Multiple interchanges in a single input card are not permitted. If the input contains multiple interchanges, you get the message "Input valid but unknown data found", and only the first interchange is processed.

- **Card 2: unique ID**

This string value specifies a unique ID that identifies the input interchange. For file output, this ID is included as part of the output file names to help link the acknowledgments, output transactions, and error logs to the original interchange.

- **Card 3: configuration information**

This input card contains XML formatted data that describes processing options.

- **Card 4: duplicate interchange flag**

The duplicate interchange flag is a flag that can be passed to indicate that the interchange is a duplicate.

- **Card 5: table data**

This input card contains table data that is used to help validate restriction lists for X12, RAIL, and EDIFACT envelope segments.

- **Card 6: message code list**

This input card contains the text for error messages that are written to the log file.

- **Card 7: structure detail file**

This input file contains structure information about the expected order of the segments.

- **Card 8: system information file**

The system information file, edicc.ini, defines system information.

- **Card 9: trace flag**

If this is set to T or TU, the map writes an XML formatted trace file to the map directory.

---

## Card 1: EDI data

This is the input X12, RAIL, or EDIFACT interchange that is to be validated. It must be a single X12, RAIL, or EDIFACT interchange. Multiple interchanges in a single input card are not permitted. If the input contains multiple interchanges, you get the message "Input valid but unknown data found", and only the first interchange is processed.

The default for this is just a sample X12, RAIL, or EDIFACT file in the data directory, so this card is almost always overridden. To specify an input file, you might use the -IF1 option with Command Server, for example:

```
-IF1 \DataDir\MyEDIData.txt
```

---

## Card 2: unique ID

This string value specifies a unique ID that identifies the input interchange. For file output, this ID is included as part of the output file names to help link the acknowledgments, output transactions, and error logs to the original interchange.

This string value specifies a unique ID that identifies the input interchange. For file output, this ID is included as part of the output file names to help link the acknowledgments, output transactions, and error logs to the original interchange.

The default value for this is NoUniqueID. When this value is used, an ID is generated based on the current time stamp. You must then override the generated ID with your own unique ID so you can ensure that the ID is unique and link the output to other processes. To specify the unique ID, you can use the -IE2 option, for example:

```
-IE2 ID1234567890
```

---

## Card 3: configuration information

This input card contains XML formatted data that describes processing options.

The default value for this is a sample configuration in the data directory, so this card is normally overridden. To specify a configuration file, you can use the -IF3 option, for example:

## Card 4: duplicate interchange flag

The duplicate interchange flag is a flag that can be passed to indicate that the interchange is a duplicate.

The compliance check maps work only with a single interchange at a time, and do not track the interchanges that are processed before, or after the current interchange. You can, however, track other transactions, and use a preprocessing step to set this flag if the interchange is recognized as a duplicate. The value must be either `T` (interchange is a duplicate), or `F` (the default interchange is not a duplicate). If the flag is set to `T`, then the EDI compliance checking maps generate the appropriate logs and acknowledgment for the duplicate interchange. The default value for the flag is `F`. To set the flag to `T`, you can use the `-IE4` option, for example:

`-IE4 T`

## Card 5: table data

This input card contains table data that is used to help validate restriction lists for X12, RAIL, and EDIFACT envelope segments.

These files, `x12codes.txt` for X12, `railcodes.txt` for RAIL, and `edfcodes.txt` for EDIFACT, are included with the EDI pack. Changes to these files might require changes to the schemas, so do not change these files. Also, do not override the input card.

## Card 6: message code list

This input card contains the text for error messages that are written to the log file.

The error message can be translated to another language. Do not reformat the file or change or remove error code values.

Do not override this input card.

## Card 7: structure detail file

This input file contains structure information about the expected order of the segments.

This input file is used by an exit library that the map calls to check the transaction or message structure. You do not need to change this file, although in some cases you can improve performance by removing the structure information for versions and transactions that you do not use. If you modify this file, be sure to back up the original.

With the extensive list of EDI standard versions supported and the compliance checking process to validate the structure of the messages/transactions, the structure validation file has become large. Utilities in the EDIFACT and X12 components can be used to reduce the size of the structure validation file by selecting standard versions and message/transaction type being used to eliminate unused records.

The Utilities can be found in the utilities folder under the EDIFACT and X12 components.

In addition, a referenced structure file can be found in the `mapsandschemas` folder to be used with input card #7. This file contains a reference record for each message/transaction that identifies each standard version a particular structure record can be used with.

## Card 8: system information file

The system information file, `edicc.ini`, defines system information.

The system information that it defines includes the location of the exit library, which are used during system runtime executions for compliance checking. Do not override this input card.

The system information file, `edicc.ini`, defines certain system-specific information. You update this file when the EDI compliance checking maps are deployed to a system. After that time, the file does not need to be updated. The properties that this file includes are described here.

- **ismvs**

This setting is for z/os processing and should contain the value yes for z/os processing only. For all other platforms, the value should be no.

- **mvsrunmaplog**

This setting is for z/os processing and is used to suppress z/os run map logs during execution. The value `/NL` will be used with each run map call. This can improve performance with large files. For all other platforms, the value should be N/A.

- **exitlib**

The exitlib value should specify the path to the directory that contains the `edisvu` shared library. This can be either a relative path (relative to the map directory), or a fully qualified path name. For Unix systems, this would be path to `edisvu.so` or `edisvu.sl` file, and on Windows it will be the path to `edisvu.dll`.

## exitlib

The exitlib value should specify the path to the directory that contains the edisvu shared library. This can be either a relative path (relative to the map directory), or a fully qualified path name. For Unix systems, this would be path to edisvu.so or edisvu.sl file, and on Windows it will be the path to edisvu.dll.

The exit edisvu is installed with the Design Studio with the default setting below of ./.

The exit edisvu is installed with the Design Server with the default setting below of NA. Script files for replacing the exit are in the ediPlatformSupport project.

If the location of the edisvu exit has not changed, there are no updates needed for the edicc.ini file.

- Setting ./ is relative to map execution folder
- Setting N/A or NA (Not Applicable) identifies the ITX install folder
- Web UI Design Server execution must use N/A or NA

## Design Studio

---

The default setting for the exit module location is relative to the map execution folder and the exit library should be found at this location. The platform\_support folder contains the exit library for all operating systems.

For z/os the exit library should be in the JCL steplib.

## Design Server

---

The exit library must be in the ITX install folder and the exit module should be found at this location.

The ediPlatformSupport\* project zip files for the Design Server. This project also contains scripts to assist removal and placing the exit library to the correct location.

The exit library must be installed with the Windows Design Server and must be in the ITX install folder.

## Card 9: trace flag

---

If this is set to T or TU, the map writes an XML formatted trace file to the map directory.

The default value for this is None. To set this to T, or TU, you can use the -IE9 option, for example:

-IE9 T

## System information file

---

The system information file, edicc.ini, defines certain system-specific information.

## Configuration information details

---

Configuration information is in XML format, and specified by the third input card. The configuration information is similar for X12, X12U, RAIL, IATA, and EDIFACT, but differs slightly because of standards-specific information.

The XML formats are defined by the XML schema:

- x12cfg.xsd for X12
- x12cfg.json for X12U
- railcfg.xsd for RAIL
- edfcfg.xsd for EDIFACT and IATA

The configuration information can be in a static location, or it can be generated for each message by a preprocessing step based on some criteria such as trading partner information. Many settings have default values, so they do not need to be specified whether or not you want to use the default behavior.

This documentation describes the options that can be configured. Unless otherwise noted, the elements apply to X12, X12U, RAIL, IATA, and EDIFACT.

- **RuntimeInfo**  
The settings provided here contain information that is related, typically, to your runtime environment, rather than to specific trading partners.
- **CardOutput**  
If the CardOutput element is specified, output is written to map output cards, and the PUT calls are skipped.
- **FileOutput**  
If the FileOutput element grouping is specified, the compliance check maps write the output to files.
- **QueueOutput**  
If the QueueOutput element is specified, the compliance check maps write the output to MQ queues. This element also contains information that is needed by the MQ Adapter in order to write the data, and identifies to which queues the different types of output data should be written.
- **WireOutput**  
If the WireOutput element is specified, the compliance check maps write the output to wire terminals.
- **CreateRejectLog**  
The CreateRejectLog element specifies whether the compliance check maps are to create plain text error logs associated with any rejected transactions, groups, or interchanges.
- **JsonReport**  
The JsonReport element specifies whether the compliance check map should create JSON reports associated with any accepted/rejected transactions, groups, or interchanges. The default setting is N for Never.

- **[MaxNumErrors](#)**  
The MaxNumErrors element specifies whether the compliance check map report all errors or limit the number of errors generated. A value of zero indicates all errors will be generated.
- **[WorkSpace](#)**  
The WorkSpace element specifies whether the compliance check maps should set the workspace option to use memory or disk for runmap calls. The default setting is memory-based (-WM). A value (-WU) identifies workspace to use file-based options.
- **[SkipITValidation](#)**
- **[ISXcontrol - X12](#)**  
The ISXcontrol group specifies how the ISX control segment should be processed.
- **[PartnerInfo](#)**  
This element contains information that is based on trading partner relationships. For example, it includes validation and acknowledgment options.
- **[GenerateTA1 - X12, RAIL only](#)**  
The GenerateTA1 element specifies whether a TA1 interchange acknowledgment is generated when requested by the inbound interchange.
- **[AckInfo](#)**  
For X12, this documentation specifies whether 997/999 acknowledgments are generated, and how they are generated.
- **[CheckDuplicateGroupControl](#)**  
The CheckDuplicateGroupControl element specifies whether the compliance check maps are to check the interchange for duplicate groups.
- **[CheckDuplicateTrxControl](#)**  
The CheckDuplicateTrxControl element specifies whether the compliance check maps are to check each group for duplicate transactions and messages.
- **[Authority - X12, RAIL only](#)**  
This documentation specifies whether the Authorization information elements (ISA01 and ISA02) are to be checked for specific values, and if they are, what values they are to contain.
- **[Security - X12, RAIL only](#)**  
This element specifies whether the Security information elements (ISA03 and ISA04) are to be checked for specific values, and if so, what values they contain.
- **[RecipRef - EDIFACT only](#)**  
This documentation specifies whether the Recipient Reference composite, UNB06 is to be checked for specific values, and if so, what values it must contain. If this checking is requested, and the subelements do not contain the correct values, then the interchange is rejected and appropriate acknowledgments are generated.

## RuntimeInfo

The settings provided here contain information that is related, typically, to your runtime environment, rather than to specific trading partners.

This includes settings that describe where the output is to be written, and whether the error log is to be written.

One of the following element groupings is required:

- CardOutput - indicates that outputs are to be written to map output cards.
- FileOutput - indicates that output is to be written to files.
- QueueOutput - indicates that output is to be written to queues.
- WireOutput - indicates that output is to be written to wire terminals (used with IBM Sterling Transformation Extender Only).

## CardOutput

If the CardOutput element is specified, output is written to map output cards, and the PUT calls are skipped.

The following table describes which cards are used for each type output.

### Output

- Cards 1 - 4 continue to be used for internal processing, and output from these cards is not intended to be used by the map caller. These are used to gather information reported with the map trace Input card 9.
- By default, all the output cards in the following section are set to use the Sink adapter, so no physical output is generated for them. Although there is no physical output file to be downloaded with the Sink adapter, outputs may be viewed using the Design Server UI.

If you are using the CardOutput element, you must override the output cards that you use:

Card number	Card name	Description
5	ValidData	X12 interchange that contains all valid and accepted transactions. This output is only produced if one or more valid transaction sets are present in the input data that is to be validated.
6	InvalidData	X12 interchange that contains all invalid and rejected transactions and groups. This output is only produced if there are one or more invalid transaction sets or groups, or if the interchange header, or trailer errors are found in the input data that is validated.
7	Error log	Flat file that contains text descriptions of any errors that are detected, and where the errors are located.  This output is only produced if there are one or more invalid transaction sets, or groups, or if the interchange header, or trailer errors are found in the input data that is to be validated.  This information can also be parsed by use of a user application to reformat error information, if needed.
8	AckIn	X12 interchange that contains all inbound 997, or 999 transactions from the interchange. This output is only produced if the input data that is to be validated contains 997, or 999 acknowledgments.
9	AckOut	X12 Interchange containing generated 997 acknowledgments.  This output is only produced if functional acknowledgment (997) transactions are generated for the input data that is to be validated.
10	TA1In	X12 interchange that contains all inbound TA1 acknowledgments from the interchange. This output is only produced if the input data that is to be validated contains TA1 acknowledgments.

Card number	Card name	Description
11	TA1Out	X12 Interchange containing generated TA1 acknowledgments. This output is only produced if TA1 acknowledgments are generated for the input data that is to be validated.
12	FrameworkTrace	XML-formatted trace log for problem determination. This is only created if there are certain framework errors, or if the value T, or TU is specified for input card 9 (trace flag).
13	JsonLog	X12 JSON formatted report providing information related to the Interchange, Group, and Transaction errors and acknowledgments.

## FileOutput

If the FileOutput element grouping is specified, the compliance check maps write the output to files.

See [File naming conventions](#) for information about how the files are named, and the subdirectories to which they are written.

This element contains the FileDirectory element. The FileDirectory element specifies the top-level directory for the file output. The output is written to subdirectories within this directory.

The FileDirectory element is required if the FileOutput element is specified.

- [FileDirectory](#)

The FileDirectory element is required if the FileOutput element is specified.

## FileDirectory

The FileDirectory element is required if the FileOutput element is specified.

When you use file output, this directory, and the following sub-directories, must be created before the compliance check maps are run. Otherwise, the file adapter fails when it attempts to write the output files.

edifact	EDIFACT only
edifact/reject	EDIFACT only
fa	EDIFACT and X12
fa/inbound	EDIFACT and X12
fa/outbound	EDIFACT and X12
ta1	X12 only
ta1/inbound	X12 only
ta1/outbound	X12 only
x12	X12 only
x12/reject	X12 only

Note: The default directories are included with the pack compliance check installation. You can copy this directory structure to your output location.

Note: Eclipse does not copy empty folders when you import a project.

## QueueOutput

If the QueueOutput element is specified, the compliance check maps write the output to MQ queues. This element also contains information that is needed by the MQ Adapter in order to write the data, and identifies to which queues the different types of output data should be written.

This QueueOutput element contains the elements listed here.

- QueueServerClient
- QueueManagerName
- QueueNames
- [QueueServerClient](#)  
The QueueServerClient element indicates which WMQ mode the adapter is to use.
- [QueueManagerName](#)  
The QueueManagerName element applies to both X12, RAIL, and EDIFACT. It indicates which WMQ queue manager that the adapter must use. This value is required if the QueueOutput element is used.
- [QueueNames](#)  
The QueueNames element includes the queue names that are to be used.

## QueueServerClient

The QueueServerClient element indicates which WMQ mode the adapter is to use.

The allowed values are listed here.

- SERVER - The adapter uses WMQ server mode.
- CLIENT - The adapter uses WMQ client mode.

This value is required if the QueueOutput element is specified.

## QueueManagerName

The QueueManagerName element applies to both X12, RAIL, and EDIFACT. It indicates which WMQ queue manager that the adapter must use. This value is required if the QueueOutput element is used.

## QueueNames

The QueueNames element includes the queue names that are to be used.

This value is required if the QueueOutput element is specified.

For X12 data, the QueueNames element contains the following elements. All are required.

For EDIFACT data, the QueueNames element contains the following elements. All are required.

Scripts to create these queues are included in the compliance check data directories. The create\_x12\_queues.mqsc script creates the queues for X12, create\_rail\_queues.mqsc for RAIL, and the create\_edf\_queues.mqsc script creates the queues for EDIFACT. The runmqsc utility from MQ is used to run both scripts. Additional information is included in the comments at the beginning of the scripts.

These queues are not accessed until the compliance check maps are ready to write data to them. If you know that certain types of output are not to be generated, for example TA1 acknowledgments, you can specify a nonexistent queue name in the configuration such as "DUMMY\*". However, this can cause adapter errors if the output is later generated due to other configuration changes. A better solution is to route any unexpected output types to a special queue, such as EDI\_OTHER.

X12_TransactionQueue	Used for valid transaction output.
X12_RejectQueue	Used for rejected transactions, groups, and interchanges.
X12_RejectLogQueue	Used for error logs associated with the rejected transactions, groups, and interchanges.
FA_InboundQueue	Used for received, or inbound, 997 and 999 acknowledgments.
FA_OutboundQueue	Used for generated, or outbound 997 and 999 acknowledgments.
TA1_InboundQueue	Used for received, or inbound, TA1 acknowledgments.
TA1_OutboundQueue	Used for generated, or outbound, TA1 acknowledgments.
EDIFACT_TransactionQueue	Used for valid transaction output.
EDIFACT_RejectQueue	Used for rejected messages, groups, and Interchanges.
EDIFACT_RejectLogQueue	Used for error logs associated with the rejected messages, groups, and interchanges.
CONTRL_InboundQueue	Used for received, or inbound, CONTRL acknowledgments.
CONTRL_OutboundQueue	Used for generated, or outbound, CONTRL acknowledgments
X12_JsonOutQueue	Used for X12 JSON formatted report to provide information related to the Interchange Group, and Transaction and acknowledgments.
X12_JsonRejQueue	Used for X12 JSON formatted report to provide information related to the Interchange Group, and Transaction errors and acknowledgments.

## WireOutput

If the WireOutput element is specified, the compliance check maps write the output to wire terminals.

The WireOutput element can be used only with IBM Sterling Transformation Extender.

## CreateRejectLog

The CreateRejectLog element specifies whether the compliance check maps are to create plain text error logs associated with any rejected transactions, groups, or interchanges.

These values are allowed.

- T (True) - Create the plain text reject log.
- F (False) - Do not create the plain text reject log.

The default value is T (True).

## JsonReport

The JsonReport element specifies whether the compliance check map should create JSON reports associated with any accepted/rejected transactions, groups, or interchanges. The default setting is N for Never.

The valid values are:

- ( A )ccepted
- ( F )ull
- ( N )ever
- ( R )ejected

Full report indicates both accepted and rejected. The default is Never.

There are additional configuration settings to report the Interchange, Group, and Transaction images.

## MaxNumErrors

The MaxNumErrors element specifies whether the compliance check map report all errors or limit the number of errors generated. A value of zero indicates all errors will be generated.

Large files that contain thousands of errors, present performance issues related to memory usage. To address this issue, this value will restrict the number of errors to be generated. The default value is 0 (zero meaning there is no maximum). During processing if the maximum number of errors has been exceeded, the number of errors generated in the reject log and acknowledgment will be truncated and the following error E9004 will be logged.

E9004 The number of segment and element errors has exceeded the maximum (100). The error log and acknowledgment output has been truncated.

## WorkSpace

The WorkSpace element specifies whether the compliance check maps should set the workspace option to use memory or disk for runmap calls. The default setting is memory-based (-WM). A value (-WU) identifies workspace to use file-based options.

The default setting uses work files that are in memory, and is best for small to medium size transactions. The transaction size at which performance improves by using a work file on disk, depends upon the amount of memory available after other applications and services are taken into account.

The current framework is set up to handle small data as quickly as possible. If your large data is taking longer than expected to process, you can use this setting to use disk instead of memory with the value -WU.

The setting can also include page size, for example, -WM -P128:16.

## SkipTTValidation

There are three main processes performed during the compliance checking, as follows:

- Standard Type Tree validation
- Structure checking
- Segment checking

The Standard Type Tree validation uses the full standard to determine if the input file contains errors. If there are no errors, no further validation is needed. If there are errors, the second and third processes are executed. This first process can take a considerable amount of time depending on the standard and the message type.

The SKIPTTVALIDATION setting will skip the first process and always execute the structure checking and segment checking and eliminate the amount of data parsing and restarts. It also eliminates the ttval run map execution.

Valid values are T(rue) and F(alse). The default value is F(also) to identify execution of the three processes.

## ISXcontrol - X12

The ISXcontrol group specifies how the ISX control segment should be processed.

When present, the ISX segment can identify release character syntax, X12 code list version overrides, and the character encoding of the interchange following the ISX segment. Additional processing is required and will affect performance. These configuration settings override default processing and control how the data is processed when the ISX is present or absent. It is not required to define these controls unless you want to override the default processing. However, if the ISX is present in the input data stream, minimum configuration is required to avoid unexpected ISX between partners and requires the ISXcontrol group to be present in the configuration file.

IgnoreISX component must be set.

```
<ISXcontrol>
<IgnoreISX>F</IgnoreISX>
</ISXcontrol>
```

- **IgnoreISX**

The IgnoreISX component indicates whether to ignore the ISX overrides (ISX02, ISX03).

- **ReleaseCharacter**

The ReleaseCharacter group identifies optional configuration related to the ISX segment and the Release Character definition that will be used to indicate that the characters following it should be interpreted as the data value, not as a delimiter used for syntax and parsing.

- **CharacterEncoding**

The CharacterEncoding group identifies optional configuration related to additional processing required to perform inbound (source) and outbound (target) character encoding.

- **CodeListVersionOverride**

The CodeListVersionOverride group identifies optional configuration related additional processing required to perform code list validations using a different X12 standard version of the code list than is defined with the GS08.

---

## IgnoreISX

The IgnoreISX component indicates whether to ignore the ISX overrides (ISX02, ISX03).

Valid value are T and F. If not specified, the default is F.

---

## ReleaseCharacter

The ReleaseCharacter group identifies optional configuration related to the ISX segment and the Release Character definition that will be used to indicate that the characters following it should be interpreted as the data value, not as a delimiter used for syntax and parsing.

Specify the DefaultReleaseCharacter to define a release character syntax when the ISX01 value is missing and the X12 default syntax should not be used. Specify the OverrideReleaseCharacter to define a release character syntax to override the ISX01 value.

DefaultReleaseCharacter – default release character syntax

OverrideReleaseCharacter - default release character syntax ISX01

---

## CharacterEncoding

The CharacterEncoding group identifies optional configuration related to additional processing required to perform inbound (source) and outbound (target) character encoding.

The AllowEncoding value F is the default to not allow ISX02 and character encoding definitions. A value of T will enable any character encoding specifications. The AllowEncoding indicates whether to allow character encoding and perform additional decode/encode processing. If not specified, additional processing will be performed for the encoding identified (default to F).

Character encoding requires new processing found in component x12u and map ccx12u. The values below define a character encoding for inbound (source) or outbound (target). If specified, the value must be supported by both x12 and TX. This value will be used when the ISA version is 00704 or higher. Supported values can be found in the documentation and x12codes.txt file.

The DefaultSourceEncoding specifies a value to be used when the ISX02 value is absent.

The OverrideSourceEncoding specifies a value to be used to override the ISX02 value.

The DefaultTargetEncoding specifies a value to be used when the ISX02 is absent for all compliance checking outputs.

The OverrideTargetEncoding specifies a value to be used to override the ISX02 value. If encoding is not defined, the default encoding for ccx12 compliance checking is Western/Native and for ccx12u is Western/UTF-8.

---

## CodeListVersionOverride

The CodeListVersionOverride group identifies optional configuration related additional processing required to perform code list validations using a different X12 standard version of the code list than is defined with the GS08.

The AllowCodeListOverride value F is the default to not allow ISX02 and code list version overrides. A value of T will enable any code list version specifications. The AllowCodeListOverride indicates whether to allow code list version override and perform additional validations. If not specified, additional validations will be performed for the version identified (default to F). If specified, the value must be T or F. A value of F will ignore code list version override.

The values below define a code list version to be used for code list validation. If specified, the value must be supported by both x12 and TX. This default will be used when the ISA version is 00801 or higher.

The DefaultCodeListVersion specifies a value to be used when the ISX03 value is absent.

The OverrideCodeListVersion specifies a value to be used to override the ISX03 value.

---

## PartnerInfo

This element contains information that is based on trading partner relationships. For example, it includes validation and acknowledgment options.

This element is optional. If present it includes these elements:

- GenerateTA1 - X12, RAIL only
- AckInfo
- CheckDuplicateGroupControl
- CheckDuplicateTrxControl

- Authority - X12, RAIL only
  - Security - X12, RAIL only
  - RecipRef - EDIFACT only
- 

## GenerateTA1 - X12, RAIL only

The GenerateTA1 element specifies whether a TA1 interchange acknowledgment is generated when requested by the inbound interchange.

The allowed values for this element are listed here.

- R (Requested) - Generate TA1 if requested by ISA14 value. If the ISA14 value is "1", this setting always generates a TA1, even if the interchange envelope is valid.
- N (Never) - Never generate TA1, even if requested by ISA14 value.

The default value is R(Requested).

Note: TA1 acknowledgments are not generated if the data does not contain any transactions. This is to avoid a TA1 loop, per X12 standards.

---

## AckInfo

For X12, this documentation specifies whether 997/999 acknowledgments are generated, and how they are generated.

For EDIFACT, the element specifies whether CONTRL acknowledgments are to be generated, and how they are generated.

This element includes the AckType (X12 only), AckLevel and the AckVersion (X12, RAIL only) elements.

- **AckType**  
The Acktype element specifies 997 or 999 acknowledgments should be generated. The default is 997.
  - **AckLevel**  
This AckLevel element specifies whether acknowledgments are to be generated, and if so, at what level.
  - **ValidateContrl**
  - **AckVersion - X12, RAIL only**  
If acknowledgments are generated, the AckVersion element specifies which version is to be used. The version must be a valid X12, RAIL version such as "004010". If no value is specified, the default is to use the same version as the group that is being acknowledged.
- 

## AckType

The Acktype element specifies 997 or 999 acknowledgments should be generated. The default is 997.

---

## AckLevel

This AckLevel element specifies whether acknowledgments are to be generated, and if so, at what level.

The allowed values for the AckLevel element are described here.

- N (Never) - Do not generate 997/999 acknowledgments for X12, RAIL, or CONTRL acknowledgments for EDIFACT.
- T (Transaction) - Generate 997/999 or CONTRL acknowledgments at the transaction/message level. Explicitly acknowledge valid transactions by generating AK2/AK5, or UCM segments for each transaction/message, whether valid or invalid.

The default value for this element is T (Transaction).

---

## ValidateContrl

The ValidateContrl element specifies whether inbound EDIFACT **CONTRL** acknowledgments are to be checked for compliance. The value of T will validate the inbound **CONTRL** message and produce valid or invalid output with a reject log.

Valid values are (T)rue and (F)alse. The default value is (F)alse.

---

## AckVersion - X12, RAIL only

If acknowledgments are generated, the AckVersion element specifies which version is to be used. The version must be a valid X12, RAIL version such as "004010". If no value is specified, the default is to use the same version as the group that is being acknowledged.

For EDIFACT data, this element is not used, and the CONTRL acknowledgment is generated by using the same syntax version as the interchange that is acknowledged.

---

## CheckDuplicateGroupControl

The CheckDuplicateGroupControl element specifies whether the compliance check maps are to check the interchange for duplicate groups.

The allowed values for the CheckDuplicateGroupControl element are described here.

- T (True) - Check the interchange for duplicate groups. For X12 and RAIL, the CheckDuplicateGroupControl element checks for duplicate group control numbers. For EDIFACT, the CheckDuplicateGroupControl element checks the combination of application sender, application receiver, and group reference number.
- F (False) - Do not check the interchange for duplicate groups.

The default value is T (True).

## CheckDuplicateTrxControl

The CheckDuplicateTrxControl element specifies whether the compliance check maps are to check each group for duplicate transactions and messages.

The allowed values for the CheckDuplicateTrxControl are described here.

- T - (True) Check groups for duplicate transactions and messages. For X12 and RAIL, CheckDuplicateTrxControl checks for duplicate transaction control numbers. For EDIFACT, CheckDuplicateTrxControl checks the combination of message ID and message reference number.
- F - (False) Do not check groups for duplicate transactions and messages. Skipping the duplicate checking might save time for groups that contain many transactions.

The default value is T (True).

## Authority - X12, RAIL only

This documentation specifies whether the Authorization information elements (ISA01 and ISA02) are to be checked for specific values, and if they are, what values they are to contain.

If checking is requested, and the fields do not contain the correct values, the interchange is rejected and appropriate acknowledgments are generated.

These elements are described in this documentation.

- [Validate](#)  
The Validate element specifies whether the compliance check maps are to check the Authorization information values.
- [AuthSecInfo](#)  
If Authority information checking is requested, this element specifies the values.

## Validate

The Validate element specifies whether the compliance check maps are to check the Authorization information values.

The Validate element includes these values.

- T (True) - Verify that the Authorization information elements in the ISA match the values that are specified in the AuthSecInfo element.
- F (False) - Do not check the Authorization information elements.

The default value is F (False).

## AuthSecInfo

If Authority information checking is requested, this element specifies the values.

Authority information checking includes the qualifiers described here.

- Qualifier - The value that must be specified in the Authorization information qualifier (ISA01).
- AuthSecID - The value that must be specified in the Authorization information element (ISA02).

If authority validation is requested, both of these elements must be specified and must match the values in the ISA for the interchange to be accepted.

## Security - X12, RAIL only

This element specifies whether the Security information elements (ISA03 and ISA04) are to be checked for specific values, and if so, what values they contain.

If this checking is requested and the fields do not contain the correct values, then the interchange is rejected and appropriate acknowledgments are generated.

These elements are described here.

- [Validate](#)  
The Validate element specifies whether the compliance check maps are to check the Security Information values.
- [AuthSecInfo](#)  
If Security Information checking is requested, this element specifies the values.

## Validate

The Validate element specifies whether the compliance check maps are to check the Security Information values.

The allowed values for this element are described here.

- T (True) - Verify that the Security Information elements in the ISA match the values specified in the AuthSecInfo element.
- F (False) - Do not check the Security Information elements.

The default value is F (False).

## AuthSecInfo

If Security Information checking is requested, this element specifies the values.

Security Information checking includes these elements.

- Qualifier - The value that must be specified in the Security Information Qualifier (ISA03).
- AuthSecID - The value that must be specified in the Security Information element (ISA04).

If Security validation is requested, both of these elements must be specified and must match the values in the ISA for the interchange to be accepted.

## RecipRef - EDIFACT only

This documentation specifies whether the Recipient Reference composite, UNB06 is to be checked for specific values, and if so, what values it must contain. If this checking is requested, and the subelements do not contain the correct values, then the interchange is rejected and appropriate acknowledgments are generated.

These elements are described here.

- **Validate**  
This element specifies whether the compliance check maps are to check the Recipient Reference values.
- **RecipRefInfo**  
If Recipient Reference checking is requested, this element specifies the values.

## Validate

This element specifies whether the compliance check maps are to check the Recipient Reference values.

The allowed values for the Validate element are described here:

- T (True) - Verify that the Recipient Reference elements in the UNB match the values that are specified in the RecipRefInfo element.
- F (False) - Do not check the Recipient Reference values.

The default value is F (False).

## RecipRefInfo

If Recipient Reference checking is requested, this element specifies the values.

Recipient Reference checking includes the following elements:

- RecipRefInfo - The value that must be specified in the Recipient Reference element (UNB0601).
- RecipRefQual - The value that must be specified in the Recipient Reference Qualifier element (UNB0602).

If Recipient Reference validation is requested, both of these elements must match the values in the UNB, or be omitted from the configuration and from the UNB, for the interchange to be accepted.

## Deploy compliance check maps

Compile the maps for the target operating system, and then copy the compiled maps, the exit library, and other associated files to the target system. These steps are similar for both X12, RAIL, and EDIFACT.

- **Map deployment procedure**

For all operating systems, follow the steps provided here.

## Map deployment procedure

For all operating systems, follow the steps provided here.

## Procedure

---

### 1. Build the maps:

There are several ways to build the maps including individual map selections using the Design Studio or Design Server, the Command Server, Deployment and Packages, Deployment and Build, etc.

### 2. Copy the following files to the target system as binary and place the files in the same directory as the compiled maps:

- All compiled maps. If you are sure that certain versions of the standards are not going to be used again, you can skip the seg\* and ttv\* maps that correspond to those versions. As an example, If you are not going to use X12 version 002040 again, you do not need to upload the ttv2040 or seg2040 compiled maps. However, if these maps are not present and an X12 002040 transaction is encountered, the compliance check maps cannot process the transaction correctly.
- Configuration schema x12cfg.xsd (X12), railcfg.xsd (RAIL), or edfcfg.xsd (EDIFACT).
- The shared exit library, which is found under the platform\_support directory. Be sure to choose the operating system and bit type (32-bit or 64-bit) that matches your Transformation Extender installation.

### 3. Copy the following files to the target system as ascii, and then place the files in the same directory as the compiled maps:

Note: When you copy these files to UNIX systems, the CR characters (x0d) must be removed. If the process that you use to copy them (for example, FTP) does not remove them, then you must remove them manually.

- The file that contains error message text, errcodes.txt (X12, RAIL, and EDIFACT).
- Files with table data used by the compliance check maps: x12codes.txt and x12struct.txt (X12), railcodes.txt and railstruct.txt (RAIL), or edfcodes.txt and edfstruct.txt (EDIFACT).
- The system information file edicc.ini.

### 4. Edit the edicc.ini file on the target system, and set the exitlib value to point to the shared exit library that you uploaded in the earlier step.

- For Unix systems, this would be path to edisvu.so or edisvu.sl file, and on Windows it will be the path to edisvu.dll. (DO NOT INCLUDE FILE NAME IN PATH)
- Setting ./ is relative to map execution folder.
- Setting N/A or NA (Not Applicable) identifies the ITX install folder.

### 5. Set up your configuration file to point to the output location that you want to use. If you are using file output, you must create the following subdirectories under the output directory that you specify in your configuration:

edifact	EDIFACT only
edifact/reject	EDIFACT only
fa	EDIFACT, X12, and RAIL
fa/inbound	EDIFACT, X12, and RAIL
fa/outbound	EDIFACT, X12, and RAIL
ta1	X12, RAIL only
ta1/inbound	X12, RAIL only
ta1/outbound	X12, RAIL only
x12	X12 only
x12/reject	X12 only
rail	RAIL only
rail/reject	RAIL only

When the preceding steps are complete, the ccx12, ccrail, and ccedf maps are ready to run on the target system.

### [z/OS Implementation](#)

#### [Installing the shared exit library component on z/OS](#)

After installing the Supply Chain EDI Component on a Windows workstation, perform the following steps for each z/OS execution platform. This is done prior to porting and executing the Supply Chain X12 Compliance Check application on that platform.

#### [Installing the shared exit library component on UNIX](#)

After installing the Supply Chain EDI Component on a windows workstation, perform the following steps for each UNIX execution platform. This is done prior to porting and executing the Supply Chain X12 Compliance Check application on that platform.

---

## z/OS Implementation

### Batch Processing

The edicc.ini file contains system information for the runtime system.

- The exitlib value should be set to EDISVU.  
exitlib=EDISVU
- The ismvs value should be set to yes if IBM/TX is running in z/OS batch mode.  
ismvs=yes
- The mvsrunmaplog value should be set to no if run map logs should be suppressed.  
mvsrunmaplog=no

### CICS Processing

The edicc.ini file contains system information for the runtime system. Although the settings in edicc.ini file are not used during the CICS processing, the file is used as input card 8 for compliance checking.

With CICS processing, the EDISVU exitlib is defined with the CICS CSD definitions. The recommended CICS CSD definitions are as follows:

- DEFINE PROGRAM (EDISVU)
- GROUP (DTXCICS)
- DESCRIPTION (Packs Compliance)

- LANGUAGE (LE)
- CONCURRENCY (THREADSAFE)
- DATA (ANY)

Note: Refer to the following topics for more information related to the CICS CSD definitions:

- [Updating CICS CSD definitions and DFHRPL](#)
- [Using the EXIT function](#)

Additional changes may be needed depending on the pack version used. Verify the compliance check maps: ccedf, ccx12, etc to confirm the COMPLIANCE map function is being used.

- main map : Output Card 1

```
ExitDLLVersion DLLValidate Element:Runtime:SetupFramework
```

- map rule:

```
= VALID (COMPLIANCE ("X12", "GetHCIPVersion", "n/a", ExitDLL DLLValidate Element:Runtime:SetupFramework), "FAILED - GetHCIPVersion call: " + LASTERRORCODE() + "/" + LASTERRORMSG () + "<NL>Check the edicc.ini file to make sure that it correctly specifies the edisvu shared library")
```

- RunDLLSegCheck function map:

```
= RunDLLCompliance(
 Transaction,
 Structure_Detail_File,
 Runtime)
```

The Supply Chain EDI components for compliance using X12, RAIL, and EDIFACT contain XML configuration files that are processed using input card 3. The default setting on input card 3 is set to schema validation and there are two options for handling the schema validation:

- Modify the schema name in the metadata location on input card 3 to contain //DD:name, for example //DD:X12CFG which corresponds to the data set name in the execution JCL. The XML schema should be uploaded as binary to the dataset containing the maps.
- Modify the schema validation setting to Well Formed.

The Supply Chain EDI component for compliance X12U contains a JSON configuration file that is processed using input card 3. Input card 3 is using a JSON template for processing. The JSON template is not required for processing on z/OS.

Each Supply Chain EDI component contains an XML or JSON configuration file to control output to files. Output cards 2 and 3 are defined as SINK with ONSUCCESS APPEND. Output cards 2 and 3 will require an output data set to be defined and an execution parameter override for -OF2 and -OF3 to be identified in the execution JCL.

#### File Output:

The default setting is file output and the map putmvs\* in each component main execution map source file will be used to direct output for z/OS processing. With CICS execution, review the [z/OS CICS execution commands](#).

Pre-defined data set names will be used as follows:

Card number	Card name	Description
Output Card 5	DD:DATAIN	Contains valid data results.
Output Card 6	DD:REJECT	Contains rejected data results.
Output Card 7	DD:REJLOG	Contains validation error messages.
Output Card 8	DD:ACKIN	Contains 997 or 999 EDI found in the inbound EDI messages.
Output Card 9	DD:ACKOUT	Contains 997 or 999 outbound EDI messages.
Output Card 10	DD:TA1IN	Contains TA1 found in the inbound EDI message.
Output Card 11	DD:TA1OUT	Contains the TA1 outbound EDI message.
Output Card 12	DD:EDILOG	Fatal Error with framework.
Output Card 13	DD:JSONOUT	Optional JSON log report.x
	DD:JSONREJ	

#### Card Output:

Card output can be identified to override each output card which is defined as SINK.

```
<RuntimeInfo>
 <!-- This section will send output to map output cards -->
 <CardOutput/>
```

The z/OS JCL for a map execution provides output card overrides via the -OFx parameter. Any SINK output card may use the execution parameter override in the JCL.

## Installing the shared exit library component on z/OS

After installing the Supply Chain EDI Component on a Windows workstation, perform the following steps for each z/OS execution platform. This is done prior to porting and executing the Supply Chain X12 Compliance Check application on that platform.

## Procedure

1. Transfer the load library module file install\_dir/packs/supplychain\_edi\_vn.n.n/platform\_support/zos/edisvu.loadlib as binary to a z/OS dataset of 80-byte fixed length records named **my.exit.upload**.  
Note: There should be no ASCII to EBCDIC translation and no elimination of carriage return or line feed characters.
2. After the file is on z/OS, issue the TSO Receive command to create a load library. Perform the following steps:

- a. At the prompt, enter **ready**
  - b. Enter the following:
 

```
receive inda ('my.exit.upload')
receive inda ('userid.edisvu.upload')
```

The following messages or similar messages are displayed:

```
INMR901I Dataset EDISVU, UPLOAD from USER on NODENAME
INMR906A Enter restore parameters or 'DELETE' or 'END' +
```
  3. Enter the name of the load library to be created as follows:
 

```
da ('my.exit.loadlib')
da ('userid.edisvu.vxxx.loadlib')
```

Messages indicating that the load library is being created are displayed.
4. In the JCL, which is responsible for executing the Supply Chain X12 Compliance Check application, include a STEPLIB for the created load library that appears before the IBM Transformation Extender Core Install loadlib.
- Example**
- ```
//STEPLIB DD DSN=MY.exit.LOADLIB,DISP=SHR
// DD DSN=MY.ITX.INSTALL.LOADLIB,DISP=SHR
```

Installing the shared exit library component on UNIX

After installing the Supply Chain EDI Component on a windows workstation, perform the following steps for each UNIX execution platform. This is done prior to porting and executing the Supply Chain X12 Compliance Check application on that platform.

Procedure

1. Copy the UNIX shared library object edisvu.so or edisvu.sl for the Supply Chain X12 structure validation utility from `install_dir/packs/Supply Chain/platform_support/os`
- Where os indicates the operating system folder; e.g., aix, zos, etc. to the directory `install_dir/libs` on the target machine.
- Note: The read and execute permissions are set on the shared library.
2. If the setup script in the IBM Transformation Extender installation library is not run by the profile of the user, then the `edisvu.sl` or `edisvu.so` file should be copied to the location on the target machine that contains the compiled compliance check application maps.

File naming conventions

This section describes naming conventions that the X12, RAIL, and EDIFACT compliance check maps use to write the files when the file output option is configured.

In this documentation, some notations are used to represent values that are determined during map execution. The actual values are set as follows:

- `fileDirectory` the value from the *FileDirectory* element of the configuration
- `parentUid` the unique ID that identifies the interchange, from input card 2

The default setting is `NoUniqueID` which indicates date-time `parentUid`

The directories must be created before the map is run, or an error is returned by the File adapter.

- [X12 and RAIL file naming conventions](#)
The following show the file naming conventions for X12 and RAIL data.
- [EDIFACT file naming conventions](#)
The following show the file naming conventions for EDIFACT data.

X12 and RAIL file naming conventions

The following show the file naming conventions for X12 and RAIL data.

Note: In the following descriptions, UNIX style forward slashes (/) are used for the path separator in the Format description, and Windows style backslashes (\) are used in the examples. The map uses the appropriate path separator, based on platform.

- [Valid, accepted transactions](#)
The format and an example of valid, accepted transactions is shown here:
- [Invalid, rejected data](#)
The format and an example of invalid rejected data is shown here:
- [Generated, outbound, functional acknowledgments](#)
The format and an example of generated, outbound TA1 acknowledgments is shown here:
- [Generated, outbound, TA1 acknowledgments](#)
The format and an example of generated, outbound, received functional acknowledgments is shown here:
- [Inbound, received, functional acknowledgments](#)
The format and an example of inbound, received functional acknowledgments is shown here:
- [Inbound, received, TA1 acknowledgments](#)
The format of inbound, received TA1 acknowledgments is shown here:

Valid, accepted transactions

The format and an example of valid, accepted transactions is shown here:

| |
|---|
| Format: |
| <i>fileDirectory/x12/parentUid_x12.in</i> |
| <i>fileDirectory/rail/parentUid_rail.in</i> |
| Example: |
| c:\output\x12\1234567890_x12.in |
| c:\output\rail\20210226-154218_rail.in |

Invalid, rejected data

The format and an example of invalid rejected data is shown here:

| |
|--|
| Format for data file: |
| <i>fileDirectory/x12/reject/parentUid_x12.reject</i> |
| <i>fileDirectory/rail/reject/parentUid_rail.reject</i> |
| Format for error log file: |
| <i>fileDirectory/x12/reject/parentUid_x12.log</i> |
| <i>fileDirectory/rail/reject/parentUid_rail.log</i> |
| Example: |
| c:\output\x12\reject\20210226-154218_x12.reject |
| c:\output\x12\reject\20210226-154218_x12.log |
| c:\output\rail\reject\1234567890_rail.reject |
| c:\output\rail\reject\1234567890_rail.log |

Generated, outbound, functional acknowledgments

| |
|--|
| Format: |
| <i>fileDirectory/fa/outbound/parentUid_ack.out</i> |
| |
| Example: |
| c:\output\fa\outbound\20210226-154218_ack.out |
| |

Generated, outbound, TA1 acknowledgments

The format and an example of generated, outbound TA1 acknowledgments is shown here:

| |
|---|
| Format: |
| <i>fileDirectory/ta1/outbound/parentUid_ta1.out</i> |
| |
| Example: |
| c:\output\ta1\outbound\20210226-154218_ta1.out |
| |

Inbound, received, functional acknowledgments

The format and an example of inbound, received functional acknowledgments is shown here:

| |
|--|
| Format: |
| <i>fileDirectory/fa/inbound/parentUid_ack.in</i> |
| |
| Example: |
| c:\output\fa\inbound\20210226-154218_ack.in |
| |

Inbound, received, TA1 acknowledgments

The format of inbound, received TA1 acknowledgments is shown here:

| |
|---|
| Format: |
| <code>fileDirectory/ta1/inbound/parentUid_ta1.in</code> |
| |
| Example: |
| <code>c:\output\ta1\inbound\20210226-154218_ta1.in</code> |

EDIFACT file naming conventions

The following show the file naming conventions for EDIFACT data.

Note: In the following descriptions, the UNIX style forward slash (/) is used for the path separator in the Format description, and the Windows style backslash (\) is used in the examples. The map uses the appropriate path separator, based on platform.

- [Valid, accepted messages](#)
- [Invalid, rejected data](#)
- [Generated, outbound, CONTRL acknowledgments](#)
- [Inbound, received, CONTRL acknowledgments](#)

Valid, accepted messages

| |
|---|
| Format: |
| <code>fileDirectory/edifact/parentUid_edf.in</code> |
| |
| Example: |
| <code>c:\output\edifact\1234567890_edf.in</code> |

Invalid, rejected data

| |
|--|
| Format: |
| <code>fileDirectory/edifact/reject/parentUid_edf.reject</code> |
| |
| Format for error log file: |
| <code>fileDirectory/edifact/reject/parentUid_edf.log</code> |
| |
| Example: |
| <code>c:\output\edifact\reject\1234567890_edf.reject</code> |
| <code>c:\output\edifact\reject\1234567890_edf.log</code> |

Generated, outbound, CONTRL acknowledgments

| |
|--|
| Format: |
| <code>fileDirectory/fa/outbound/parentUid_ack.out</code> |
| |
| Example: |
| <code>c:\output\fa\outbound\1234567890_ack.out</code> |

Inbound, received, CONTRL acknowledgments

| |
|--|
| Format: |
| <code>fileDirectory/fa/inbound/parentUid_ack.in</code> |
| |
| Example: |
| <code>c:\output\fa\inbound\1234567890_ack.in</code> |

Performance tuning

Performance can be affected by many factors. The following information is related to map execution and may improve performance.

- Reduce the size of the structure validation file.
Review Card 7: structure detail file.
- With z/os processing, suppress the run map log.
Review Card 8: system information file.
- Skip the initial compliance processing step with SkipTTValidation setting.
Review the Configuration information details section.
- Workspace and page overrides.
Review the Configuration information details section and the following information.

The best configuration settings for the map RUN calls for the schema validation map, and the segment validation map, depend on the size of the transactions to be processed by the compliance check maps. The default setting uses work files that are in memory, and is best for small to medium size transactions. The transaction size at which performance improves by using a work file on disk, depends upon the amount of memory available after other applications and services are taken into account.

The recent compliance check maps use the setting in the configuration files without map modifications. You can review the **WorkSpace** configuration setting. However, older versions of the compliance check maps may not be using the configuration setting. Therefore, it may require the following map modifications.

- [Override schema validation maps](#)
- [Override segment detail validation maps](#)

Override schema validation maps

To override the setting for the schema validation maps, add the setting for map work files ("‐WU") to the rule where the schema validation maps are called:

- For X12 and RAIL data, update the ValidateTrxDetail functional map to add the setting in the rule defined for element
TTMapStatusTTValidate Element:TTValCheck:TrxDetailCheck
- For EDIFACT, update the ValidateMsgDetail functional map to add the setting to the rule defined for element
TTMapStatus TTValidate element:TTValCheck:MsgDetailCheck

The updated rule might look similar to the following rule shown here (the updated option is underlined):

```
= VALID(
RUN( TTMapName TTValidate Element:TTValCheck:TrxDetailCheck,
// pass data using ECHOIN if < 1 MB (approx), otherwise use HANDLEIN
IF (SIZE(TTMapData TTValidate Element:TTValCheck:TrxDetailCheck) < 1000000,
ECHOIN( 1, TTMapData TTValidate Element:TTValCheck:TrxDetailCheck ),
" -WU " + HANDLEIN( 1, TTMapData TTValidate Element:TTValCheck:TrxDetailCheck )
)
+ " -OE1"
),
"MAPFAIL|" +TTMapName TTValidate Element:TTValCheck:TrxDetailCheck + "|" +
LASTERRORCODE() + "/" + LASTERRORMSG() )
```

Override segment detail validation maps

To override the setting on the segment detail validation maps, change the following settings:

In the RunAuditEltCheck functional map, add the setting for running map work files: ‐WU in the Data_Audit_Result AuditLog Element:AuditEltCheck rule.

The updated rule looks similar to the following (updated option is underlined):

```
= VALID(
RUN ( SegMapName AuditLog Element:AuditEltCheck,
"-WU -ADM " +
ECHOIN ( 1 , SegMapData AuditLog Element:AuditEltCheck ),

"MAPFAIL|" +SegMapName AuditLog Element:AuditEltCheck + "|" +
LASTERRORCODE() + "/" + LASTERRORMSG()
)
```

Debugging

Logs are created by the compliance check maps to help with problem determination.

This section describes the logs and offers help to resolve some of the more common errors.

- [Compliance check logs](#)

The compliance check logs are written for any EDI interchanges that fail validation, and include information about any errors that were found. If there are no errors, then no compliance check logs are written.

- **Framework log**

If the compliance check maps are not providing the expected results, the primary problem determination tool is the framework log. This log is an XML-style log that is written to the map directory if certain severe errors occur, or if the trace input option is enabled. This log includes information about each step that was performed by the compliance check map. For large EDI interchanges, the framework log can become large.

- **Troubleshooting tips**

These troubleshooting tips are intended as suggestions to help resolve some of the more frequently encountered setup errors.

Compliance check logs

The compliance check logs are written for any EDI interchanges that fail validation, and include information about any errors that were found. If there are no errors, then no compliance check logs are written.

The compliance check logs are written to the following locations:

- File output – The x12/reject, rail/reject, or edifact/reject directory under the directory that is specified by the <FileDirectory> element in the configuration.
- Queue output – The queue that is specified by the <X12_RejectLogQueue>, <RAIL_RejectLogQueue>, or <EDIFACT_RejectLogQueue> element in the configuration.
- Wire output – The wire terminal that is specified by the <X12_RejectLogTerminal>, <RAIL_RejectLogTerminal>, or the <EDIFACT_RejectLogTerminal>.

The compliance check logs are intended to be read by humans. However, the format is also defined in the ccx12.mtt and ccedf.mtt schemas (type ErrorLog ErrorHandling EDI). You can use these schemas to create maps that convert the logs to other formats such as XML or HTML, or to do other automated processing with the logs.

If you do not want the compliance check log to be generated, you can set the configuration parameters to disable it. If the log is disabled, you must look at any generated acknowledgments to determine the errors. If the log is disabled, it can make it difficult to determine problems. For example, in cases where no acknowledgment is generated, no acknowledgment is requested or if the EDI data cannot be parsed.

Also, the compliance check log often includes additional information that is not included in the functional acknowledgment. To disable the creation of the compliance check log, set the CreateRejectLog element in the configuration to F (False). See [CreateRejectLog](#).

Framework log

If the compliance check maps are not providing the expected results, the primary problem determination tool is the framework log. This log is an XML-style log that is written to the map directory if certain severe errors occur, or if the trace input option is enabled. This log includes information about each step that was performed by the compliance check map. For large EDI interchanges, the framework log can become large.

To enable the trace option, set input card 9 to one of the following values:

- T – Trace log is written to file edilog.xml.
- TU – Trace log is written to a unique file, which is based on the unique id that is specified in input card 2. If no unique id is specified, an id is generated based on the current time stamp. The file name is in the format, edilog_uniqueid.xml

Note: If you specify the CardOutput element in your configuration, then the framework log is written to output card 12 instead of to the edilog.xml or edilog_uniqueid.xml file.

For example, to enable this trace you can specify the command line option:

-IE9 T

Note: Use caution when you use the TU option. The option can cause many log files to be generated, since a new log is created for each unique id. This log is intended to be read by a person, and in most cases an editor or viewer that can interpret XML data is useful to help read the log. However, in some cases this log might not be strictly well-formed XML - particularly if the EDI data contains XML control characters, such as, <, >, and &. To reduce complexity and processing overhead, these characters are not escaped, so there might be cases where XML processes do not handle these logs.

Troubleshooting tips

These troubleshooting tips are intended as suggestions to help resolve some of the more frequently encountered setup errors.

Issue: Map returns “One or more inputs was invalid” error

Troubleshooting: The compliance check maps never return the “One or more inputs was invalid” error for bad EDI data. If the EDI data is bad, the maps reject the data.

An error is then written to the compliance check log to indicate that it cannot be parsed, as X12, RAIL, or EDIFACT data.

The “One or more inputs was invalid” error typically indicates an error in one of the other input files. The error might be in the configuration file or one of the other text files that is used as input to the map.

Use the -TI option on the ccx12, ccrail or ccedf map to generate an input trace, which can help you identify which input is considered invalid.

Make sure that your configuration file is valid, and conforms to the XSD. See the sample configuration files in the data directory for valid examples.

If you are running on a system other than a Windows system, make sure that all files are copied correctly.

Issue: Map returns “Input valid but unknown data found” message

Troubleshooting: This issue can occur because there are multiple interchanges in the input EDI data (input card 1), or other data that follows the EDI interchange. The EDI compliance check maps support one interchange at a time in the input EDI data.

The first interchange of the input does get processed in this case. If there is only one interchange in your EDI input, use the -TI option on the ccx12, ccrail, or ccedf map to generate an input trace. Then, use the trace file to help you identify which input has unknown data at the end.

Issue: Map returns “FAIL function aborted map” error

Troubleshooting: This error occurs if there is a severe error – typically the map is not able to write to the file, queue, or wire terminal. When this type of failure occurs, a framework log edilog.xml is written to the map directory. For these failures, the log is written whether the trace was enabled or not.

Search the framework log for the string "<FrameworkErrorOccurred>". Immediately before this string there is usually some information about the error that occurred, such as an error writing to a specific file, queue, or wire terminal.

When you are writing to file output, the directories must already be created (and have write access) before you can run the map. The map does not create them. Also, be aware that path names are relative to the map directory, not necessarily to the directory you were in when you ran the command.

Issue: Schema validation map failed (E9002/E9003 error in compliance check log)

Troubleshooting: Make sure the schema validation map that is listed in the error message was compiled. If you are running on a system other than Windows, make sure that it was copied to the system as binary.

Issue: Segment detail validation map failed (E9015 error in compliance check log)

Troubleshooting: Make sure the segment detail validation map that is listed in the error message was compiled. If you are running on a system other than Windows, make sure that it was copied to the system as binary.

Issue: Exit/DLL call failed (E9010 error in compliance check log)

Troubleshooting: Make sure that the DLL/shared library used for the exit call is copied to the target system (as binary), and that the edicc.ini file has the correct path to the exit library and verify the exit library exists in the path. The default path using Design Studio is relative to the map location. The path using Design Server is the ITX install location. For z/os the exit library should be in the JCL steplib.

Pack for Supply Chain EDI samples

The Pack for Supply Chain EDI includes sample deenvelope and envelope tests for EDIFACT, and X12.

EDIFACT deenvelope and envelope tests

The EDIFACT sample tests are based on the batchfile.txt file that is in the *install_dir\examples\edi\edifact* directory.
The samples run these EDIFACT tests:

- A deenvelope test that uses good data input file, edifactin-multi.txt.
- A deenvelope test that uses bad data input file, edifactin-multi-bad.txt.
- An envelope test that uses good data input file poin-multi.txt.

The EDIFACT sample tests are based on the batchfile.txt file that installs with the imported project. The samples run these EDIFACT tests:

- A deenvelope test that uses good data input file, edifactin-multi.txt.
- A deenvelope test that uses bad data input file, edifactin-multi-bad.txt.
- An envelope test that uses good data input file poin-multi.txt.

X12 deenvelope and envelope tests

The X12 sample tests are based on the batchfile.txt file that is in the *install_dir\examples\edi\x12* directory.

The samples run these X12 tests:

- A deenvelope test that uses good data input file, x12in-multi.txt.
- A deenvelope test that uses bad data input file, x12in-multi-bad.txt.
- An envelope test that uses good data input file, poin-multi.txt.

The X12 sample tests are based on the batchfile.txt file that installs with the imported project. The samples run these X12 tests:

- A deenvelope test that uses good data input file, x12in-multi.txt
- A deenvelope test that uses bad data input file, x12in-multi-bad.txt
- An envelope test that uses good data input file, poin-multi.txt

Trading Manager Overview

Trading Manager is a Business to Business (B2B) management environment that uses the IBM WebSphere Transformation Extender to integrate trading partner transactions with enterprise applications.

Installation and Configuration

The documentation provided here describes how to install and configure Trading Manager.

Note the following before installing Trading Manager:

- the database associated with Trading Manager can reside on the same machine as the IBM WebSphere Transformation Extender Packs with Launcher, or on another machine that is network-accessible
 - the Design Studio must be installed prior to installing Trading Manager
- [**Overview**](#)
Trading Manager is a Business to Business (B2B) management environment that uses the IBM WebSphere Transformation Extender to integrate trading partner transactions with enterprise applications.
 - [**Installation requirements**](#)
 - [**Installing Trading Manager**](#)
 - [**Configuring Trading Manager**](#)
 - [**Installing and using Message Manager**](#)
 - [**Support for proprietary data types**](#)
 - [**Using alias files in maps**](#)
 - [**Checklists**](#)
 - [**Implementing custom X12 versions**](#)
 - [**TPEC to Trading Manager conversion**](#)

Overview

Trading Manager is a Business to Business (B2B) management environment that uses the IBM WebSphere Transformation Extender to integrate trading partner transactions with enterprise applications.

It includes two components: Partner Manager, a graphical administration component, and Message Manager, a runtime subsystem that uses trading profiles maintained by Partner Manager to automate document exchange among EDI partners. Electronic commerce data flows from external sources to internal destinations and from internal sources to external destinations, and possibly to other internal organizations.

- [**About the Trading Manager Environment**](#)
- [**IBM WebSphere Transformation Extender Packs for EDI**](#)
- [**IBM WebSphere Transformation Extender with Launcher**](#)

About the Trading Manager Environment

The Trading Manager B2B environment consists of the following components. Each component is described in detail in the following sections:

- Partner Manager
 - Message Manager
 - Design Studio
 - Integration Flow Designer
 - Launcher
 - Resource Adapters
- [**Partner Manager**](#)
 - [**Message Manager**](#)
 - [**Design Studio**](#)
 - [**Integration Flow Designer**](#)
 - [**Launcher**](#)
 - [**Resource Adapters**](#)

Partner Manager

Partner Manager is a graphical user interface (GUI) application used to create a comprehensive database of an organization's electronic commerce information and to manage its e-commerce activity. Partner Manager is used to define your trading relationships with your e-commerce partners. It also maintains the information for each trading partner including address information, contact persons, and the individual organizational entities with which you trade business information. Partner Manager allows for multiple databases to be used, or multiple copies may also access a single database.

Message Manager

Message Manager consists of a pre-defined system of maps specifically designed to process electronic commerce data. The executable maps are used for validation and routing of data.

Design Studio

The Design Studio is used to create maps that perform enterprise application integration by converting data from one form to another (typically to integrate e-commerce data with applications) and routing that information from its source to its target. The Design Studio features include the Integration Flow Designer, Map Designer, Type Designer, Metadata Type Importers for XML, EDI, and DBMS applications, and prebuilt adapters for messaging and transporting.

Integration Flow Designer

The Integration Flow Designer is used to define and organize maps into logical groups or comprehensive workflow systems. The Integration Flow Designer is used by a system analyst or system administrator responsible for linking internal applications with the Message Manager components provided as part of Trading Manager. Using the information defined in Partner Manager, the Integration Flow Designer is used to tailor the Message Manager for your organization's needs and to define run time settings for execution on the Launcher. It is a component of the Design Studio.

Launcher

The Launcher is used to execute the Message Manager maps. It is installed from the IBM WebSphere Transformation Extender. The Launcher can run on a single server or across multiple servers to distribute processing. Run-time event triggers are configured graphically using the Integration Flow Designer. During execution, there is seamless data flow between system components.

Resource Adapters

Resource Adapters provide connectivity to a wide array of types of data sources and targets, such as databases, messaging systems, FTP, or e-mail.

IBM WebSphere Transformation Extender Packs for EDI

The IBM WebSphere Transformation Extender Packs for EDI are collections of type trees representing the different versions of an EDI standard. Integration packages for the following EDI standards are available:

- X12
- EDIFACT
- TRADACOMS
- ODETTE
- EANCOM

IBM WebSphere Transformation Extender with Launcher

The IBM WebSphere Transformation Extended Edition provides the server with components to execute the IBM WebSphere Transformation Extender with Launcher. It includes the Launcher, the Management Tools (for Windows 7, Windows 2008, and Windows XP only), and the Command Server.

The Launcher must be installed before deploying Message Manager. The IBM WebSphere Transformation Extender Packs with Launcher helps you automate the execution of system maps. This map execution is based on time or source events, in contrast to the Command Server that executes maps according to commands that you supply involving only one map at a time.

Installation requirements

These instructions describe the requirements for installing Trading Manager. The database associated with Trading Manager can reside on the same machine as the IBM WebSphere Transformation Extender Packs with Launcher, or on another network-accessible machine.

The Design Studio must be installed prior to installing Trading Manager.

- [Minimum system requirements](#)
- [Prerequisites](#)
- [Installation checklists](#)
- [Launcher](#)

Minimum system requirements

The following documentation describes the requirements for Partner Manager, as well as the Windows 7, Windows 2008, Windows XP, UNIX, and Linux platforms.

- [System requirements for Partner Manager](#)

- [UNIX platform](#)

System requirements for Partner Manager

The minimum requirements to install and run Partner Manager are:

- Microsoft Windows 7, Windows 8, or Windows XP
- 512 MB RAM, however, 1 GB RAM is recommended

UNIX platform

The Launcher for UNIX (or Linux) is a multi-threaded daemon that runs on UNIX platforms.

The **cmsvu** shared libraries for each UNIX operating system are installed in the *install_dir\tmgr_vn.n\platform_support* directory. Before starting Message Manager on the server, ensure that the correct **cmsvu** file is copied to the **libs** directory located in the IBM install directory on the UNIX server.

Refer to the Launcher documentation for UNIX installation and configuration information.

Prerequisites

- **E-mail** - Ensure that connectivity and access to e-mail service and a user account for Message Manager is in place. Message Manager supports MS Mail, MS Exchange, Lotus CCMail, Lotus Notes and SMTP/POP3. Message Manager uses the e-mail service to send dynamic problem alerts to individuals responsible for managing the electronic commerce program.
- **Trading partner requirements** - To analyze your requirements, consider the following:
 - Number of trading partners to be supported.
 - Versions of standards and EDI documents (810, 850, 997, and others) to be exchanged.
 - Application format specifications for any internal applications interfacing with the EDI system.
 - Communication requirements for transporting data between application systems and the Trading Manager server. Requirements include the type of communications to use (e-mail, FTP, messaging product, and others), userids, passwords, naming conventions, and responsible parties.

Installation checklists

Before installing Trading Manager, please complete the installation checklist for your database software. See the Checklists documentation for more information.

Launcher

The Launcher is installed as part of the IBM WebSphere Transformation Extender. It should already be installed. If it is not installed, refer to the documentation for installation instructions.

Installing Trading Manager

About this task

These instructions describe the installation of the Trading Manager software components.

If you have an earlier version of Trading Manager installed, delete the **MessageManager.msl** file located in the *install_dir\systems* directory. If you do not delete this **.msl** file, the system will fail. The new default filename is **msg_mgr.msl**.

Refer to the Launcher documentation for installation information.

After the Launcher and Partner Manager are installed, you can configure the system as described in the documentation titled "[Configuring Trading Manager](#)".

Note: If you are a HIPAA customer, refer to: [Using Trading Manager with HIPAA](#).

Note: If you are converting from a prior release, refer to the readme file, **readme_tmgr.txt**, that is shipped with the Trading Manager application, for conversion instructions. This readme is located in the *install_dir* where *install_dir* represents the location of the core IBM WebSphere Transformation Extender product.

- [Determining the Trading Manager configuration](#)
- [Installation checklists](#)
- [Installing components](#)
- [Removing Trading Manager components](#)
- [Using Trading Manager with HIPAA](#)

Determining the Trading Manager configuration

Trading Manager enables you to manage your installation depending on your e-commerce processing needs. You can install all Trading Manager components on a single machine, or assign Trading Manager functions to different and dedicated machines.

- [Installing Trading Manager on a single workstation](#)
- [Installing Trading Manager on separate workstations](#)
- [Installing only Partner Manager](#)

On individual workstations using only Partner Manager to create information about trading and application partners, post offices, etc., only the Partner Manager application need be installed.

Installing Trading Manager on a single workstation

On a single workstation running all Trading Manager client/server components including the database, the following components must be installed:

- Server
- Partner Manager
- Message Manager
- Database Scripts

Note: Ensure that you have installed the Design Studio and IBM WebSphere Transformation Extender first.

Installing Trading Manager on separate workstations

If you are installing Trading Manager on separate workstations and servers in a client/server configuration, install these components:

- **Client workstation:** Partner Manager and Message Manager
- **Server machine where the Launcher is also installed:** this installation provides the Launcher and directory structure required for Message Manager map execution.
- **Database Server:** Database scripts

Installing only Partner Manager

On individual workstations using only Partner Manager to create information about trading and application partners, post offices, etc., only the Partner Manager application need be installed.

Installation checklists

Before installing the Trading Manager, please complete the installation checklist for your database software. See [Checklists](#).

Installing components

About this task

Before you begin installation, close any other programs running on your computer. Having other programs running may cause installation errors.

Before installing Trading Manager, the Design Studio must be already installed. If you have not done so already, perform this installation first by inserting the Design Studio CD into the CD-ROM drive and following the installation instructions.

You can optionally install the IBM WebSphere Transformation Extender for system execution on the local Windows 7, Windows 2008, or Windows XP client workstation using the Launcher. It is available on a separate IBM WebSphere Transformation Extender CD that installs the Launcher and related tools.

- [Installing Trading Manager on Windows](#)
- [Installing Message Manager components of Trading Manager on UNIX](#)

To install Trading Manager on HP-UX, Sun Solaris, or IBM RS/6000 AIX, Linux, or under UNIX System Services on z/OS, perform the following steps:

Installing Trading Manager on Windows

About this task

To install Trading Manager components on Windows, perform the following steps:

Procedure

1. Insert the Trading Manager CD into your CD-ROM drive.

2. If the program does not automatically start, go to the Start menu and select Run.
3. Enter **d:\install.exe** and click OK (where **d:** indicates the CD-ROM drive).
4. Follow the setup instructions to complete installation.

Installing Message Manager components of Trading Manager on UNIX

To install Trading Manager on HP-UX, Sun Solaris, or IBM RS/6000 AIX, Linux, or under UNIX System Services on z/OS, perform the following steps:

Procedure

1. Copy the **tmsinstall.ksh** file to the IBM WebSphere Transformation Extender install directory on the target machine.
2. Running in a korn shell, execute **tmsinstall.ksh** on the target machine.
Refer to the Launcher documentation for more information.
3. Navigate to the platform folder and copy the shared libraries to the **libs** directory under the IBM WebSphere Transformation Extender install directory on your UNIX machine.

Removing Trading Manager components

About this task

To remove Trading Manager components, perform the following steps:

Procedure

1. From the Start menu, select Settings > Control Panel > Add/Remove Programs.
2. Select the product component in the list provided.
3. Click Add/Remove.

Using Trading Manager with HIPAA

The **hipaa_x12_qualifier.dat** and **hipaa_x12_structure.dat** files must be imported into Trading Manager before any HIPAA trade links can be created. For a complete description of this activity, refer to the IBM WebSphere Transformation Extender Pack for HIPAA EDI documentation.

Configuring Trading Manager

These instructions describe how to configure Trading Manager with the databases, the and Partner Manager.

How you configure Trading Manager depends upon:

- The type of database your system is using: Microsoft Access, Microsoft SQL Server, Oracle, Sybase, or DB2
- Where the database is located
- Which applications interface with your system
- Whether you have a standalone or multi-machine configuration
- [Before you begin](#)
- [Creating the Trading Manager Database](#)
- [Determining the ODBC driver](#)
- [Defining an ODBC data source name](#)
- [Resolving ODBC difficulties](#)
- [Configuring Partner Manager](#)
- [Import structure validation](#)

Before you begin

Before configuring Trading Manager, ensure that:

- The database is created and ready for installation scripts
- Trading Manager components have been installed

You may then configure the Trading Manager database, Partner Manager settings, and Message Manager. The general configuration steps are:

- Create the Trading Manager database.
- Run the SQL scripts for IBM DB2, Microsoft SQL Server, Oracle, or Sybase to create the Trading Manager database objects.
- Configure Partner Manager to point to the Trading Manager database.

- Configure Message Manager to point to the Trading Manager database. These instructions describe how to change the database type in the *install_dir\tmgr_vn.n\mmgr\update.mdq* file to match the database type for your installation.
- [SQL scripts installed by Trading Manager](#)

SQL scripts installed by Trading Manager

Trading Manager installs the following SQL script files in the Trading Manager directory *install_dir\tmgr_vn.n\pmgr\sql*. The following descriptions apply to the scripts for Oracle, Microsoft SQL Server, Sybase, and IBM DB2.

- Installation script (***instnn.sql**): creates the tables, indexes, stored procedures, and views used by Partner Manager.
- Database load script (***loadnn.sql**): loads the database with data such as Folders, Version tables, Transaction Sets, and Functional IDs.
- Drop script (***dropnn.sql**): deletes all Partner Manager tables and data in the database.
- Update scripts (***update82to821.sql**): these upgrade your database from the 8.2 release. For customers converting from 7.8 and earlier, refer to the product readme for use of these files. These files are also used on a new customer installation of Trading Manager 8.2.1 and are described below.

| Database | Script |
|----------------------|--|
| Oracle | orainstnn.sql
oraloadnn.sql
oradropnn.sql |
| Microsoft SQL Server | sqlinstnn.sql
sqlloadnn.sql
sqldropnn.sql |
| Sybase | sybaseinstnn.sql
sybasedropnn.sql
sybaseloadnn.sql
sybaseloadnn_2.sql |
| IBM DB2 | db2instnn.sql
db2loadnn.sql
db2dropnn.sql |

Creating the Trading Manager Database

About this task

See the following topics to create a database.

- ["Using an Access Database"](#)
- ["Creating a Microsoft SQL Server Database"](#)
- ["Creating an Oracle Database"](#)
- ["Creating a Sybase Database"](#)
- ["Creating an IBM DB2 Database"](#)
- [Using an Access database](#)
- [Creating a Microsoft SQL Server Database](#)
- [Creating an Oracle Database](#)
- [Creating a Sybase Database](#)
- [Creating an IBM DB2 database](#)
- [IBM DB2 stored procedures](#)

The following procedures install eight Stored Procedures and one User Defined Function (UDF) into your database. These procedures should be performed by your DB2 DBA.

Using an Access database

An Access database (**tmgrnn.mdb**) is shipped with the Trading Manager client. The Access database is installed in the *install_dir\tmgr_vn.n\pmgr* directory. This database can be placed locally or on a server. If located on a server, it will be necessary to map a drive to that location before configuring the ODBC DSN Connection. See the documentation titled [Installing Trading Manager](#) for further information.

Due to performance reasons, it is recommended that you use the Access database for test or training purposes only; it should not be used in a production environment.

Creating a Microsoft SQL Server Database

About this task

When using Microsoft SQL Server, create the database data and log devices and then create the Trading Manager database using these devices. The name of the database you just created cannot contain spaces or special characters. Minimum disk space for the database data device is 100 MB and a recommended size is 500 MB. Minimum disk space for the database log device is 20 MB and a recommended size is 100 MB.

If any part of this procedure needs to be repeated, use **sqldropnn.sql** to delete all Partner Manager tables and data in the database.

Use caution: **sqldropnn.sql** cannot be reversed and should be used *only* if these steps must be repeated.

To run Microsoft SQL Server scripts:

Procedure

1. Start your **SQL Query** tool and connect to the Trading Manager database you just created.
2. See the Trading Manager release notes for the scripts to run.

What to do next

Note: These scripts do not contain a **COMMIT** statement. You must manually issue a commit if your database environment is not set to **auto-commit**.

Creating an Oracle Database

About this task

When using Oracle Server, use the default ORCL instance, or create an instance specifically for Trading Manager. The minimum disk space required for the instance is 100 MB. If you expect heavy transaction volume, use the auto-expand feature of Oracle.

If any part of this step needs to be repeated, use **oradropnn.sql** to delete all Partner Manager tables and data in the database. Use caution as **oradropnn.sql** cannot be reversed and should be used only if these steps must be repeated.

To run Oracle SQL scripts, perform the following steps:

Procedure

1. Log on to the desired Oracle instance in SQL*PLUS.
2. Create a spool output by entering the following (you can change this file name, or if available on your version of SQL*PLUS, use the File Spool option to set it):

```
spool C:\temp\inst.log
```

3. Process **orainst82.sql** (the path for the SQL files are in the directory where you installed Partner Manager, for example):

```
install_dir\mgr_vn.n\pmgr\sql\orainstnn.sql
```

4. Execute **oraupdate82to821.sql**.

5. When complete, shut off spooling with **spool off**, or use the File Spool option, if available.

6. Check the spool file created in Step 2 for errors. Resolve any errors before proceeding.

7. If there are no errors, enter **commit** in SQL*PLUS to save the changes.

8. Create another spool output by entering:

```
spool C:\temp\load.log
```

9. Process **oraloadnn.sql**:

```
install_dir\mgr_vn.n\pmgr\sql\oraloadnn.sql
```

10. When complete, shut off spooling with **spool off** or use the File Spool option, if available.

11. Check the spool file created in Step 8 for errors. Resolve any errors before proceeding.

12. If there are no errors, enter **commit** into SQL*PLUS to save the changes.

13. Exit SQL*PLUS.

Creating a Sybase Database

About this task

When using Sybase, create the database data and log devices and then create the Trading Manager database using these devices. The minimum disk space for the database data device is 100 MB. The minimum disk space for the database log device is 20 MB.

Note: If any part of this step needs to be repeated, a file named **sybasedropnn.sql** can drop all the tables created by **sybaseinstnn.sql**. Use caution, as the **sybasedropnn.sql** routine deletes all Partner Manager tables and data and cannot be reversed. It should only be used if these steps need to be repeated.

To create the Sybase Database, perform the following steps:

Procedure

1. Open the SQL editing tool.
2. Select **File > Open** and open the installation script from the default Trading Manager installation directory. If you changed the default directory during installation, use the correct directory for your install. Select the **sybaseinst82.sql** script.
3. Execute the **sybaseinst82.sql**.
4. Execute the installation script and wait for the last output message to be displayed. The message will read: **Install Complete**.
Note: On some versions of Sybase, you will receive the following warning: Row size could exceed row size limit, which is xxxx bytes (where xxxx can vary). Ignore this warning.
5. Check the output for errors; resolve any errors before proceeding.
6. Select **File > Open** and open the file **sqlupdate82to821.sql**.

7. Execute **sqlupdate82to821.sql** and wait for completion.
 8. Select File,>Open and open the file **sybaseunloadnn.sql**.
 9. Execute **sybaseunloadnn.sql** and wait for it to complete.
 10. Check the output for any errors.
 11. Select File,>Open and open the file **sybaseunloadnn_2.sql**.
 12. Execute **sybaseunloadnn_2.sql** and wait for it to complete.
 13. Check the output for errors.
- Note: These scripts do not contain a **COMMIT** statement. You must manually issue a commit if your database environment is not set to auto-commit.
-

Creating an IBM DB2 database

About this task

Construct the database using scripts loaded during the Trading Manager installation. The minimum disk space required is 200 MB.

To create the IBM DB2 Database, perform the following steps:

Procedure

1. Contact your Database Administrator (DBA) to get a database name, a user name, and a password.
 2. Verify with the DBA that the autocommit is on. If the autocommit is not on, either issue commits manually, or modify the **.sql** scripts to include the commits.
 3. Open the **db2inst82.sql** file in an editor. Change the first executable line to match your database name and user/password. For example, change **sample** and **db2admin** in the following executable line:
connect to sample user db2admin using db2admin
 4. Change the tablespace name (search and replace in **USERSPACE1**) as directed by your DBA. Trading Manager requires a DB2 tablespace page size of at least 8K; 32K is recommended. Ask your DBA to create such a tablespace, and replace **USERSPACE1** with the name of this tablespace
 5. Save the file.
 6. Using the DB2 Command Window, or the Command Line Processor, execute the new **db2instnn.sql** saved in step 5. Recommended usage is:
db2 -tvfs 'filepath\db2instnn.sql'>'filepath\logfilename'.
The Mainframe database tool '**SPUFI**' may also be used to execute scripts.
Note: Your DBA will instruct you on how to run SQL scripts. The scripts supplied are semi-colon delimited, plain text, SQL statements.
 7. After execution, review the results for errors.
 8. Open the **db2update82to821.sql** file in an editor. Change the first executable line to match your database name and user/password. For example, change **sample** and **db2admin** in the following executable line:
connect to sample user db2admin using db2admin
 9. Change the tablespace name (search and replace in **USERSPACE1**) as directed by your DBA. Trading Manager requires a DB2 tablespace page size of at least 8K; 32K is recommended. Ask your DBA to create such a tablespace, and replace **USERSPACE1** with the name of this tablespace
 10. Save the file.
 11. Using the DB2 Command Window, or the Command Line Processor, execute the new **db2update82to821.sql** saved above. Recommended usage is:
db2 -tvfs 'filepath\db2instnn.sql'>'filepath\logfilename'.
The Mainframe database tool '**SPUFI**' may also be used to execute scripts.
 12. After execution, review the results for errors.
 13. Open **db2loadnn.sql** in an editor and change the connect statement for the database, user name, and password and then save the file.
 14. Using the DB2 Command Window, or the Command Line Processor, execute the new **db2loadnn.sql** saved above.
 15. After execution, review the results for errors.
- If error free, the installation of **.sql** scripts is complete.
-

IBM DB2 stored procedures

The following procedures install eight Stored Procedures and one User Defined Function (UDF) into your database. These procedures should be performed by your DB2 DBA.

Note: DB2 Stored Procedures are supported on DB2 LUW (Linux, Unix and Windows) on versions 9.1 or greater.

Note: If multiple databases are used (for example, test and production databases), repeat these steps for each database.

If the stored procedures were previously installed with DB2 Data Studio, you may have stored procedures already defined, but named with a lowercase letter. If so, delete these stored procedures manually using the DB2 Control Center before continuing with this procedure. Delete the user defined function: **TMGR_TS_FMT**, as well as the following stored procedures:

```
eDF_INCREMENTMSG
eDF_INCREMENTNB
gETSTAMP_INCREMENTST
INCREMENTISA
INCREMENTGS_BY_TP
INCREMENTGS_BY_FID
INCREMENTGS_BY_AP
```

Preparation

Before you begin, make certain that you have the following software: DB2 Command Editor, DB2 Control Center (if required), and DB2 database, version 9.1 or greater. Make sure of the following conditions before installing stored procedures:

- Make certain that no Trading Manager activity is taking place during the installation process. It is suggested that you back up your database.

- Obtain the user id and password for the user that connects to the database in the Trading Manager maps. See [update.mdq](#). That user should be used to create these objects. If that user does not have create permission, it will require that GRANT EXECUTE statements be issued on all objects created here. Contact your DBA for assistance.
 - If the stored procedures and the User Defined Function already exist from a prior version or were created using Developer Workbench, delete them from the Control Center, or with the DROP statements defined later in this procedure.
- [**Installing DB2 stored procedures**](#)
The following procedure installs stored procedures and one User Defined Function (UDF) into your database. These procedures should be performed by your Database Administrator.

Installing DB2 stored procedures

The following procedure installs stored procedures and one User Defined Function (UDF) into your database. These procedures should be performed by your Database Administrator.

Procedure

1. Connect to the database using the Command Editor. Use the same user id and password specified in [update.mdq](#).
2. Set statement terminator to @ in Command Editor (at the bottom of the screen).
3. Set the Command Editor option to **Stop Execution on Error** (Tools > Tools Settings > Command Editor).
4. Open the SQL File, DB2V9TMSP.SQL in an editor and change the connect statement to the user id and password assigned in Trading Manager's [update.mdq](#), as described in the Preparation steps. DB2V9TMSP.SQL is located in the *install_dir/tmgr_dir/pmgr/sql* directory, where *install_dir* indicates the installation location of the core WebSphere Transformation Extender product and *tmgr_dir* indicates the location of Trading Manager.
5. Copy the changed SQL file and paste into the Command Editor and execute. If any errors are reported, contact your Database Administrator and/or IBM support.
6. The SQL file contains a suggested test call that can be run at this point.
7. To use the stored procedures in Trading Manager, set **Use Stored Procedures** in Partner Manager: Utilities > Message Manager Config > Database tab

- [**Copying drop statements**](#)

Copying drop statements

If drop statements are needed you can copy the following into Command Editor with an "@" statement terminator:

```
DROP PROCEDURE GETSTAMP
@
DROP PROCEDURE INCREMENTISA
@
DROP PROCEDURE INCREMENTGS_BY_TP
@
DROP PROCEDURE INCREMENTGS_BY_AP
@
DROP PROCEDURE INCREMENTGS_BY_FID
@
DROP PROCEDURE INCREMENTST
@
DROP PROCEDURE EDF_INCREMENTUNB
@
DROP PROCEDURE EDF_INCREMENTMSG
@
DROP FUNCTION TMGR_TS_FMT
@
```

Determining the ODBC driver

Before you use Partner Manager, you must configure an open database connectivity (ODBC) data source name (DSN) on each client machine that will run Partner Manager. The DSN associates an ODBC driver with your database type. Review the Database Requirements to determine the ODBC driver to select before defining your DSN. Ask your Database Administrator to assist you, then proceed with defining the DSN.

Defining an ODBC data source name

About this task

After you determine the ODBC driver to use, complete the following steps to define the DSN.

To define an ODBC DSN, perform the following steps:

Procedure

1. Select Settings...> Control Panel > Administrative Tools > Data Sources (ODBD) the ODBC Data Source Administrator displays.
2. Select the **System DSN** tab (not **User DSN**).
Selecting **System DSN** lets you create a DSN that is visible to all users; selecting **User DSN** creates a DSN that is only visible to the specified user on a particular machine.
3. Click **Add**.
The Create a New Data Source Name window opens.
4. Select the appropriate driver for your database from the list.
5. Click **Finish**.
The **ODBC Setup** window opens.
6. Enter the driver-specific setup parameters.
The name you assign in the Data Source Name window entry may be any name. However, IBM recommends that you use **Trading_Manager_TEST** (for testing) and **Trading_Manager_PROD** (for production) to simplify support and consulting consistency. The name entered will also be the selection for your DSN when you configure Partner Manager. You may enter an optional description in this window.
7. Complete the window as required by the driver provider.
8. Click **OK**.
9. The name you assigned appears in the **Name** list in the **ODBC Data Source Administrator**.
10. Click **OK**.

Resolving ODBC difficulties

About this task

If you experience difficulties with the ODBC connection when first installing or running Partner Manager, get the latest ODBC drivers from your database vendor. Ensure the driver is for your version of the database and client operating system.

Oracle users: ODBC driver version varied depending upon your Oracle configuration. See the Oracle Metalink article titled "Supported ODBC Configurations"; document ID 66403.1 for the proper Oracle driver number.

Contact your Database Administrator for a password to view this article.

http://metalink.oracle.com/metalink/plsql/ml2_documents.showDocument?p_database_id=NOT&p_id=66403.1

In the event this link is out of date, search MetaLink for "Supported ODBC Configurations".

Database vendors supply free ODBC drivers and they are available for download from their web sites.

Configuring Partner Manager

About this task

The first time that Partner Manager starts, it prompts for the DSN you configured in the documentation titled "[Installing Trading Manager](#)". To configure Partner Manager, perform the following steps:

Procedure

1. Start Partner Manager by selecting Start...>Programs>IBM WebSphere Transformation Extender n.n.>Trading Manager.>Partner Manager.
2. The first time Partner Manager starts, the Partner Manager Database Selection window opens.
3. Select a vendor from the **Database Vendor** drop-down list
4. Select the **Data Source Name** you created in "[Installing Trading Manager](#)" from the **DSN** drop-down list.
5. Enter **User ID** and **Password** (optional).
6. Enter an **Optional Description** if desired. This description appears on the title bar when Partner Manager is running.
7. Click **OK**.

Import structure validation

About this task

For all databases created other than Access, the X12 Structure Validation file must be imported. If you process EDIFACT transactions, the EDIFACT Structure Validation file must be imported as well.

The file need only be imported once per database created.

To import an X12 Structure Validation file

Procedure

1. Start Partner Manager with the DSN created in "[Configuring Partner Manager](#)".
2. Click the **Utilities** icon on the tool bar.
The **Utilities** navigator displays.
3. Click Standards.

4. In the Standards Maintenance window, select X12 Structure Validation: **Import** from the **X12 Standards** drop-down box.
5. Click the **Find File** button and select the **X12_StructureValidation.detl** file, which installed in the *install_dir\tmgr_vn.n\pmgr* directory.
6. Click the **Begin Import** button to import this data.
7. If you process EDIFACT data, repeat steps 4, 5 and 6 using the **EDF_StructureValidation.detl**: **Import** from the **EDIFACT Standards** drop-down box, using the **EDF_StructureValidation.detl** if you process EDIFACT data.

Installing and using Message Manager

These instructions describe how to install and use the Message Manager . The Message Manager is the business logic that provides the routing, archival, standards validation, audit, and alert functions within the Trading Manager. The Message Manager runs on the Launcher and is viewed, implemented, and configured using the Integration Flow Designer. The Message Manager provides a generic electronic commerce framework with EDI-specific functions.

- [Installing Message Manager](#)
- [General configuration and setup](#)
- [Starting Message Manager](#)
- [Message Manager deployment tasks](#)
- [Using Message Manager](#)
- [Message Manager subsystems](#)

Installing Message Manager

About this task

The Message Manager source files are installed as part of the Trading Manager. See "[Installing Trading Manager](#)" for installation instructions.

- [Where to install Message Manager](#)

Where to install Message Manager

About this task

The Message Manager can be installed onto a client workstation or the Trading Manager Launcher machine. The source Message Manager source maps and trees are installed within a valid Message Manager directory structure, such as *install_dir\tmgr_vn.n\mmgr*. There are advantages and disadvantages to each approach, which are discussed below.

- [Client workstation installation](#)
- [IBM WebSphere Transformation Extender Launcher installation](#)

Client workstation installation

Installing the Message Manager on a client workstation is the default option. This option requires connectivity to the IBM WebSphere Transformation Extender Launcher machine and the complete Design Studio environment. There are advantages and disadvantages to this type of installation.

As an advantage, this installation provides a single point of control for the Trading Manager operations. Also, better performance is achieved while running Design Studio as there is no network overhead.

As a disadvantage, it is difficult to share Message Manager responsibilities with other resources. Also, an extra step is required when adding new versions or transactions to the system.

IBM WebSphere Transformation Extender Launcher installation

Installing the Message Manager on the Trading Manager IBM WebSphere Transformation Extender Launcher machine (Windows 7, Windows 2008, or Windows XP platform) is also an alternative. It still requires connectivity between the client workstation of the individual administering the system and the server. This option also assumes the installation of the complete Design Studio environment on that administration client. It does not require the installation of the Design Studio components on the server. There are advantages and disadvantages to this installation.

An advantage of this installation is that it simplifies the sharing of Message Manager responsibilities due to centralized location of the source. It also eliminates the need for an extra copy step when building maps.

As a disadvantage, it may slow performance while running Design Studio clients due to network overhead. Another disadvantage is that standard drive mappings are required if multiple users are supported. For example, each client must map the server machine as Y.

General configuration and setup

About this task

You must complete the following activities before you can begin using Trading Manager:

- Install the Launcher Trading Manager components. See the documentation titled "[Installing Trading Manager](#)" for installation instructions.
 - Complete the configuration. See "[Configuring Trading Manager](#)".
 - Define trading partner information including trading relationships to the Partner Manager database using the Partner Manager client. See the Partner Manager documentation for more information.
-

Starting Message Manager

About this task

You start the Message Manager by opening the Message Manager system definition file **msg_mgr.msd** in the Integration Flow Designer, installed as part of the Design Studio.

To start the Message Manager:

- From the **Start** menu, select Programs->IBM WebSphere Transformation Extender->Trading Manager->Message Manager.

The Integration Flow Designer opens the **msg_mgr.msd** system definition file.

Message Manager deployment tasks

About this task

After you have completed the Trading Manager general configuration and setup steps, you are ready to prepare the Message Manager for deployment. Deploying Message Manager consists of the following activities:

- Define the server types in the **msg_mgr.msd** file.
 - Modify the **msg_mgr.msd** default pathname settings to match your physical environment
 - Deploy Message Manager on UNIX.
 - Modify the email alert settings to match your email environment.
 - Configure the Launcher service startup parameters.
 - Modify the Message Manager database query file for your database type.
 - Define Partners, Post Offices and Trade Links in Partner Manager.
 - Run the EDI Wizard.
 - Enable the **stampandsortFTP**, **stampandsortVAN**, and **stampandsortEmail** maps.
 - [Step 1. Define the server types in the msg_mgr.msd file](#)
 - [Step 2. Modify the msg_mgr.msd default path name settings](#)
 - [Step 3. Deploying Message Manager on UNIX](#)
If the target server is a UNIX based, build Message Manager maps and deploy to the UNIX server.
 - [Step 4. Create a configuration file](#)
 - [Step 5. Modify the E-mail alert settings](#)
 - [Step 6. Configure the Launcher service startup parameters](#)
 - [Step 7. Modify the Message Manager database/query file](#)
 - [Step 8. Identify the Trading Manager database](#)
 - [Step 9. Define Partners, Post Offices and Trade Links](#)
 - [Step 10. Run the EDI Wizard](#)
 - [Step 11. Enable stampandsortFTP, stampandsortVAN, and stampandsortEmail maps](#)
 - [Step 12. Modify Femail.mtt to support proprietary data](#)
 - [Step 13. Add application maps and link to Message Manager](#)
 - [Step 14. Use the deploy facility in the Integration Flow Designer](#)
 - [Step 15. Build and copy application and communication maps](#)
 - [Step 16. Start the Launcher service](#)
-

Step 1. Define the server types in the msg_mgr.msd file

About this task

In the following steps, use the Integration Flow Designer to define the server types in the **MessageManager** and **RunMaps** systems.

To define the server types, perform the following steps:

Procedure

1. From the Start menu, select Programs->IBM WebSphere Transformation Extender n.n->Trading Manager->Message Manager.
2. In the IFD navigator, click the Composition tab to show the **msg_mgr** system definition file.
3. Select **msg_mgr**.
4. From the Server menu, choose **New**.

The Add Server window opens.

5. Enter all values and click OK.

Step 2. Modify the msg_mgr.msd default path name settings

About this task

The Message Manager map and other components are configured assuming a default directory structure of:

C:\install_dir\tmgr_vn.n\mmgr

If you installed Message Manager to a different location, you need to update the Message Manager path names to reflect your environment.

To update the path names, follow the example steps to open the **msg_mgr.msd** system definition file in the Integration Flow Designer and use the **Replace** feature. The **Replace** feature enables you to change the Design-Time Paths and Run-Time Paths for **msg_mgr.msd**:

- **Design-Time Paths** are the directory paths for your system definition file and referenced source/compiled map components
- **Run-Time Paths** are the directory paths for input and output cards, audit log, trace and work files, and server map

The following example steps assume that you installed and will operate in the **d:\apps** directory and your client directory structure mirrors your server directory structure. After completing this step, your map components point to the correct path and file on your **d:\install_dir\tmgr_vn.n\mmgr** directory structure. You can verify this by trying to access any of the map components in Message Manager.

- [Changing design time, run time paths](#)

To change to design time, and run time paths:

1. From the Start menu, select Programs > IBM WebSphere Transformation Extender n.n > Trading Manager > Message Manager.
2. In the IFD navigator, click the Composition tab to show the **msg_mgr** system definition file.
3. Double-click MessageManager to show the **MessageManager** system.
4. From the **Edit** menu, choose **More Find/Replace**.
The Find/Replace window opens. Select the Replace tab.
5. Specify the **Find What** search criteria. To locate all path names that begin with C:, enter C: in the **Find What** field and select Path Names for the **Search Criteria**.
6. Click the **Settings** button to specify additional search parameters.
The Path Name Settings window opens.
7. Ensure that **Run-Time Paths** is disabled, and **Design-Time Paths** is enabled.
Note: Set the Design-Time settings before the Run-Time settings.
8. Click OK to save the settings and close the Path Name Settings window.
The Path Name Settings window closes and you are returned to the Find/Replace window.
9. With the Replace tab view selected, click Find.
A list of the run-time paths that match the search criteria is displayed in the list. In this case, the **C:\install_dir** directory structure is found.
10. In the **Replace** field, enter the path name you wish to replace: C:.
11. In the **With** field, specify the actual location of your **Run-Time Paths**; in this example, d:\apps.
12. Click Select All to select all entries in the list of paths matching the search criteria.
13. Click Replace.
14. Repeat steps 6 through 13; in step 7, enable **Run-Time Paths** and disable **Design-Time Paths**.
15. After performing this procedure for the **MessageManager** system, repeat all steps for the **RunMaps** systems.
Your map components now point to the correct path and file on your **D:\apps\IBM\tmgr_vn.n\mmgr** directory structure. You can verify this by trying to access any of the map components in Message Manager.

Changing the .mrn path names

To change the **.mrn** path names:

1. Open the **Resource Registry**.
2. Right-click on the **Resource Files** folder and choose **Open**.
3. Navigate to the **install_dir\tmgr_vn.n\mmgr** and open the **mmgr.mrn** file.
The file opens in the Resource Registry window.
4. Expand the **mmgr** file and expand the **Resources** folder.
5. Double-click each of the resources under the Resources folder (for example, **mmgr_share**).
The **EditResource** window opens.
6. In the **Resource Value** column, enter the new location of the share directory for the required server.
7. Repeat steps 5 and 6 of this procedure for each resource.

See the Resource Registry documentation for further information.

Step 3. Deploying Message Manager on UNIX

If the target server is a UNIX based, build Message Manager maps and deploy to the UNIX server.

About this task

To do this, the server location of all Message Manager maps, must be changed to reflect the deployment directory.

Message Manager maps use standard Windows directory nomenclature which must be changed to UNIX nomenclature (as an example, \ replaced by /).

Procedure

1. From the Start menu, select Programs->IBM WebSphere Transformation Extender n.n->Trading Manager->Message Manager.
2. In the IFD navigator, click the Composition tab to show the **msg_mgr** system definition file.
3. Double-click MessageManager to show the **MessageManager** system.
4. From the **Edit** menu, choose **More Find/Replace**.
- The Find window opens.
5. Click the Replace tab.
6. The Replace window opens.
7. Specify the **Find What** search criteria. To locate all path names that begin with C:, enter C: in the **Find What** field and select Path Names for the **Search Criteria**.
8. Click the **Settings** button to specify additional search parameters.
- The Path Name Settings window opens.
9. Disable the **Design Time Paths** checkbox.
10. Enable the **Run Time Paths** checkbox.
11. Disable all checkboxes in the **Run Time Paths** area of the window except the **Audit Log** and **Server Map Location** checkboxes.
12. Click OK.
- The Path Name Settings window closes and you are returned to the Find/Replace window.
13. In the **Find What** field, enter C:\install_dir.
14. Click Find.
15. Select all show entries.
16. In the **With** field enter the UNIX installation directory (excluding the **tmgr_vn.n/mmgr**).
17. Repeat steps 11 to 13, changing the FIND string to \, and the Replace string with /.

Step 4. Create a configuration file

Use the Resource Registry Tool to create a **.mrc** file. This file should point to the **mmgr.mrn** and to the **msg_mgr.msl** files that are created during deployment for the Launcher, Command Server and Global settings. Save this file as **mmgr.mrc**.

Step 5. Modify the E-mail alert settings

About this task

The Message Manager is configured with a default e-mail type of MAPI. If your mail system is other than a MAPI mail system (such as Microsoft Mail or Microsoft Exchange), modify a map rule to prepare the Message Manager for your environment.

To change the default email alert setting perform the following steps:

Procedure

1. Open the **Resource Registry**.
2. Open the **msg_mgr.msd** and **mmgr.mrc** files.
3. Select the relevant server. For example, **LocalServer**.
4. Select the **mmgr_e-mail_settings** Resource.
5. Select the **Resource Value** and enter the override settings.
6. Turn **Encryption On** or **Off**.
Note: If you use passwords, you will want to use the encryption feature.
7. Click OK.
8. Exit out by selecting Exit from the File menu.
9. Save changes.

Results

See the E-mail Adapter documentation for information regarding the e-mail adapter.

Step 6. Configure the Launcher service startup parameters

About this task

The following procedure is optional and is applicable to Windows 7, Windows 2008 and Windows XP only.

Some e-mail options supported by Trading Manager require a modification to the default Launcher service Startup parameters. MAPI mail systems such as Microsoft Mail and Microsoft Exchange require access to a user profile when receiving MAPI calls from a service. Configuration requires the creation of a user profile under an existing user account.

It is recommended that you create an **M4EC** user with appropriate permissions including the **User Rights Policy - Log on as a service**. This permission allows the Launcher Service to be accessed by the **M4EC** userid even if the server is running in unattended Automatic mode.

After the **M4EC** userid is established, create a **Mail** profile named **M4EC** that has permission to send mail to any appropriate parties. The **M4EC** mail profile does not need to receive mail to fulfill the alert functions. If you use the pager alert option, ensure that your mail administrator has given the profile sufficient permissions.

After you or your system administrator creates the **M4EC** userid and **M4EC** mail profile, modify the default **Startup** parameters for the Launcher Service.

To modify the default Launcher service startup parameters on Windows 7, Windows 2008 and Windows XP perform the following steps:

Procedure

1. From the Start menu, open the **Control Panel**.
2. Double-click **Administrative Tools**.
3. Double-click **Services**.
The Services window opens.
4. Double-click **Launcher**.
The service name may also include a version number.
5. On the General tab, select **Automatic** from the **Startup type** drop-down box.
6. Click the **LogOn** tab, enable the **This account** field and enter **M4EC**.
7. Enter the M4EC password in the **Password** and **Confirm password** fields.
8. Click **Apply** and then click **OK**.

Step 7. Modify the Message Manager database/query file

About this task

You need to change the database type in the *install_dir\tmgr_vn.n\mmgr\update.mdq* file to match the database type for your installation, for example Microsoft SQL Server or Oracle. Trading Manager is configured to use Microsoft Access as its database. If you have built your prototype environment using Microsoft Access, you must convert from Access to your database type in preparation for your production environment.

The Message Manager default database definition is an open database connectivity (ODBC) database for the Windows platform. See the Partner Manager documentation for more information.

Each Partner Manager client must have an ODBC System data source name (DSN) for both test and production servers.

To modify the **update.mdq** database/query file perform the following steps:

See the Database Interface Designer documentation for information on defining databases.

Procedure

1. Open the Database Interface Designer.
2. Open the **Update.mdq** file in *install_dir\tmgr_vn.n\mmgr*.
3. Expand **Update** to show the **TmrcMgrDB** database definition; select TmrcMgrDB.
4. From the Database menu, select **Edit**.
The **Database Definition** window opens.
5. Expand the **Adapter** setting.
6. From the **Type** drop-down list, select the appropriate database type.
The database-specific fields appear in the **Database Definition** window.
7. Specify the **Platform** from the drop-down list.
8. Enter the appropriate information for the selected database.
9. Click **OK** to set these database settings.
10. From the File menu, choose **Save** to save the database/query file.

Step 8. Identify the Trading Manager database

About this task

In this step, identify the ODBC Data Source Name for each Partner Manager client. Every client must have an ODBC System data source name (DSN) for both test and production servers.

- [Identify the Trading Manager database connection](#)

Identify the Trading Manager database connection

About this task

It is not necessary to define an ODBC Data Source Name (DSN) for your database because you are no longer using ODBC connectivity from the Message Manager server. However, Partner Manager requires an ODBC DSN definition on every client machine from which it will be used.

To identify the ODBC DSN for each client, perform the following steps:

Procedure

1. On the Trading Manager client machine, start Partner Manager.
2. From the File menu, choose **Database Settings** and enter the appropriate information into the fields in the Database Selection window.

Step 9. Define Partners, Post Offices and Trade Links

About this task

Using the Partner Manager, you must define at least one of each of the following to prepare for the remaining steps:

- Internal trading partner
- External trading partner
- **GET** post office
- **PUT** post office
- Trade link

See the Partner Manager documentation for information on defining trading relationships in Partner Manager.

Step 10. Run the EDI Wizard

About this task

After defining one or more trade links in Partner Manager, run the EDI Type Tree Wizard to create the **x12mail.mtt** type tree required by Message Manager. See the Partner Manager documentation for instructions.

Ensure that you create or copy the **X12mail.mtt** in the *install_dir\tmgr_vn.n\mmgr* directory. Message Manager uses this file for validation purposes.

Step 11. Enable stampandsortFTP, stampandsortVAN, and stampandsortEmail maps

About this task

Note: This step is optional.

Refer to "[stampandsortFTP/VAN/Email Maps](#)" for instructions to set up FTP, VAN, and e-mail communications in the **StampAndSort** subsystem.

Step 12. Modify Ecmail.mtt to support proprietary data

About this task

Note: This step is optional.

If you will be supporting non-EDI data types, modify the **Ecmail.mtt** type tree before building the Message Manager maps.

Step 13. Add application maps and link to Message Manager

About this task

Message Manager does not address your organization's specific application mapping requirements. It retrieves or sends data, validates, audits, and routes it to a directory for further action; it does not transform that data without a user-written map.

See the Integration Flow Designer documentation for information about creating **.msd** files and adding systems, subsystems, and map components.

- [Adding a Message Manager application map](#)
- [Configuring Launcher settings](#)

Adding a Message Manager application map

It is recommended that all application and communication maps are specified in a system definition file other than **msg_mgr.msd**. Having a separate file allows you to manage your version control and allows you to upgrade your Message Manager at any time without affecting your specific implementation.

To add an application map, perform the following steps:

1. Create a new system definition file named **AppMap**.
 - Select System Definition Files in the navigator.
 - From the File menu, choose New.
 - Right-click the new system file (**SystemFile1**) and save it as **AppMap**.
2. Add the user application map to the system and name it **P0toFlat**.
 - Enable the Toolbox by choosing **Toolbox** from the View menu.
 - Click the **AddSourceMap** tool on the Toolbox.
 - Click the **AppMap** system window.
 - In the **AddSourceMap** window, browse for the **.mms** map source file to add, name it **P0toFlat**, and click OK

Configuring Launcher settings

You configure the Launcher settings so that the **P0toFlat** map component executes (or triggers) when the EDI data arrives in your Put File post office. Here, configure the **P0toFlat** map component to have an Input Event trigger for the first input card with the File source defined as **d:\apps\IBM\tmgr_vn.n\mmgr\3050850*i.APP**

For this example, assume the following:

- The Put File PO Nickname is APP and its location is defined in Partner Manager as **d:\apps\IBM\tmgr_vn.n\mmgr\3050850**
- The EDI data is arriving from a VAN post office named **VAN**, which retrieves its data every hour
- As Message Manager gets the data from VAN, it passes through the **stampandsortfile**, **sortx12**, and **routex12** maps before being written to the Put File Post Office as
d:\apps\IBM\tmgr_vn.n\mmgr\3050850\VAN00001199809150001oi.APP
- In the **AppMap.msd** system window, ensure that the **Execution Mode** drop-down menu is set to Launcher.
- Click the execution settings button to display the Launcher Settings window.
- Click OK to save the Launcher settings.

Step 14. Use the deploy facility in the Integration Flow Designer

About this task

The Integration Flow Designer documentation describes the Deploy facility in more detail.

The Message Manager is delivered as a set of source maps. To complete the deployment tasks, you may build compiled map objects, transfer them to their appropriate server directory locations, and generate the Launcher control files (.msl) as separate procedures using the Integration Flow Designer (IFD).

However, the IFD Deploy facility automates these build, transfer, and generate steps that prepare systems to run on designated servers. The IFD includes four Deploy scripts to automate these three separate tasks. The Deploy scripts are **DeployMessageManager**, **BuildRunMaps**, **Deploy-MRN-Update** and **GenerateMSLonly**.

The following procedures enable you to:

- Build the MessageManager and RunMaps maps
- Generate the **msg_mgr.msl** Launcher control file
- Copy the compiled map files (.mmc) and Launchercontrol file (.msl) to the corresponding server directory (optional)
- [Using the IFD deploy facility](#)

Using the IFD deploy facility

To use the Deploy facility, perform the following steps:

1. In the IFD navigator, click the Composition tab to show the **msg_mgr** system definition file.
2. Expand **msg_mgr** to show the **MessageManager** and **RunMaps** systems.
3. Right-click **MessageManager** and select **Deploy...>Definitions**.
4. Select **Generate and Transfer Launcher Control File** and click Details.
5. In the window, select the path and filename to save the Launcher control file (.msl).
The default full path is **C:\install_dir\system\msg_mgr.msl**
6. Click Save and **Close**.
7. Right-click **MessageManager** and select **DeployMessageManager**.
The IFD builds the **MessageManager** maps and generates the **msg_mgr.msl** Launcher control file in the specified folder. It also copies compiled map objects (.mmc) to the server directory.
8. Click OK to view the specified deployment results.
9. Right-click **RunMaps** in the navigator and select **BuildRunMaps**.
The IFD builds the **RunMaps** maps.
10. Click OK to view the deployment results.

Step 15. Build and copy application and communication maps

About this task

The following steps can be performed in a deploy script you create. The Integration Flow Designer documentation describes the Deploy facility in more detail.

Build your application and communication maps as described in the following procedure; ensure that you select the application and communication map system definition file. All maps should compile without error. If any errors occur, those errors must be resolved before copying them to the server. Then copy the compiled map files to the corresponding server directory.

To build the application and communication system source map objects and copy the compiled map objects to the server, perform the following steps:

Procedure

1. Open the system definition file (**.msd**) in the Integration Flow Designer.
2. Display the System window for the system.
3. From the System menu, choose **Build Maps**.
4. Click Results to view a detailed summary of the build process and verify that all maps have been built successfully.
All maps should compile without error. If any errors occur, those errors must be resolved before continuing
5. Click OK.
6. Copy the compiled map objects (**.mmc**) to the corresponding server directory.
 - [Generate the Launcher control file for an application system](#)

Generate the Launcher control file for an application system

About this task

To generate the Launcher control file for an application system, perform the following steps:

Procedure

1. Open the application's system definition file in the Integration Flow Designer.
2. Display the System window for the system containing the application map components.
Always generate the Launcher control file **.msl** at the top level system. For instance, if **AppSysA** contains subsystem components **AppSys1** and **AppSys2**, generate the file for **AppSysA**.
3. From the System menu, choose **Generate**.
The Save As window opens to prompt you to enter the file name and location for the **.msl** file.
4. Enter the name and location for the **.msl** file:
 - Navigate to, and save, the **.msl** file to the **Systems** directory on the Trading Manager server.
5. Click Save to generate the file.

Step 16. Start the Launcher service

About this task

The final step in deploying the Message Manager is to start the Launcher Service. The Launcher documentation contains detailed information about configuring and starting the Launcher on both the Windows and UNIX platforms.

To start the Launcher service on Windows 7, Windows 2008, and Windows XP perform the following steps:

Procedure

1. From the Start menu, open the **Control Panel**.
2. Double-click **Administrative Tools**.
3. Double-click **Services**.
The Services window opens.
4. Select **Launcher**, right-click, and select **Start**. Or, click **(Start Service)** in the tool bar.
The service name may also include a version number.
5. When the Launcher is started, the **Status** column will display **Started**.

Results

See the Launcher documentation for Launcher starting information.

- [Launcher Manager tools](#)
-

Launcher Manager tools

About this task

The following are the Manager Tools for the Launcher:

- Launcher Administration
- Management Console
- Launcher Monitor
- Snapshot Viewer

You can use the Management Tools for status information and debugging. The Launcher management tools are described in the Launcher documentation. Use the Configuration file **mmgr.mrc** in the Launcher Administration setting.

Using Message Manager

The following topics are described here:

- The **MessageManager** system, which is defined in the **msg_mgr.msd** system definition file
- Inbound and outbound data flows
- Subsystems in **msg_mgr.msd**

As installed, the **MessageManager** system is a set of source maps. The system retrieves or sends data, then validates, audits, and routes it for further action. After adding your application map(s) according to your e-commerce configuration, you compile the maps and transfer them to their Trading Manager server locations before starting the Launcher. After you start the Launcher, event triggers execute the compiled maps and process your data.

The **MessageManager** system implements the logical functionality and EDI-specific features of the Trading Manager. It also provides a general framework for any data you choose to support in the system. The **MessageManager** system includes six major subsystems: **Archive**, **EDIFACT**, **StampAndSort**, **TRADACOMS**, **Update**, and **X12**. Each major subsystem contains maps or other subsystems.

In the top level view of the **MessageManager** system, the **StampAndSort** and **Update** subsystems provide a general data-handling framework. The **X12** subsystem is specific to ANSI X12 EDI functionality.

To fully deploy the Message Manager, you must add:

- Application maps
- Implementation-specific configuration settings
- Trading partner information supplied through the Partner Manager
- Actual data activity (that is, events on the Launcher)

Message Manager subsystems

The **msg_mgr.msd** system definition file has two executable systems: **MessageManager** and **RunMaps**. The **MessageManager** system has six subsystems: **Archive**, **EDIFACT**, **StampAndSort**, **TRADACOMS**, **Update**, and **X12**.

- [Archive subsystem](#)
- [EDIFACT subsystem](#)
- [EDIFACT inbound subsystem](#)
- [EDIFACT outbound subsystem](#)
- [StampAndSort subsystem](#)
- [TRADACOMS subsystem](#)
- [TRADACOMS inbound subsystem](#)
- [TRADACOMS outbound subsystem](#)
- [Value Added Tax \(VAT\) reports](#)
- [Update subsystem](#)
- [X12 subsystem](#)
- [X12 inbound subsystem](#)
- [X12 outbound subsystem](#)

Archive subsystem

The primary function of the Archive subsystem is to archive both inbound and outbound data. The Archive components archive files processed by the system and place them into the **archive\<process>\<process>_date.zip** file. Folder **stampfil** is used to archive all input data to Trading Manager, inbound or outbound. Folders **sendx12**, **sendedf**, and **sendtrc** are used to archive outbound files going out from Trading Manager.

You can selectively turn on/off the archiving process by going into the "Message Manager Configuration" options in Partner Manager. Options available are:

- Turn on/off archiving of Inbound files into Trading Manager.
- Turn on/off archiving of Outbound files into Trading Manager.

- Turn on/off archiving of Outbound files out from Trading Manager.
The **batchutil** component will perform batch archiving of online traffic information into the online archive. If is scheduled to run every night at 11:30 PM. Batch archiving is turned on/off and the corresponding options are specified through the "Message Manager Configuration" options in Partner Manager.

EDIFACT subsystem

The **EDIFACT** subsystem contains two subsystems: **EDIFACT Inbound** and **EDIFACT Outbound**.

- The **EDIFACT Inbound** subsystem provides sorting, validating, and routing functions for data flowing from external trading partners to internal trading partners.
- The **EDIFACT Outbound** system provides enveloping, validating, and routing functions for data flowing from internal trading partners to external trading partners.

EDIFACT inbound subsystem

The **EDIFACT Inbound** subsystem performs the validation and routing functions on inbound EDIFACT formatted data. The EDI data flows through three primary maps: **Process EDIFACT Inbound**, **edifact_in_alert**, and **EAudit_Inbound**.

- [edifact_in_alert map](#)

edifact_in_alert map

The **edifact_in_alert** map is a common map component used throughout the Message Manager. The **edifact_in_alert** map is configured to send alert notifications through e-mail or pager whenever data is rejected.

In the **EDIFACT Inbound** subsystem, the **edifact_in_alert** map is responsible for sending alert notifications when an error condition occurs in the **Process EDIFACT Inbound**, **edifact_in_alert**, or **EAudit_Inbound** maps. The rejected data is written to the *install_dir\tmgr_vn.n\mmgr\reject* directory and the appropriate form of notification is sent to the user-specified destination.

EDIFACT outbound subsystem

The **EDIFACT Outbound** subsystem performs the enveloping, validating, and routing functions on outbound EDIFACT formatted data. The EDI data flows through two primary maps: the **envedf** map and the **sendedf** map.

- [Send problem data alerts map](#)

Send problem data alerts map

The **Send Problem Data Alerts** map is configured to send alert notifications through e-mail or pager whenever data is rejected.

In the **EDIFACT Outbound** subsystem, the **Send Problem Data Alerts** map is responsible for sending alert notifications when an error condition occurs in the **envedf** or **sendedf** maps. The rejected data is written to the *install_dir\tmgr_vn.n\mmgr\reject* directory and the appropriate form of notification is sent to the user-specified destination.

StampAndSort subsystem

The purpose of the **StampAndSort** subsystem is to route different data types to their appropriate subsystem or map.

For example, if you retrieve a file from a customer's FTP server that contains ANSI X12 EDI data and proprietary flat file format, the **StampAndSort** subsystem regroups similar data types into a single file, then routes them to either the X12 subsystem or a user-defined map or subsystem designed to handle proprietary data.

The **StampAndSort** subsystem also includes the standard error-handling and alert mechanisms for unrecognizable data and an audit function for tracking purposes.

The **StampAndSort** subsystem serves as the entry and exit point for data flowing through Trading Manager.

The three primary functions of the **StampAndSort** subsystem are as follows:

- Send and receive data
 - Prepare the data for processing by assigning it a unique **ThreadID** as its filename
 - Identify databases (EDIFACT, TRADACOMS, and X12)
-
- [ThreadID definition](#)
 - [Inbound data archiving](#)
 - [stampandsortFTP/VAN/Email maps](#)
 - [Creating multiple stampandsortFTP maps](#)
 - [stampandsortfile map](#)
 - [Get file post office and put file post office](#)
 - [stampandsortack map](#)

- [Resend map](#)

ThreadID definition

The **ThreadID** contains the **TransmissionID**. The Trading Manager uses the **ThreadID** to track unique data objects as they pass through the system and to facilitate concurrent processing. The elements of the **ThreadID** are shown below. In this example we will use a typical **ThreadID** for illustrative purposes:

tes0000012000118oi.app

The elements of this **ThreadID** are defined in the following table:

| ThreadID element | Element name | Element description |
|------------------|--------------------------|--|
| tes | Get Post Office Nickname | These first three characters of the file name are the nickname of the Get post office from which the e-commerce data originated. |
| 0000001 | Mailbag Number | Number of different unique files received from a given post office per day. |
| 2000118 | Date received | Date the data goes through the StampAndSort subsystem (when ThreadID gets created). |
| o | Original | This value identifies whether this data is original data (o) going through the Trading Manager for the first time or is data that is being resent (r) through the Trading Manager. See the <i>Partner Manager Reference Guide</i> for information on resending data. |
| i | Direction | The last three characters following the period are the nickname of the Put post office where the e-commerce data is being placed. |
| app | Put Post Office Nickname | The last three characters following the period are the nickname of the Put post office where the e-commerce data is being placed. |

Inbound data archiving

As data is passed thru the Message Manager it is broken up into the respective Interchanges and archived in a specified directory.

To specify an archived directory

1. Click the **Server Directory Configuration** button on the Utilities window.
The Server Directory Configuration Maintenance window appears.
2. Type the desired archive directory in the **Archive** (contains 'in' and 'out' directories) field.
3. Click **Save**.
If you do not specify an archive directory, you will not be able to automatically archive your files. If you do create an archive, data can be edited and resent from the Interchange Traffic.

The archive directory you specify must include **in**, **out**, and **day** directories. For example, *install_dir\tmgr_vn.n\mmgr\archive*.

The archived filename will be **<thread_id>_<interchange_id>.<edi_standard>**.

Where **<thread_id>** is the internal **thread_id** of the transmission, **<interchange_id>** is the Interchange control number, and **<edi_standard>** will be EDIFACT or X12. For example, **0000010200106210000_635.X12**.

This data is used by the resend and editing functionality in Partner Manager and can be zipped or deleted from there. See the Partner Manager documentation for instructions on how to resend data.

stampandsortFTP/VAN/Email maps

The **stampandsortFTP**, **stampandsortVAN**, and **stampandsortEmail** maps are identical with the exception of the communication protocol. The configuration requirements for these three maps are described here, using the **stampandsortFTP** map as a specific example.

The **stampandsortFTP** map is designed to retrieve data using the FTP protocol and to submit that data to the Message Manager. The **stampandsortFTP** map performs the following four tasks:

- Gets data through the FTP resource adapter, using the FTP configuration information you supplied when you defined the **FTP Get** post office. An example FTP command line might be:

```
-URL ftp://ediuser:password@192.1.1.50/receive.txt?type=i
```

- Creates a unique **ThreadID** for the entire FTP transmission (the contents are concatenated as a single data stream if you use the FTP command **MGET**).
- Archives the data retrieved through FTP in the *install_dir\tmgr_vn.n\mmgr\archive\stampapp* directory with an archive name adhering to the archive naming convention requirements.
- Writes the data out to a file as *install_dir\tmgr_vn.n\mmgr\stampec*.mec*. The * represents the valid **ThreadID** assigned to the FTP transmission.

As delivered with the Message Manager, the **stampandsortFTP** map is *not* configured to run by default. The **stampandsortFTP** map does not have time or input event triggers defined. The following procedure describes how to activate the **stampandsortFTP** map.

To activate the stampandsortFTP map

Note: The following procedures assume you have experience with Partner Manager and the Design Studio.

1. Create an **FTP Get** post office in Partner Manager with a valid FTP command line.
(Create post offices using the Partner Manager Post Office navigator. In this navigator window, click Add
See the FTP Adapter documentation for information about valid FTP command lines.)

2. Open the **msg_mgr.msd** system in the Integration Flow Designer.
3. Select the **stampandsortFTP** map component in the **StampAndSort** subsystem.
4. Click the execution settings button on the **stampandsortFTP** map component to display the Launcher Settings window.
5. For the **TimeEvent** setting, add a time event trigger by setting the TimeEvent Switch to **ON** and specifying the Trigger Interval settings.
As discussed in the **export** map documentation, it is recommended that time events for FTP sessions be scheduled to start a few minutes after system startup to ensure the export process has completed before receiving data. You may also consider distributing FTP sessions over time if your systems are bandwidth-constrained.
6. Modify the **Input(s)** information for the #1 **POInfo** input card.
The GET source field for input card 1 (POInfo) of the stampandsort maps need to be updated with a string to indicate the Post Office nickname and the direction as follows **-PON EDI -DIR i**, where PON and DIR are keywords and are case-insensitive. The value for **dir** can be **i** (inbound), or **o** (outbound).
Note: The three-character **PO Nickname** must match the nickname created in Partner Manager. The Message Manager uses this three-letter nickname to create a valid and unique **ThreadID**.

Creating multiple stampandsortFTP maps

About this task

You will need an FTP session for each FTP **Get** post office that you define. You will probably need more than one FTP session to support your e-commerce needs. For each additional FTP location you want to support, create a copy of the **stampandsortFTP** map and repeat the steps in the previous procedure.

You do not need to copy the **stampandsortFTP** map into the **msg_mgr.msd** system definition file. It is recommended that you create a separate system definition file (**.msd**) for all communication maps. Having a separate file allows you to upgrade your core Message Manager at any time without affecting your specific implementation.

To create a copy of **stampandsortFTP** in the Message Manager, perform the following steps:

Procedure

1. Open the **msg_mgr.msd** system in the Integration Flow Designer.
2. Select the **stampandsortFTP** map component in the **StampAndSort** subsystem.
3. From the **Edit** menu, choose **Copy**.
4. From the **Edit** menu, choose **Paste**.
The **Copy** command creates a default map component named **stampandsortFTP(2)**. The **stampandsortFTP(2)** copy is pasted on top of the source **stampandsortFTP** map. Move the **stampandsortFTP(2)** map to view both the source and the duplicate map.
5. Rename the new **stampandsortFTP(2)** map component to uniquely identify this FTP session:
 - Right-click **stampandsortFTP(2)** and select **Edit Map Component**.
 - In the **Component Name** field, enter a new name for the map component.
 - Click **OK**.

Results

As a reminder, the **stampandsortFTP**, **stampandsortVAN** and **stampandsortEmail** maps are identical with the exception of the communication protocol they use. To configure the **stampandsortVAN** or **stampandsortEmail** maps, follow the same steps as outlined, replacing the FTP-specific information with VAN or e-mail equivalents.

stampandsortfile map

The **stampandsortfile** map is similar to the other **Stamp*** maps in that it is responsible for creating a **ThreadID**, archiving the data, and writing the data out for retrieval by the **stampandsortfile** map. The **stampandsortfile** map is unique, however, because it requires no configuration. **stampandsortfile** is pre-configured to trigger on any file event that occurs in the installed **mail** directory specified by the **mmgr_mail** resource name.

The only requirement for correctly implementing File Post Offices is the use of valid file names. Any e-commerce data object must have a valid **ThreadID** to successfully process inbound and outbound data objects. The creation of a valid **ThreadID** is done automatically for the application maps. For the **stampandsortfile** map, however, it is your responsibility to create a valid file-triggering name when writing the data to the **mail** directory. The precise file naming conventions for File Post Office files must be followed.

Get file post office and put file post office

The **stampandsortfile** map is only relevant for **Get** File Post Offices. **Put** File Post Offices are implemented directly within the Message Manager and use the Partner Manager post office definitions to route data.

It is usually a good idea to create multiple physical **Put** File Post Office directories, because they perform a "mailboxing" function as well as a triggering function. When Message Manager receives inbound data, that data is validated and routed to the **Put** post office specified in the trade link. This **Put** post office is usually defined as a File Post Office, although it can be any of the post office types (even VAN). User written application maps are then configured to trigger on the arrival of data in a particular **Put** File Post Office directory. Refer to [Step 13. Add application maps and link to Message Manager](#) for more information on the relationship between the **Put** location and triggering application maps.

When defining **Get** File Post Offices in Partner Manager, it is not necessary to create new physical directories for triggering files. You can define all **Get** File Post Offices to point to the existing folder *install_dir\mmgr_vn.n\mmgr\mail*. The role of the post office and its relationship to physical directory structures is explained in the ["StampAndSort Subsystem"](#) documentation.

There is no benefit to defining multiple physical **Get** File Post Office directories. Using the default *install_dir\tmgr_vn.n\mmgr\mail* directory simplifies your implementation if you are using this default directory for all File post offices. However, you can customize your implementation according to your needs.

stampandsortack map

The Trading Manager uses the **stampandsortack** map when generating Functional Acknowledgements. The **stampandsortack** map is configured to run and requires no user intervention. The **stampandsortack** map can also generate an alert message to the **stampandsortalert** map if the Message Manager should generate an invalid Functional Acknowledgement (**ack**) triggering name.

Resend map

The **resend** map works with the Resend feature in Partner Manager to retrieve previously archived data for resending through the Message Manager. The **resend** map has been configured to run correctly and should require no user intervention.

TRADACOMS subsystem

The **TRADACOMS** subsystem contains two subsystems: **TRADACOMS Inbound** and **TRADACOMS Outbound**.

- The **TRADACOMS Inbound** subsystem provides sorting, validating, and routing functions for data flowing from external trading partners to internal trading partners.
- The **TRADACOMS Outbound** system provides enveloping, validating, and routing functions for data flowing from internal trading partners to external trading partners.

It should be understood that the EDI wizard is not used for managing TRADACOMS standards in the Message Manager system.

TRADACOMS inbound subsystem

The **TRADACOMS Inbound** subsystem performs the validation and routing functions on inbound TRADACOMS formatted data. The EDI data flows through three primary maps: **TRADACOMS Inbound**, **TRADACOMS_in_alert_data**, and **TRADACOMS_Traffic_Inbound**.

- [**TRADACOMS_in_alert_data map**](#)
-

TRADACOMS_in_alert_data map

The **TRADACOMS_in_alert_data** map is a common map component used throughout the Message Manager. The **TRADACOMS_in_alert_data** map is configured to send alert notifications through e-mail or pager whenever data is rejected.

In the **TRADACOMS Inbound** subsystem, the **alert** map is responsible for sending alert notifications when an error condition occurs in the **TRADACOMS Inbound** map. The rejected data is written to the *install_dir\tmgr_vn.n\mmgr\reject* directory and the appropriate form of notification is sent to the user-specified destination.

TRADACOMS outbound subsystem

The **TRADACOMS Outbound** subsystem performs the enveloping, validating, and routing functions on outbound TRADACOMS formatted data. The EDI data flows through two primary maps: the **alert_data** map and the **TRADACOMS_Traffic_Outbound** map.

- [**alert_data map**](#)
-

alert_data map

The **alert_data** map is a common map component used throughout the Message Manager. The **alert_data** map is configured to send alert notifications through e-mail or pager whenever data is rejected.

In the **TRADACOMS Outbound** subsystem, the **alert_data** map is responsible for sending alert notifications when an error condition occurs in the **TRADACOMS_Outbound** map. The rejected data is written to the *install_dir\tmgr_vn.n\mmgr\reject* directory and the appropriate form of notification sent to the user-specified destination.

Value Added Tax (VAT) reports

About this task

The TRADACOMS subsystem (inbound and outbound) will create VAT reports for any invoice type messages. For example: INVOIC, CREDIT, and UTLBIL. The VAT report will automatically be created in the default **C:\install_dir\tmgr_vn.n\mmgr\VAT_Reports** directory.

If you wish to change the default directory, you can change the location in the **Edit Resource** window in the **Resource Registry**.

To change the default directory perform the following steps:

Procedure

1. Open the **Resource Registry**.
 2. Select Open from the File menu and navigate to the *install_dir\tmgr_vn.n\mmgr\mmgr.mrm*
 3. Expand the **mmgr** file.
 4. Expand the **Resources** file.
 5. Double-click the **mmgr_trc_VatReport** file.
The **Edit Resource** window opens.
 6. Edit the **Resource Value** to the desired location.
 7. Click OK.
-

Update subsystem

The **Update** subsystem provides the link between the Partner Manager database and the control files used by the Message Manager. The Partner Manager server database stores all the trading partner information needed to dynamically route, validate, envelope and monitor e-commerce data. This control information needs to be available to the Message Manager during run time to support dynamic routing, validation, and enveloping of data.

The **Update** functions can be scheduled to run as frequently as desired.

At user-defined intervals, configuration, routing and validation information is extracted from the Partner Manager database and made available to the Message Manager through a set of files with a **.tsv** file name extension. These **.tsv** files contain the configuration, routing, and validation information used as data flows through the Message Manager to dynamically route and transform the data for delivery to its final destination.

- **export map**
 - **dbalert map**
-

export map

The **export** map is responsible for extracting the data from the Partner Manager database and updating certain state information. The **export** map is triggered to run immediately each time the Launcher is restarted. The run time of the **export** map varies in duration based on the number of trading relationships defined.

As delivered with Message Manager, the **export** map is also triggered to run each night at approximately 12 AM. If you would like to run the **export** map more or less frequently, change the TimeEvent...Trigger settings in the Launcher Settings window for the **Trigger Every Night** map of the Update subsystem. This will allow you to apply immediate changes. Determining the ideal update frequency is a function of how often trading partners are added or updated. If you follow a process that introduces trading partner relationships once a day, the default setting running the **export** map once a day at midnight may be sufficient. If you require more frequent changes, you may want to run the **export** map every 30 minutes.

The **export** map must complete before you can begin processing data. It is recommended that you schedule any user maps with time event triggers to begin a few minutes after the Launcher startup to ensure the availability of the **.tsv** files.

You can trigger the export to run on demand from Partner Manager without cycling the Launcher. Every time the **export** map runs, any changes to trade link information are updated and made available to the **MessageManager** system. The **MessageManager** controls the transition from the old **.tsv** files to the new ones, so there is not a problem with information changing in the middle of a transaction.

It is not necessary to restart the Launcher when adding new trading relationships for existing version/document combinations. Those trading relationships will be active after the **export** map is run. You must restart the Launcher when adding new version/document combinations, as they require a rebuild of the **sortX12** and **sendX12** maps.

dbalert map

The **dbalert** map monitors the **export** map for error conditions. If an error condition occurs, the **dbalert** map sends an e-mail message with an explanation of the error to the e-mail recipient specified in the first output card. The **dbalert** map also saves a copy of the error information to the *install_dir\tmgr_vn.n\mmgr\reject\dberror.log* file.

You must configure the **dbalert** map with the appropriate mail type and recipient ID before running the **MessageManager** system. By default, the **Alert** output card is configured to write the message to: *install_dir\tmgr_vn.n\mmgr\share\Alert.tmp*. See ["Step 4. Modify the E-Mail Alert Settings"](#).

X12 subsystem

The **X12** subsystem contains two subsystems: **X12 Inbound** and **X12 Outbound**.

- The **X12 Inbound** subsystem provides sorting, validating, and routing functions for data flowing from external trading partners to internal trading partners.

- The **X12 Outbound** system provides enveloping, validating, and routing functions for data flowing from internal trading partners to external trading partners.

X12 inbound subsystem

The **X12 Inbound** subsystem performs the validation and routing functions on inbound ANSI X12 formatted data. The EDI data flows through the **sortx12 map**.

- [sortx12 map](#)
- [X12 FA inbound map](#)
- [X12_In_Alert_Data map](#)

sortx12 map

The **sortx12** map checks the ISA and GS control structures for valid trading partner information.

The **ISA** fields examined include:

- Interchange Sender ID Qualifier
- Interchange Sender ID
- Interchange Receiver ID Qualifier
- Interchange Receiver ID
- Test Indicator

The **GS** fields examined include:

- Functional ID Code
- Application Sender ID
- Application Receiver ID
- Version Release Indicator Code

These values are compared to trade links defined in Partner Manager based on the **Get** post office. If no match is found and an **Unknown Partner** link has not been defined, the data is rejected through the **alert** map.

X12 FA inbound map

The X12 FA Inbound map updates outbound traffic information with the status reflected in inbound Functional Acknowledgements.

X12_In_Alert_Data map

The **X12_In_Alert_Data** map is a common map component used throughout the Message Manager. The **X12_In_Alert_Data** map is configured to send alert notifications through e-mail or pager whenever data is rejected.

In the **X12Inbound** subsystem, the **X12_In_Alert_Data** map is responsible for sending alert notifications when an error condition occurs in the **sortx12** map. The rejected data is written to the *install_dir\tmgr_vn.n\mmgr\reject* directory and the appropriate form of notification is sent to the user-specified destination.

X12 outbound subsystem

The **X12 Outbound** subsystem performs the enveloping, validating, and routing functions on outbound ANSI X12 formatted data. The EDI data flows through the **sendx12** map.

- [x12_out_alert map](#)

x12_out_alert map

The **x12_out_alert** map is a common map component used throughout the Message Manager. The **x12_out_alert** map is configured to send alert notifications through e-mail or pager whenever data is rejected.

In the **X12 Outbound** subsystem, the **x12_out_alert** map is responsible for sending alert notifications when an error condition occurs in the **sendx12** map. The rejected data is written to the *install_dir\tmgr_vn.n\mmgr\reject* directory and the appropriate form of notification sent to the user-specified destination.

Support for proprietary data types

These instructions describe how to add proprietary data types to the Trading Manager, which supports EDI standard and proprietary data types. You can add other user-defined data types to the Trading Manager to take advantage of the Message Manager.

- [Inbound data flow](#)
 - [Adding proprietary data types](#)
-

Inbound data flow

The Message Manager consists of several subsystems that perform specific functions. The Message Manager maps represent a framework for handling e-commerce data. Any data type may enter the Message Manager through one of the resource adapters. Regardless of direction, the e-commerce data always passes through the **StampAndSort** system first. The **StampAndSort** system assigns a unique **ThreadID** to the e-commerce data object to facilitate multithreaded processing. The **ThreadID** also provides a way to track the e-commerce object through the rest of the system. For information on the Message Manager maps and systems, see the Message Manager subsystems documentation in ["Installing and Using Message Manager"](#).

After the **StampAndSort** subsystem processes the e-commerce data, the subsystem receives it for sorting. **StampAndSort** is based on the default **Ecmail.mtt** type tree and uses the partitioning concept to route e-commerce data dynamically based on its type.

The default **Ecmail.mtt** type tree supports ANSI X12, UN/EDIFACT, TRADACOMS and a Trading Manager proprietary data type. **StampAndSort** rejects any other data type as invalid and passes it into the reject handling system.

You may want to add other user-defined data types to Trading Manager to take advantage of the general e-commerce framework.

Adding proprietary data types

About this task

You only need to define enough of the proprietary data structure to make it distinguishable from the other partitioned subtypes.

To add support for a new data type

Procedure

1. Modify the type tree **Ecmail.mtt** to include a new data type **Test** under the partitioned object **M4EC**.
2. Analyze the **Ecmail.mtt** type tree to ensure your new data type is distinguishable.
3. Save the **Ecmail.mtt** type tree.
4. Build the **stampandsortfile** map component in **MessageManager** using the **Integration Flow Designer**.

The Message Manager is now configured to accept proprietary data. For inbound and outbound proprietary data, follow the same rules and naming conventions as you do with EDI data.

To add application processing for proprietary data

Procedure

1. Add your own map or system of maps designed to transform the proprietary data and pass it to an appropriate application.
2. Configure the map to trigger on: **C:\install_dir\tmgr_vn.n\mmgr\sortec*i.Test** (this configuration assumes inbound transmission).
Note: ***i.Test** represents the actual partition name with **Ecmail.mtt**. For instance, if a proprietary data type is added for **NACHA** data formats, the resulting filename from which to trigger would be ***.NACHA**.

Results

Your proprietary data will be processed like any data type in terms of audit trails and alerts. You will be able to track inbound and outbound transmissions in the Reports section of Partner Manager.

Using alias files in maps

Trading Manager maintains three additional pieces of user data when you define an application partner. This data is contained in the Alias Name, User Defined 1, and User Defined 2 fields and maintained in the Partner Manager database.

You may use this data in maps to reference application partner identification and EDI address information. To use this data:

- Define the alias or user defined fields
- Make the alias available to the application maps
- Choose the alias file to use
 - [Defining alias name and user defined fields](#)
 - [Making the alias data available to application maps](#)
 - [Choosing the alias file to use](#)
 - [Alias files type tree \(Alias.mtt\)](#)
 - [Creating user type trees](#)

Defining alias name and user defined fields

About this task

You define alias and user defined fields when you create or edit application trading partners.

For more information on how to specify an alias name and user defined fields for an application partner, see the Partner Manager documentation.

Making the alias data available to application maps

About this task

Every time the Message Manager **export** map runs, any changes to trade link information are updated and made available to the **Message Manager** system. It produces four files when it is run; each file contains information about each application partner, including:

- Alias Name
- User Defined 1
- User Defined 2

Each file contains the same data but the columns are ordered differently within the row and the rows are sorted differently within the file:

| Description | File Name | Row Sort Order |
|-------------------------------------|--------------|----------------------|
| Alias Information by Partner ID | AliasByP.tsv | By ISA Address (X12) |
| Alias Information by Alias Name | AliasByA.tsv | By Alias Name |
| Alias Information by User Defined 1 | AliasBy1.tsv | By User Defined 1 |
| Alias Information by User Defined 2 | AliasBy2.tsv | By User Defined 2 |

The alias files are written to the Message Manager .\tmgr_vn.n\mmgr\share directory.

The Message Manager does not use these files. They are provided to cross-reference information between your application's identification methods and the corresponding EDI addresses. For instance, you might use Alias Name to store your internal customer ID, account number, or client code. Using the appropriate alias files in your application maps, you could move easily between your internal coding systems and your EDI partner addresses.

Choosing the alias file to use

About this task

Because the four alias files are sorted differently, you can choose which file most efficiently finds the application coding value or EDI address value, based on your application needs and how you have defined the alias and user-defined fields in the Partner Manager.

For example, in an inbound EDI application map, you could use **AliasByP.tsv** (alias by partner ID) to get the appropriate alias name or user defined field for the external trading partner. For an outbound EDI application map where your application data contains the account number for each trading partner, use **AliasBy1.tsv** (alias by User Defined 1, if you have defined the User Defined 1 field to contain your account number).

Alias files type tree (Alias.mtt)

The **Alias.mtt** type tree describes the format of each alias file and is installed in the same directory as the Message Manager source maps and trees.

The **Alias.mtt** type tree contains definitions for each alias file:

| Type | Description |
|---------------------|----------------------------|
| AliasByAlias | By Alias Name |
| AliasByPartner | By ISA Address |
| AliasByUserDefined1 | By each User Defined field |
| AliasByUserDefined2 | By each User Defined field |

All four files use the same column definitions; the only difference between each Row definition is the sequence of the columns within the row. The first column corresponds to the sort order of the file.

This type tree can be used in your application maps where you want to use one or more of the alias files.

Creating user type trees

Overview

The User Type trees are created by the EDI Wizard located in Partner Manager. These type trees are for use in your application maps. If you have both X12 and EDIFACT data, two trees will be generated: **x12user.mtt** and **edifuser.mtt**. These trees are generated in the same directory as the mail trees used by Message Manager. The default

location for this is the Message Manager directory: **C:\install_dir\tmgr_vn.n\mmgr**. For information on how to use the EDI Wizard, see the Partner Manager documentation.

The EDI Wizard creates the type trees based on the X12 and EDIFACT inbound and outbound trade links defined in the Partner Manager. It automatically creates a type tree that contains the type definitions for all transaction sets for all versions that the internal trading partners and external trading partners exchange.

Checklists

The following checklists can be used as site preparation and installation checklists and may be reproduced in any form for use with Trading Manager systems.

Server Installation Checklists:

- [Server Installation Checklist for Microsoft SQL Server](#)
- [Server Installation Checklist for Oracle Database](#)
- [Server Installation Checklist for Sybase](#)
- [Server Installation Checklist for IBM DB2 Database](#)

Client Installation Checklists:

- [Client Installation Checklist for Microsoft SQL Server](#)
- [Client Installation Checklist for Oracle Server](#)
- [Client Installation Checklist for Sybase](#)
- [Client Installation Checklist for IBM DB2 Server](#)

- [Server installation checklist for Microsoft SQL Server](#)
- [Client installation checklist for Microsoft SQL Server](#)
- [Server installation checklist for Oracle database](#)
- [Client installation checklist for Oracle Server](#)
- [Server installation checklist for Sybase](#)
- [Client installation checklist for Sybase](#)
- [Server installation checklist for IBM DB2 database](#)
- [Client installation checklist for IBM DB2 Server](#)

Server installation checklist for Microsoft SQL Server

The following checklist identifies the major installation steps required to install and configure Trading Manager for use with a Microsoft SQL Server database. During installation, fill in the specific information in the blanks provided in this list for future reference.

| | |
|------|---|
| ____ | System meets minimum requirements |
| ____ | Machine Name _____ |
| ____ | IP Address _____ |
| ____ | Modem Brand _____ |
| ____ | Modem Model _____ |
| ____ | Modem COM Port _____ |
| ____ | Microsoft SQL Server version 6.5, 7.0 (with Microsoft SQL Server Service Pack 3) or Microsoft SQL Server Version 2000 on the server |
| ____ | Database Device Name _____ |
| ____ | Log Device Name _____ |
| ____ | Database Name _____ |
| ____ | Userid _____ |
| ____ | Password _____ |
| ____ | Install the Launcher |
| ____ | Directory Path _____ |
| ____ | Install Trading Manager Server |
| ____ | Directory Path _____ |

Client installation checklist for Microsoft SQL Server

The following checklist identifies the major steps required to install and configure Trading Manager when connecting to a Microsoft SQL database server. During installation, fill in the specific information in the blanks provided in this list for future reference.

| | |
|------|---|
| ____ | System meets minimum requirements |
| ____ | Microsoft SQL Server database has been constructed on the server |
| ____ | Microsoft SQL Client Services have been installed on each Partner Manager |
| ____ | Check ODBC driver for SQL on each Partner Manager |
| ____ | Install Design Studio |
| ____ | Define ODBC DSN |
| ____ | Install Message Manager - Message Manager Client (either Application or Design) |
| ____ | Directory Path _____ |

Server installation checklist for Oracle database

The following checklist identifies the major installation steps required to install and configure Trading Manager for an Oracle database server. During installation, fill in the specific information in the blanks provided in this list for future reference.

| | |
|------|---|
| ____ | System meets minimum requirements |
| ____ | Machine Name _____ |
| ____ | IP Address _____ |
| ____ | Modem Brand _____ |
| ____ | Modem Model _____ |
| ____ | Modem COM Port _____ |
| ____ | Install Oracle version 7.x, Oracle8.x., Oracle9.x, or Oracle10x on the server |
| ____ | User Name _____ |
| ____ | Password _____ |
| ____ | Install the Launcher |
| ____ | Directory Path _____ |
| ____ | Install Trading Manager Server |
| ____ | Directory Path _____ |

Client installation checklist for Oracle Server

The following checklist identifies the major steps required to install and configure Trading Manager connecting to an Oracle database server. During installation, fill in the specific information in the blanks provided in this list for future reference.

| | |
|------|---|
| ____ | System meets minimum requirements |
| ____ | Oracle database version 7.x, Oracle8.x, or Oracle9.x installed |
| ____ | Oracle Client Utilities have been installed on each Partner Manager |
| ____ | Install ODBC driver for Oracle on each Partner Manager |
| ____ | Install Design Studio |
| ____ | Define ODBC DSN |
| ____ | Install Trading Manager Server |
| ____ | Directory Path _____ |

Server installation checklist for Sybase

The following checklist identifies the major installation steps required to install and configure Trading Manager for a Sybase. During installation, fill in the specific information in the blanks provided in this list for future reference.

| | |
|------|-----------------------------------|
| ____ | System meets minimum requirements |
| ____ | Machine Name _____ |
| ____ | IP Address _____ |
| ____ | Modem Brand _____ |
| ____ | Modem Model _____ |
| ____ | Modem COM _____ |
| ____ | Install Sybase on the server |
| ____ | User Name _____ |
| ____ | Password _____ |
| ____ | Install the Launcher |
| ____ | Directory Path _____ |
| ____ | Install Trading Manager Server |
| ____ | Directory Path _____ |

Client installation checklist for Sybase

The following checklist identifies the major steps required to install and configure Trading Manager connecting to a Sybase. During installation, fill in the specific information in the blanks provided in this list for future reference.

| | |
|------|--|
| ____ | System meets minimum requirements |
| ____ | System has been constructed on the server |
| ____ | Sybase Client Services have been installed on each Partner Manager |
| ____ | Install ODBC driver for Sybase on each Partner Manager |
| ____ | Install Design Studio |
| ____ | Define ODBC DSN |
| ____ | Install Trading Manager Server |

Server installation checklist for IBM DB2 database

The following checklist identifies the major installation steps required to install and configure Trading Manager for an IBM DB2 database server. During installation, fill in the specific information in the blanks provided in this list for future reference.

| | |
|-------|--|
| ----- | System meets minimum requirements

Machine Name _____

IP Address _____

Modem Brand _____

Modem Model _____

Modem COM _____ |
| ----- | Install IBM DB2 Database version 6.1 or higher on the server

User Name _____

Password _____ |
| ----- | Install the Launcher

Directory Path _____ |
| ----- | Install Trading Manager Server

Directory Path _____ |

Client installation checklist for IBM DB2 Server

The following checklist identifies the major steps required to install and configure Trading Manager connecting to an IBM DB2 database server. During installation, fill in the specific information in the blanks provided in this list for future reference.

| | |
|-------|--|
| ----- | System meets minimum requirements |
| ----- | IBM DB2 database version 6.1 or higher installed |
| ----- | Install ODBC driver for DB2 on each Partner Manager |
| ----- | Install Design Studio |
| ----- | Define ODBC DSN |
| ----- | Install Trading Manager Server
Directory Path _____ |

Implementing custom X12 versions

This section describes how to implement custom X12 versions such as 004010VICS, 005010RAIL, and/or R2/8.

- [Overview](#)
- [Type trees](#)
- [Partner Manager](#)
- [Message Manager](#)
- [Creating structure records](#)

Overview

Trading Manager uses the **x12mail** Type Tree to validate X12 transactions, and Partner Manager Structure Records to identify segment and data element errors if the transaction fails validation by **x12mail**.

If a custom X12 version is a subset of an ANSI X12 version (for example, 4010VICS is a subset of 4010) Trading Manager will first attempt to use Type Tree and/or structure definitions for the specific subset. If these structure definitions are not found, Trading Manager will use the Type Tree and/or structure definitions for the related ANSI X12 version as long as there also are trade links specified for the functional group ids used in the custom X12 version that are using the ANSI X12 versions. When implementing custom X12 versions, create an **x12mail** Type Tree that includes all of the custom X12 versions to be processed.

You can bypass creating structure records but you should define acknowledgments (997s) to report on the group or transaction level only.

Type trees

For each custom X12 version, a Master Type Tree must be created that follows specific rules so that the EDI Wizard can generate the **x12mail** and **x12user** type trees. These rules are as follows:

- The overall organization of the Type Tree must conform to the organization of the X12 Type Trees contained in the IBM WebSphere Transformation Extender Packs for EDI (See the X12 type trees documentation.)
- Under **Inbound/Outbound Partner Funct'l Group ANSI EDI** the name of the custom group must be **F** followed by the custom version (excluding the leading zeroes). For example, to support 004010VICS the name of the group would be **F4010VICS**. Characters in the version name not supported by the Type Designer should be removed, for example, for **R2/8** the name of the group would be **FR28**.
- The component rule for the GS segment inside the custom group has to have the LEFT function removed and the literal must be the complete version string as it appears in the GS Segment. For example, for **4010VICS** the component rule must read: **VersionReleaseIndustryIDCd Element:\$ = "004010VICS"**
- Under **ANSI EDI** the category that contains the transactions for the custom group must be **V** followed by the custom version (excluding the leading zeroes). For example, to support 004010VICS the name of the category would be **V4010VICS**. Characters in the version name not supported by the Type Designer should be removed, for example, for **R2/8** the name of the category would be **VR28**.

Any changes to the transactions, segments, and/or elements inside the custom category will be picked up by the EDI Wizard and included in the resulting **x12mail** and **x12user** trees.

Versions that contain characters not supported (R2/8) must be manually added to the **x12mail** and **x12user** trees.

Partner Manager

About this task

To define tradelinks using custom X12 versions perform the following steps:

Procedure

1. Start Partner Manager and select **File > Utilities**.
The **Utilities** Navigator opens.
2. Select **Configuration > Standards**.
The Standards Maintenance window opens.
3. Select **X12 Versions** from the **X12 Standards** drop-down menu.
4. Enter the custom X12 version in the **Version** field as it shows in the GS Segment (excluding leading zeroes). For example: to support 004010VICS enter **4010VICS**.
5. Click **Save**.
6. Perform one of the following:
 - copy the Transactions and Functional ID's from any other X12 version
or
 - enter the Transactions and Functional ID's manually (see the Partner Manager documentation for more information).
If the custom version being implemented is a subset of an ANSI version (for example 4010VICS is a subset of 4010) then you do not need to create structure records. If it is not (for example R2/8), then create structure records and load them into Partner Manager. See "[Creating Structure Records](#)".
7. Create X12 tradelinks that use the custom X12 version.
8. Run the EDI Wizard and point to the Master Type Trees created previously.
Versions that contain characters not supported (R2/8) must be manually added to the **x12mail** and **x12 user** trees.

Message Manager

About this task

Once the EDI Wizard has been executed and an **x12mail** Type Tree created, the **RunMaps** system must be deployed. See "[Installing and Using Message Manager](#)."

Creating structure records

About this task

To create Partner Manager Structure Records:

Procedure

1. Locate the **StructureValidation.detl** file, in the **tmgr_vn.n\pmgr** folder and make a copy of it for the custom X12 version. Ensure that the filename extension is **.detl**.
2. Select a set of entries that most resemble the transactions under the custom X12 version you are implementing. Delete all other records.

3. In the records selected, change the first 10 characters in each row to represent the custom X12 version as it shows in the GS Segment (excluding leading zeroes). Ensure that the size of the field remains the same, for example: replace 4010_____ into 4010VICS__ (where _ represents spaces).
4. Ensure that columns 11 and 12 contain the string **T1** then save the updated file.
5. Start Partner Manager and select File> Utilities.
6. Select Configuration>Standards to open the **Standards Maintenance** window.
7. Choose **X12 Structure Validation: Import** from the **X12 Standards** drop-down menu.
8. Click Find File to browse to the directory where you placed the file that you created previously and select the file.
9. Click Begin Import to complete the process.

Results

Contact Trading Manager Support if more complex changes to the Partner Manager Structure Records for the custom X12 version are required.

TPEC to Trading Manager conversion

- [TPECAddressBook.xml file](#)
- [TPECPostOffices.xml file](#)
- [TPECTradelinks.xml file](#)
- [TPEC2CMConfig.txt file](#)
- [Running the TPEC to Trading Manager](#)
- [Setting up the TPEC2CM conversion map](#)
- [Running the "ConvertTPEC2CM" executable map](#)
- [TPEC control number conversion](#)
- [Running the Partner Manager XML Autoload Utility](#)
- [Warnings and error messages](#)
- [TPEC header trailer plug-in for Trading Manager](#)

If you are converting from TPEC, the instructions provided here describe the changes made to Message Manager to support header and trailer data. These changes are supplied as a plug-in to Trading Manager for handling the ability to maintain and wrap outbound messages with header and trailer information.

Overview

The Trading Partner E-Commerce (TPEC) to Trading Manager Conversion tool aids in the migration of X12 trading information from TPEC to Trading Manager. The tool automates the transfer of information related to partners and relationships and reduces the amount of tedious and error prone typing that you are required to do. This tool uses as input a file created using the TPEC Partner Extract and creates three files for Trading Manager XML Autoload. It also creates a report with warnings and errors found during the conversion.

After the tool is executed and the information is loaded into Trading Manager, some level of editing/customization will become necessary. The tool does provide for a configuration file that allows you to set parameters specific to your particular conversion.

TPECAddressBook.xml file

This file contains Partner definitions for Trading Manager. They are grouped into "external partners" (the sending partner of an inbound transaction) and "internal partners" (the receiving partner of an inbound transaction) each one with a unique ISA qualifier and ID combination. It is generated from the Relationship records in TPEC with the partner name from TPEC Partner records, ISA information from the TPEC Interchange records (if present, see the TPEC2CM Configuration File documentation), and GS information from the TPEC Relationship record. The conversion tool will generate as many Trading Manager partners as possible, even if they do not have any TPEC trading specifications defined. The TPEC alias records are not used by the conversion. Please note the following about internal and external partners.

- [Internal partners](#)
- [External partners](#)

Internal partners

- ISA definitions are defaulted since Trading Manager does not validate them.
- Partner names are built from the **ISACompanyPrefix** field in the TPEC2CM Configuration File and the ISA qualifier and ID.
- Application Partner names are built from the **GSCompanyPrefix** field in the **TPEC2CM** Configuration File and the GS ID.

External partners

- ISA definitions are migrated from the TPEC Interchange Records.
- Interchange Control Standard is hard coded to "U" for versions previous to 00402 and to the value specified by **DefaultRepetitionChar** in the TPEC2CM Configuration File for versions 00402 and higher.
- Partner Names are build from either the Corporate ID in the Partner Record or, if not found, from either the TPEC Interchange or TPEC Relationship record as specified by **DefaultPartnerNameFrom** in the TPEC2CM Configuration File. If there are duplicates the ISA Qualifier and ID are appended to the name.
- Application Partner names are built from the Relationship name. If there are duplicates the GS ID is appended to the name.

TPECPostOffices.xml file

This file contains Post Office definitions for Trading Manager. They represent logical inputs and outputs of data and there is no equivalent representation in TPEC. Since they are required, they are generated as follows:

- **Get Post Office:** A single post office is defined for all inbound and outbound data going into Trading Manager. Its nickname is hard coded to "EDI" and the path to the directory is defined in the TPEC2CM Configuration File.
- **Outbound Put Post Office:** A single post office is defined for all outbound data going out from Trading Manager to the external partners. Its nickname is hard coded to "EDO" and the path to the directory is defined in the TPEC2CM Configuration File.
- **Inbound Put Post Office:** One inbound put post office is defined for each unique combination of inbound Functional Group and Version/Release. This simplifies the implementation of the user maps need to convert inbound EDI data into the application format. The nickname is the two-character Functional Group ID plus one character representing an index for the Version/Releases found. These post offices are built from the TPEC Inbound Trading Specs. The conversion tool verifies that each Transaction and Version/Release combination is defined in Trading Manager.

TPECTradelinks.xml file

This file contains Tradelink definitions for Trading Manager. It describes the Functional Groups traded between the partners and has to be loaded last into Trading Manager since it requires CM Partner and Post Office records. It is loosely related to the TPEC Trading Specs records that represent Transactions traded between the Partners. Please note:

- [Inbound functional groups:](#)
- [Outbound functional groups](#)

Inbound functional groups:

- Both Test and Production Interchanges will be accepted and routed.
- Functional Acknowledgments will be configured based on the settings in TPEC. 999 Transaction Acknowledgments are not supported by the conversion tool and will be replaced by 997 Transactions.
- Functional Acknowledgments are configured to have the same GS Version/Release that the acknowledging inbound Functional Group have.
- HIPAA transactions are hard coded for Type 1 validation if the Acknowledge option is turned on in TPEC.
- Validation By is defaulted to Transaction.
- Tradelinks for Inbound Functional Acknowledgments are not generated (not required for CM Internal processing) unless the option is selected in the TPEC2CM Configuration File.
- All transactions in a Functional Group will be validated and routed.

Outbound functional groups

- Outbound Acknowledgements are generated only for those Inbound Tradelinks that have the Acknowledgment option turned on.
- ISA Control Number Scheme is set to "I" (by Interchange).
- GS Control Number Scheme is set to "T" (by Trading Partner).
- ST Control Number Scheme is set to "R" (Relative to the Functional Group).
- Application Format is set to "P" (Partial).
- Validate By is defaulted to "I" (Interchange).
- Last used Control Numbers are not converted from TPEC.

TPEC2CMConfig.txt file

The purpose of the TPEC2CM Configuration file is to allow the user to define parameters to drive the conversion. There are two types of parameters: site-specific constants and override values for fields in the ISA segment. Please note that the beginning and end of the file should not be changed, but entries and comments can be added in between.

Site-specific constants are required to complete the conversion process: they provide values need to successfully create the XML files for the Partner Manager XML Autoload utility. The following site-specific constants are available:

- ISACCompanyPrefix
 - Company Name Prefix for CM internal ISA Partners. Max length allowed is 15 characters.
- GSCompanyPrefix
 - Company Name Prefix for CM internal GS partners. Max length allowed is 15 characters.
- DefaultPartnerNameFrom
 - In TPEC Relationships are supposed to be linked with a Partner definition. In case one is not found the conversion program should default to Interchange (I) or Relationship (R) Records for the CM Partner Name.
- InboundPUTFolder
 - Single folder path for all Inbound PUT Post Offices. Make sure you include the last "\" (or "/" if Unix).
- OutboundPUTFolder
 - Single folder path for the EDO Outbound PUT Post Office. Make sure you include the last "\" (or "/" if Unix).
- GETFolder
 - Single folder path for all inputs into Trading Manager. Should match the StampAndSortFile input directory. Make sure you include the last "\" (or "/" if Unix).
- IgnoreInbound997
 - Inbound 997's are processed internally in Trading Manager. Tradelinks are only required if the user wants to process them into some application of their own. Values are Y/N
- DefaultRepetitionChar

- The ISA Control Standard is hard coded to "U" for ISA Control versions less than "00402". For "00402" and higher this Repetition Character will be hard coded. Character must be defined in ASCII HEX.

Override values are not required. The XML Autoload will verify that the old values for these parameters are valid in Trading Manager and fail if not. If they are not valid the user has the option of adding the failing values to Trading Manager or (by using these parameters) specifying replacement new values that are valid in Trading Manager. The following override values for fields in the ISA segment are available:

```
IchgQual:XX|YY
ElementSeparator:XX|XX
SubElementSeparator:XX|YY
SegmentTerminator:XX|YY
IchgCtrlVerNo:XXXXX|YYYYY
```

In the above values:

- X represents the old TPEC value; Y represents the new CM value.
- Old and New values are fixed length.
- All Old and New values have to be represented in ASCII HEX with the exception of IchgCtrlVerNo that has to be a text string.
- Old SegmentTerminator value is one character long. New value can be up to three characters long represented with a space in between. For example, a terminator of "~<CR><LF>" would be represented as: SegmentTerminator:0A|7E 0D 0A

Running the TPEC to Trading Manager

About this task

Creating the TPEC Partner Extract file

About this task

The conversion tool requires a specific file from TPEC. It can be generated by running the following JCL:

```
//USERIEXT JOB ()
//*****
//** TPJ - DATABASE EXTRACT
//*****
//*
//DBEXT EXEC PGM=TPBDRIVE,REGION=4M,
// PARM='DATABASE-EXTRACT,PROD'
//REPORT DD SYSOUT=*
//TPSVSRT DD SYSOUT=*
//SYSOUT DD SYSOUT=*
//SYSPRINT DD SYSOUT=*
//SYSUDUMP DD SYSOUT=*
//TEPLIB DD DSN=TSI.TP31.GAI.XA.LOAD,DISP=SHR
//STEPLIB DD DSN=TSI.TP31.DEV.XA.LOAD,DISP=SHR
// DD DSN=TSI.TP31.DST.XA.LOAD,DISP=SHR
//TPFORM1 DD DSN=TSI.TP31.D3.DEV.TPFORM1,DISP=SHR
//TPDATA1 DD DSN=TSI.TP31.D3.DEV.TPDATA1,DISP=SHR
//TPFIL11 DD DSN=TSI.TP31.D3.DEV.TPFIL11,DISP=SHR
//TPFIL21 DD DSN=TSI.TP31.D3.DEV.TPFIL21,DISP=SHR
//TPFIL31 DD DSN=TSI.TP31.D3.DEV.TPFIL31,DISP=SHR
//TPSTWK01 DD UNIT=SYSDA,SPACE=(CYL,(10,1))
//----- *
// EBCDIC OUTPUT FILE USED BY THE TPEC2CM CONVERSION TOOL *
//----- *
//EXTRACT DD DSN=USER.TP31.EXTRACT,
```

```

// DISP=(NEW,CATLG),
// UNIT=SYSDA,
// SPACE=(TRK,(50,50),RLSE)
/* -----
/* ASCII OUTPUT FILE NOT USED *
/* -----
//DBXTBLE DD DSN=USER.TP31.DBXTBLE,
// DISP=(NEW,CATLG),
// UNIT=SYSDA,
// SPACE=(TRK,(50,50),RLSE)
//CONTROL DD *
EXTRACT PARTNERS
/*

```

Once the job completes FTP the file created by the EXTRACT DD name in ASCII mode and place it in the **C:\install_dir\tmgr_vn.n_beta\platform_support\tpec2cm** folder. Records are fixed length, make sure no trailing spaces are removed during the FTP.

Setting up the TPEC2CM conversion map

About this task

Before you build the maps that make up this tool please configure/verify:

- The tool uses "update.mdq" from the mmgr directory where the Message Manager source is installed. It provides access to a Trading Manager database that the tool requires. It needs to be configured to point to the target CM database for this conversion process. Due to expected amount of information in the extracted data, MS Access should *not* be used. A test database of the same type as the final target is recommended.
- The target database should not contain any entries in the Address Book, Post Offices, nor Tradelinks.
- The target database has to have the X12 Structure File loaded. If the TPEC system has HIPAA transactions the HIPAA Structure File should also be loaded into the target database before XML Autoload is executed.
- The tool also uses "cmsvc.dll", "m4ecmgmt.mtt", and "mmgr_library.mms" from the mmgr directory where the Message Manager source is installed. Output card 1 (ProcessCB) of executable map "ConvertTPEC2CM" has to point to the mmgr folder.

Running the "ConvertTPEC2CM" executable map

Procedure

- Open **tpec2cm.mms** and locate executable map: **ConvertTPEC2CM**
- Edit input card 1 and point to EXTRACT file FTP'ed from the Mainframe.
- Build ALL maps.
- Run the **ConvertTPEC2CM** executable map.

Results

The map will create three files for the XML Autoload:

- TPECAccountBook.xml**
- TPECPoOffices.xml**
- TPECTradelinks.xml**

The map will also create two log files with information about the conversion:

- TPEC to CM Conversion_xxxx_yyyy_session.xml]**
- ConvertTPEC2CM.log**

The following files are intermediate files useful for conversion debugging:

- PartnerReportFile.txt**
- CleanPartnerReportFile.txt**
- PostOfficeReportFile.txt**
- TradelinkReportFile.txt**
- CleanTradelinkReportFile.txt**

TPEC control number conversion

It is important to note that the TPEC conversion described here does not convert any existing control numbers into the Trading Manager database. The steps provided here describe how to convert those numbers.

It is also important to note that the Control Number Conversion is limited to only X12 Outbound ISA and GS control numbers. The procedure provided here does *not* convert X12 Inbound control numbers, or any EDIFACT/TRADACOMS control numbers. In addition, this procedure does not convert any ST control numbers.

WARNING: Running these steps more than once will overwrite any existing X12 Outbound Control Numbers.

WARNING: This program initializes all X12 Outbound Trade Links in the database to have a GS Scheme of **Increment by Application Partner**, which is the default value. All links are set in this way, even if no control numbers are found in the input **extract** file. Do not use this program if you use a different GS control number scheme.

To perform the conversion, perform the following steps:

1. Confirm the TPEC to Trading Manager conversion as previously described in this documentation has been accomplished successfully.
2. The control number update utility is **CtlNumUpdateForTPEC.exe**. This file is located in your **pmgr** directory. Locate and run this file.
3. Select the same extract file used in the TPEC to Trading Manager conversion as described previously in this documentation.
4. Select **Validation Run** the first time to check your entries. Select **Update** to actually perform the conversion on the database.
5. Enter the database connection information.
6. Select **Begin**.
7. When the program completes, you can print the report by selecting the **Print Report** button, or you can copy the report to your Windows Clipboard by using the context (typically right mouse click) menu on the report contents. Use the **Select All** and **Copy** in the context menu to copy the report to your clipboard. The report lists any records it cannot find, such as IT and RS records in the extract.

Running the Partner Manager XML Autoload Utility

About this task

The XML files created by the conversion map have to be loaded in the order they are listed above. Please note that you might have to run the conversion process several times before you get a clean run. Therefore is best to make a backup of your target database before loading any data. Perform the following steps for each file:

Procedure

1. Copy file **tm82.dtd** from **tmgr_vn.n\pmgr** to the **tpec2cm** folder.
2. Start program **PMAutoLoad.exe** in the **pmgr** folder.
3. Select the appropriate XML input file using the **Find** button.
4. Configure the destination database to match the one configured in the **tpec2cm.mdq** file.
5. Select Validate Only; Do not change Database, **Allow Add**, **Detailed Report**, and **Reject data that exceeds size**.
6. Click Begin.

Results

At this point the XML Autoload will validate the input file for format and content using the target database and any error are reported. Usually there are two types of errors:

- ISA field values that are not supported by Trading Manager. Either update the **TPEC2CMConfig.txt** file or, using the Standards option in Partner Manager, add the unsupported values to Trading Manager.
- Duplicate ISA Qual/ID combinations. This can happen when the new value for "IchgQual" in the TPEC2CM Configuration file is already used by some other partner. This can be resolved by either selecting a new value in the configuration file that is not used or changing the ISA information in TPEC for one of the duplicate partners.

Once the files are valid then proceed with the load by selecting "Run Load" in the XML Autoload. The most typical error during this step is:

- Missing structure files in Trading Manager. Either the structure files have not been loaded into the target database of the TPEC system is using non-standard transaction-version/release combinations. The user must either load or create and load structure files indicated by the errors.

Warnings and error messages

Warnings are created to log issues during processing for information purposes only. Error messages are shown in the following list in bold face type, with an explanation for each following it in regular face type.

- **ID="10001-W">>Relationship: GSSENDER Partner Name: 'ABC COMP' is a duplicate. Replaced with: 'ABC COMP ZZISASENDERID'**
 - In the process of creating external partner names two different ISA Qualifier/ID generated the same Partner Name for Trading Manager. The are made unique by adding the ISA Qual/ID to the name.
- **ID="10003-W">>Relationship: CONTROL Interchange: BENCH ISA Qualifier: AD is invalid. Replaced with: 19**
 - The conversion tool found a TPEC ISA Qualifier that matched an old value in the TPEC2CM Configuration file and replaced it with the indicated new value.
- **ID="10004-W">>Relationship: CONTROL Interchange: BENCH ISA Control Version: 00202 is invalid. Replaced with: 00200**
 - The conversion tool found a TPEC ISA Control Version that matched an old value in the TPEC2CM Configuration file and replaced it with the indicated new value.
- **ID="10005-W">>Relationship: BSS Interchange: BSS Sub Element Delimiter: D7 is invalid. Replaced with: 01**
 - The conversion tool found a TPEC Sub Element Delimiter that matched an old value in the TPEC2CM Configuration file and replaced it with the indicated new value.
- **ID="10006-W">>Relationship: CONTROL Interchange: BENCH Element Delimiter: D7 is invalid. Replaced with: 02**
 - The conversion tool found a TPEC Element Delimiter that matched an old value in the TPEC2CM Configuration file and replaced it with the indicated new value.
- **ID="10007-W">>Relationship: GAPFUNC Interchange: GAPTEST Segment Terminator: B9 is invalid. Replaced with: OD OA**

- The conversion tool found a TPEC Segment Terminator that matched an old value in the TPEC2CM Configuration file and replaced it with the indicated new value.
- **ID="10008-W">Relationship: GSGREG Could not find Partner records for Interchange: ABC. Using ABC**
 - The conversion tool found a Relationship in TPEC that had an associated Interchange record but could not find a Partner record. Usually happens when relationships are defined but no actual Trading Specs are associated to them. A CM Partner definition is still being created.
- **ID="10009-W">Relationship: CONTROL1 Found new GS ID: 'GSRECV'. Created new Application Partner called: 'CONTROL1 GSRECV'**
 - The conversion tool found multiple Relationship records that contain the same GS ID.
- **ID="10010-W">Relationship: KAYBGIN Interchange: KAYBGIN ISA ID: 'FDEX' is a duplicate of an External Partner. Replaced with: '13#FDEX'**
 - While creating internal partners it found ISA information that matches an already defined external partner. The tool created a dummy ISA ID to allow the load but this should be analyzed.
- **ID="10011-W">Relationship: KAYBGIN TradingSpec: 000071 Outbound FA: 999 not supported by this conversion tool. Replaced with 997**
 - The conversion tool does not support 999 Acknowledgments.
- **ID="10012-W">Relationship: HELEN TradingSpec: 000056 FA Relationship: HELEN FA to third party not supported by this conversion tool. Using current Relationship**
 - The conversion tool does not support acknowledgments to third parties. The tool used the current relationship to allow the load but this should be analyzed.
- **ID="20001-C">Relationship: KAYBGIN TradingSpec: 000071 Inbound Transaction: 875 Version/Release: 3020UCS not supported in Trading Manager**
 - Inbound Transaction-Version/Release not supported. No tradelink will be generated. Users have to add this Transaction-Version/Release to Trading Manager to migrate this Trading Spec.
- **ID="20002-C">Relationship: KAYBG TradingSpec: 000166 Outbound Transaction: 875 Version/Release: 3020UCS not supported in Trading Manager**
 - Outbound Transaction-Version/Release not supported. No tradelink will be generated. Users have to add this Transaction-Version/Release to Trading Manager to migrate this Trading Spec.
- **ID="20003-C">Relationship: ERISCO TradingSpec: 000045 Outbound FA: 997 Version/Release: 3032ERISCO not supported in Trading Manager**
 - Outbound Transaction-Version/Release not supported. No tradelink will be generated. Users have to add this Transaction-Version/Release to Trading Manager to migrate this Trading Spec.

TPEC header trailer plug-in for Trading Manager

If you are converting from TPEC, the instructions provided here describe the changes made to Message Manager to support header and trailer data. These changes are supplied as a plug-in to Trading Manager for handling the ability to maintain and wrap outbound messages with header and trailer information.

- [TPEC plug-in requirements](#)
- [Planning the conversion](#)
- [TPEC installation instructions](#)
- [Description of mailbox_bundle components](#)

The following documentation describes those components of the **mailbox_bundle**.

TPEC plug-in requirements

The following conditions are required prior to installing the TPEC plug-in:

- Trading Manager V8.2, or higher is installed
- A Personal Computer with the Trading Manager GUI's installed and running. The GUI's include:
 - Partner Manager
 - PM Copy
 - XML Autoload Utility
- Write/delete access on your Message Manager system
- Trading Manager connecting to either an Access or DB2 database
- The existing "Trading Partner EC to Trading Manager Conversion Tool" has been performed on the databases
- You must have the extract file that was used to perform the existing "Trading Partner EC to Trading Manager Conversion Tool" on the databases

Planning the conversion

The following components are essential to successful TPEC to Trading Manager conversion:

- Trading Partner EC to Trading Manager Conversion Tool – you must have first used this tool to create an XML Autoload file to populate the Trading Manager database. This step is included in the base related product and is not described in documentation provided here.
- Header and trailer conversion must be complete – new tables are populated with header and trailer information from the mailbox records in the TPEC extract used above. This program also creates new 'put' post offices in the Trading Manager database, one for each unique mailbox assigned to an existing trade link. This component of the plug-in is shipped

All components of the conversion can be run only once with the same extract file. If you attempt to run the Trading Partner EC to Trading Manager Conversion tool with the extract file more than once, all existing partners will be rejected in the XML Autoload step. Using the same extract with the header trailer conversion program will either delete and re-create the header trailer data or ignore it, depending on a setting of an on-screen option.

TPEC installation instructions

The procedures described in the following documentation must be performed in the order specified. If an error is returned in any step, do not continue. Contact IBM support. Follow these steps to perform TPEC installation:

1. Under your Trading Manager **mmgr** subdirectory, create the following two subdirectories:

- bundle
 - mailbox_bundle
2. Copy the following files from the **mmgr** directory to the newly created **mailbox_bundle** subdirectory:
- poheader.mdq
 - bundle.msd
 - bundle.mms
 - config.mtt
 - mailbox.conf
 - po_header.mtt
 - potree.mtt

The settings for the system are described as follows:

Description of **mailbox_bundle** components

The following documentation describes those components of the **mailbox_bundle**.

- **mailbox.conf**
This file contains 2 lines of code that indicate where the data to be appended (bundled) prior to header trailer wrapping is to be located, and where it will be located after it is wrapped.
- **config.mtt**
The **config.mtt** file is a type tree that describes the config.txt file derived from the **mailbox.conf**.
- **poheader.mdq**
The **poheader.mdq** is the component that specifies connection to the database where the mailbox information is located.
- **potree.mtt**
The potree.mtt file is the type tree that contains all of the different elements to be used throughout the conversion maps.

mailbox.conf

This file contains 2 lines of code that indicate where the data to be appended (bundled) prior to header trailer wrapping is to be located, and where it will be located after it is wrapped.

The value for both of these lines is:

install_dir\tnmgr_vn.n.n\mmgr\bundle

where *n.n.n* represents the current version of Trading Manager

Changes can be made to the code to point to a different directory structure, but other changes will need to be made in order to accommodate such a change. Also, it is restricted to a file-based destination.

config.mtt

The **config.mtt** file is a type tree that describes the config.txt file derived from the **mailbox.conf**.

poheader.mdq

The **poheader.mdq** is the component that specifies connection to the database where the mailbox information is located.

This should be the same as the **update.mdq** used for the Trading Manager system. A query is defined in this **mdq** file to obtain all of the post offices with header trailer information.

potree.mtt

The potree.mtt file is the type tree that contains all of the different elements to be used throughout the conversion maps.

Partner Manager

The documentation provided here describes Partner Manager, the graphical user interface component of Trading Manager. Partner Manager is used to create a comprehensive database of an organization's electronic commerce (e-commerce) information and to manage its e-commerce activity.

- **Overview**

Partner Manager is a data management system for defining and viewing e-commerce activity. Typically, an e-commerce administrator such as an EDI coordinator uses Partner Manager. As part of Trading Manager, Partner Manager is used to define your trading relationships with your e-commerce partners. Information is

maintained on each trading partner including address information, contact persons, and the individual organizational entities with which you trade business information.

- [Getting started](#)

- [Address books](#)

- [Post Offices](#)

- [Trade links](#)

Trade links contain the partner addresses, post offices used, standards information, control and tracking information, and current test or production status for each individual document that you exchange with that partner

- [Reports](#)

- [Resending E-commerce data](#)

Inbound and outbound data that has been processed by Trading Manager can be viewed and edited. Run the Interchange traffic report to resend the interchange immediately or edit/resend the interchange data.

- [Utilities](#)

- [EDI Wizard](#)

- [Database copy utility](#)

- [Managing traffic data](#)

Overview

Partner Manager is a data management system for defining and viewing e-commerce activity. Typically, an e-commerce administrator such as an EDI coordinator uses Partner Manager. As part of Trading Manager, Partner Manager is used to define your trading relationships with your e-commerce partners. Information is maintained on each trading partner including address information, contact persons, and the individual organizational entities with which you trade business information.

- [Uses for Partner Manager](#)

- [Theory of operation](#)

- [Trade Links and Partner Manager](#)

- [Partner Manager interface](#)

Uses for Partner Manager

Use Partner Manager to perform the following tasks:

- Define post offices; direct links using File Transfer Protocol (FTP), Value Added Networks (VANs), or e-mail to enable you to communicate with your partners
- Define the trading relationships with your EDI partners, including details such as the individual document level for its version, control numbering schemes, tracking, error handling, status, and post office used
- Provide audit and control information collected in the Trading Manager database, which you use to monitor your organization's ongoing inbound and outbound e-commerce transaction activity
- Be alerted to error conditions in e-commerce inbound or outbound processing

Theory of operation

Use Partner Manager to define:

- One or more internal trading partners
- One or more external trading partners
- Two or more post offices
- Trade links to associate internal trading partners with external trading partners
- Information the two trading partners exchange

The information from Partner Manager is shared with the Launcher component of Trading Manager for validating, tracking, and routing, e-commerce data. Information about this e-commerce activity is then incorporated into the Partner Manager database for monitoring and reporting.

Trade Links and Partner Manager

A primary Partner Manager function is the definition of trading relationships, also known as trade links. A trade link consists of four fundamental components:

| | |
|---------------------------------|--|
| Internal Trading Partner | An internal trading partner is defined in the Address Book navigator. This information identifies the EDI settings for an internal entity that sends and receives e-commerce data to and from another organization. Internal trading partners answer the question <i>Who are we?</i> |
| External Trading Partner | An external trading partner is also defined in the Address Book navigator. This information identifies the EDI settings for an entity with which internal trading partners exchange data. External trading partners answer the question <i>Who do we exchange data with?</i> |
| Post Office | A post office identifies where data to be processed by Trading Manager is located and the necessary configuration settings that Trading Manager needs to access that data. Post offices are defined from the Post Offices navigator. Post offices answer the question <i>Where is the data retrieved and routed when its processing is complete?</i> |
| Application | An application identifies the type of messages or documents that are exchanged between an internal and external trading partner. A minimum of one application is defined as part of the Link Wizard from the Trade Links navigator. Other applications can then be defined from the Trade Links navigator. Applications answer the question <i>What types of information do we exchange with a particular trading partner?</i> |

Partner Manager interface

The Partner Manager interface includes navigators, forms, the toolbar, the menu bar, command buttons, an Editor, reports, and right-click context menus.

Partner Manager functions are available from the main menu, right-click functionality, and toolbar.

- [Navigators](#)
- [Toolbar](#)
- [Form area](#)
- [Audit Info button](#)

Navigators

Partner Manager provides four navigators to help you define and view e-commerce information:

- **Address Book** navigator enables you to define trading and application partners. Use this navigator to store information about your internal organization's set of trading partners, your external trading partners, and their departments or application groups
- **Post Offices** navigator enables you to specify routing instructions for inbound data, outbound data, and error reporting. You can define the mail routes, FTP connections, e-mail addresses, or value added networks that you use to exchange data with your trading partners
- **Trade Links** navigator enables you to define trading partner relationships. Trade links connect a trading relationship with specific instructions for routing, error handling, and tracking. Partner Manager wizards help you construct and maintain trade links quickly and easily. Trade links connect you to your trading partner for each document that you trade with that partner.
- **Utilities** navigator contains utilities that allow you to make configuration changes, change database settings, organize trading partner subfolders and to modify security information.

Toolbar

The toolbar is a part of the Partner Manager window that provides you with quick access to various tools to invoke Partner Manager actions.

Form area

The form area displays dialog boxes and forms when you click a navigator folder. You enter trade and application partner, post office, trade link, and other information in the dialog boxes and forms that display here.

Audit Info button

About this task

Audit information is available throughout the Partner Manager; for example, when you edit trading or application partners, the edit dialog box includes an **Audit Info** button. The audit facility tracks the creation and creator of each entity by using a date, time stamp, and the user ID of the person that originally created the entry. This data is also stored in the database to provide an audit trail of all changes. All subsequent changes are logged in the audit database with the date, time, and user ID of the party making the change.

To view audit information:

Procedure

1. Click Audit Info on forms where the button is available.
The **Audit Information** dialog box appears.
2. View the information.
Click OK to close the **Audit Information** dialog box.

Getting started

Procedures described in this chapter enable you to:

- Start Partner Manager from the Start menu
- Start Partner Manager from the Windows desktop using a database-specific icon
- Select a database different from the default Partner Manager database
- Create, edit, and delete a database profile
- Merge and copy databases
- [Starting Partner Manager](#)
- [System requirements](#)

Starting Partner Manager

About this task

Note: Before you start Partner Manager, ensure that you are using the latest open database connectivity (ODBC) drivers from your database vendor. For example, if you are using the Microsoft ODBC Driver for Oracle, you must also install the Oracle Client on the client machine.

- You start the Partner Manager interface from the Windows Start menu or from an icon on your desktop.
- You can also customize the Partner Manager startup configuration to use different database profiles.
- [Starting Partner Manager from the Start menu](#)
- [Starting Partner Manager using a database-specific desktop icon](#)

Starting Partner Manager from the Start menu

About this task

The Partner Manager application may be started from the Start menu or a desktop icon.

To start Partner Manager from the Start menu:

Procedure

From the Start menu, select Programs > IBM WebSphere Transformation Extender n.n > Trading Manager > Partner Manager
Where n.n is the product version number.

Note: If user and group security are used, you must login with a valid user ID. See ["Utilities"](#).
The Partner Manager window opens.

Starting Partner Manager using a database-specific desktop icon

About this task

This technique is useful when you need to run multiple instances of Partner Manager or want to create an icon for each database you use. Information about the current profile in use is available by choosing **About** from the Help menu.

To start Partner Manager by using a database-specific desktop icon

Procedure

1. Create a new database profile.
2. Open Windows Explorer.
3. Select Desktop in the folder navigator.
4. From the File menu, select New > Shortcut.
5. Enter the following in the **Command Line** field of the **Create** Shortcut dialog box:

```
install_dir\tmgr_vn.n\pmgr\partnermanager.exe profile
```

where n.n represents the current Trading Manager version, and *profile* is the case-sensitive database profile you created. See ["Utilities"](#) more information.

6. Click Next and name the shortcut.
7. Click Finish.
8. To start Partner Manager, double-click the shortcut icon on the desktop.

Note: If the profile is incorrect or does not exist, the Partner Manager application uses the most recent valid database connection.

System requirements

The minimum requirements to install and run Partner Manager are:

- Design Studio is required to use the EDI Wizard
- System data source name
- [Database configuration](#)
- [System Data Source Name \(DSN\)](#)

Database configuration

A default Microsoft Access database named **tmgrnn.mdb** is installed in the *install_dir\tmgr_vnn\pmgr* folder as part of the Partner Manager installation program (where *nn* and *n.n* refer to the current Trading Manager version).

The Utilities enables you to select a different database. Other Partner Manager utility procedures enable you to:

- Create, edit, and delete a database profile
- Merge databases
- Copy databases

Note: When entering data in Partner Manager be aware that case sensitivity can vary by Database Vendor and even by database instance. For example, Microsoft Access considers "Acme" to equal "ACME"; most other databases consider these fields unequal. Consult your DBA to determine if your database is case sensitive and plan your database entries accordingly, especially if using Partner Manager's copy and merge functionality between database vendors.

System Data Source Name (DSN)

A system data source name (DSN) is a user-defined name that links a database driver to a database. You must configure a DSN to access the default *install_dir\tmgr_vn.n\pmgr\tmgrnn.mdb* database (where *n.n* and *nn* represent the current Trading Manager version number). The Trading Manager Installation documentation describes how to create and configure a system DSN.

The DSN enables direct access to the database using an application like Partner Manager. When configuring open database connectivity (ODBC) DSN, you must create a DSN for each database to which you connect.

Address books

The **Address Book** navigator is used to define the internal and external trading partners at the corporate, application, departmental, or divisional levels.

Procedures described in this chapter enable you to define:

- Trading partners in your organization
- Trading partners in organizations you do business with

From the **Address Book** navigator, you define trading partners internal and external to your organization. The **Address Book** navigator contains links for these trading partners. Internal links represent how your company is organized to manage your e-commerce relationships. Using these links, you define your company's internal partners, who represent your company as part of a trading partner relationship. External links enable you to define your trading partners in other companies or entities. These are the organizations that you do business with and represent the external trading partnerships that you establish.

- [Trading partner and application partner relationships](#)
- [Trading partners and application partners](#)
- [Configuring internal and external address books](#)
- [Application partners](#)
- [Managing Administrative Contacts](#)

Trading partner and application partner relationships

The trading partner and application partner relationships define how the trading partners and application partner associations are defined in Partner Manager.

- [One trading partner with one application partner](#)
- [Many trading partners with one or more application partners](#)
- [One trading partner with many application partners](#)

One trading partner with one application partner

Your organization may choose to have a single *window on the world* in terms of your EDI profile: that is, defining all EDI address envelopes with a single company EDI address. Typically, companies with a centralized e-commerce function that manages all relationships use this approach.

In Partner Manager, a single trading partner representing a company may be defined with one application partner that represents the entire organization.

Many trading partners with one or more application partners

You can maintain multiple relationships with external trading partners. For example, you may have autonomous divisions who wish to do business under their own name but still share a single EDI system administered by an e-commerce group. In this case, define a new trading partner for each internal entity that requires its own address.

In Partner Manager, a trading partner may be defined for each division of a company. For example, one for Domestic Operations and one for International Operations. One application partner is defined for each trading partner.

One trading partner with many application partners

You may also want to maintain a single company address for the corporate level and separate addresses for each department or division. Define a single internal trading partner name and use the application partner entries to enable you to connect a given department with its trading partners, yet have all mail processed through the corporate partner relationship first. For example, you can define specific trading relationships with your dealer/distributor chain, your suppliers, and your customers. You may also configure relationships for each major application: the Order Entry department as an entity receives all incoming orders; the Billing department processes invoices; and Human Resources handles healthcare enrollments and claims processing.

In this case, you configure a trading partner for the company and an application partner for each division.

Trading partners and application partners

The partners with which your organization exchanges e-commerce data can also be organized using the trading partner and application partner scenarios described in the previous sections.

- [Internal address book](#)
- [External address book](#)
- [Internal trading partner and application partner](#)

Internal address book

Use the **Internal Address Book** to specify address information for partners within your organization.

- **Admin** - contains information about administrative contact persons associated with your organization's e-commerce programs, particular applications or trading partners, Trading Manager, and so on.
- **EDIFACT** - contains the definition of entities within your organization that maintain relationships with external trading partners, based on the Electronic Data Interchange for Administration, Commerce, and Transport (EDIFACT) standard.
- **TRADACOMS** - contains the definition of the entities within your organization that maintain trading relationships with external trading partners, based on the TRADACOMS UN/TDI Standard.
- **X12** - contains the definition of the entities within your organization that maintain trading relationships with external trading partners, based on the ANSI ASC X12 standard.

The EDIFACT, TRADACOMS, and X12 folders can also contain information about contact persons within these internal trading partner organizations. You may add other subfolders to organize internal trading partners into logical groupings by choosing **Folder Utilities** from the Tools menu.

External address book

Use the **External Address Book** to specify address information for partners outside your organization. The **External** folder includes these subfolders:

- **EDIFACT** - contains the definition of entities outside your organization with which your organization maintains relationships, based on the EDIFACT standard. These entities are your trading partners. You may add other subfolders to organize external trading partners into logical groupings.
- **TRADACOMS** - contains the definition of entities outside your organization with which your organization maintains relationships, based on the TRADACOMS UN/TDI Standard. These entities are your trading partners. You may add other subfolders to organize external trading partners into logical groupings.
- **X12** - contains the definition of entities outside your organization with which your organization maintains trading relationships, based on the ANSI ASC X12 standard. These are your trading partners. You may add other subfolders to organize external trading partners into logical groupings.

Internal trading partner and application partner

Select the Internal, **EDIFACT**, **TRADACOMS**, or **X12** folder in the **Address Book** navigator. A dialog box opens.

The **Internal Address Book** is displayed with two list boxes. The **Trading Partner** list contains an entry for each trading partner within your organization. Select a trading partner in the list and the **Application Partner** list shows each application partner of the trading partner.

Trading partners that exchange e-commerce information can be organized for this exchange in several ways. How each trading partner is organized determines the trading partner and application partners that are defined in Partner Manager. The following paragraphs describe certain scenarios.

Configuring internal and external address books

The **Trading Partner** list in the **Address Book** form shows existing trading partners. The **Address Book** form enables you to add, edit, delete, or copy this partner information. You can also access contact information for your trading partners. Add or edit trading partners to define the trading partner EDI information for your company at the corporate level.

The **Application Partner** list in the **Address Book** form shows the application partners for the selected trading partner. The **Address Book** form enables you to add, edit, or delete this partner information. You can also access contact information. Use the **Application Partner** area of the form to add the information for the departments or responsible parties for particular applications. This information is used to create or validate the EDI data envelopes for the trading partner. Define an application partner for each unique address or qualifier within a given trading partner.

Note: An identical dialog box appears when you select the **Internal** or **External** folders for **EDIFACT**, **TRADACOMS**, or **X12**. The following instructions apply to maintaining internal and external trading partner information. Differences between internal and external trading partners are identified.

- [Adding X12 internal trading partners](#)

- [Adding X12 external trading partners](#)
 - [Adding Internal EDIFACT Trading Partners](#)
 - [Adding External EDIFACT Trading Partners](#)
 - [Adding Internal TRADACOMS Trading Partners](#)
 - [Adding External TRADACOMS Trading Partners](#)
 - [Editing a Trading Partner](#)
 - [Copying a Trading Partner](#)
 - [Deleting a Trading Partner](#)
-

Adding X12 internal trading partners

About this task

Use the following instructions to add an X12 Internal Trading Partner:

Procedure

1. Select Address Book from the Partner Manager File menu.
2. Click Internal > X12.
The Address Book for Internal X12 window.
3. Click the **Add** button that is located to the right of the **Trading Partner** list box (upper panel).
The **Add X12 Trading/Application Partner In Internal Folder X12** window opens.

The window contains the tabs listed below. Refer to the following sections for a description of the fields displayed in each tab view:

- ["Trading Partner Tab"](#)
 - ["ISA Options Tab"](#)
 - ["Application Partner Tab"](#)
4. Complete all required fields, then complete any optional fields.
Note: The **Trading Partner Name** and **Interchange ID** field are required.
 5. Click Save.
The window closes and the new Trading Partner is added to the listbox in the Address Book for Internal X12 window.
- [Trading Partner tab](#)
 - [ISA Options tab](#)
 - [Application Partner tab](#)

Trading Partner tab

The following fields are contained in the Add X12 Trading/Application Partner In Internal Folder X12 window, with the Trading Partner tab view displayed:

| | |
|-----------------------------|--|
| Trading Partner Name | Enter a name for this trading partner. This name appears in the Trading Partner list, Trade Links dialog box, Traffic reports, and so on. This name must be unique for each trading partner defined, even if they are in different folders. |
| ISA Address | Select the unique two-character qualifier that identifies the coding structure used to define the Interchange ID. This field corresponds to ANSI ASC X12 data element I05. |
| Interchange ID | Enter the unique identification code for this trading partner to be used to route incoming and outgoing data. If the data is outbound from the trading partner, this is the ISA Sender ID. For data inbound to the trading partner, this is the ISA Receiver ID. This field corresponds to ANSI ASC X12 data elements I06 and I07. |

Note: The combination of the ISA address and Interchange ID must be unique across any implemented system of maps.

ISA Options tab

The following fields are contained in the Add X12 Trading/Application Partner In Internal Folder X12 window, with the **ISA Options** tab view displayed:

| Field | Description |
|---|--|
| Interchange Control Version ID | Select the version of the EDI standard that covers the interchange control segments. A default value of 00305 is provided. This field corresponds to ANSI ASC X12 data element I11.
You can add other versions using the X12 Standards Maintenance utility. |
| Interchange Standards ID or Repetition Character | This field takes on two meanings, depending upon the value selected in the Interchange Control Version ID field. If the value in the Interchange Control Version ID field is less than 00402, this field is called the Interchange Standards ID field. If the value is equal to, or greater than 00402, then it changes to become the Repetition Character field. Both field types are optional: the default value is "U" (caret Hex 5E) if the field denotes the Repetition Character; the default value is "U" for the Interchange Standards ID.
Note The dual meaning of this field resulted when the X12 committee stopped using the Interchange Standards ID and replaced it with the Repetition Character starting with Interchange Control version 00402. |

| Field | Description |
|-------------------|--|
| Separators | <p>Element - Select the delimiter used to separate data elements within a segment. A default value of * (asterisk) is provided.</p> <p>Sub Element - Select the delimiter used to separate component data elements within a composite data structure. A default value of : (colon) is provided. The value of this separator must be different than that of the element separator and segment terminator. This field corresponds to ANSI ASC X12 data element I15.</p> <p>Segment Terminator - Select the character used to terminate a segment. A default value of line feed is provided.</p> <p>You can add other separators to these lists using the X12 Standards Maintenance utility.</p> <p>The Separators delimiters provided as the Element, Sub Element, and Segment Terminator are used when generating outbound e-commerce data, including functional acknowledgements.</p> |

Application Partner tab

The following fields are contained in the Add X12 Trading/Application Partner In Internal Folder X12 window, with the **Application Partner** tab view displayed:

| Field | Description |
|----------------------------------|---|
| Application Partner Name | Enter a name for an application partner, such as a division, department or function within the trading partner. This name appears in the Application Partner List , Trade Links dialog box, Traffic reports, and so on. This name must be unique within a trading partner. |
| GS Address | Enter a code identifying the application partner, used to route data to and from it. If the data is outbound from the application partner, GS Address is the GS Sender ID. For data inbound to this application partner, GS Address is the GS Receiver ID. It corresponds to ANSI ASC X12 data elements 142 and 124. This address must be unique within a trading partner. |
| Alias Name | Enter an alias by which the application partner might be known, for instance an abbreviation of the application partner name, and so on. This alias is not used by Partner Manager or the Message Manager but is available for user application maps. Trading Manager validates this information to ensure that the alias name is unique. See the Trading Manager Installation documentation for more information about these files and the alias.mtt type tree. |
| User Defined 1
User Defined 2 | Enter any user-defined information for this application partner; for example, a department number, an internal customer number, and so on. Partner Manager or the Message Manager does not use this data but it is available for user application maps. See the Trading Manager Installation documentation for more information about these files and the alias.mtt type tree. |

Adding X12 external trading partners

About this task

Use the following instructions to add an X12 Internal Trading Partner:

Procedure

1. In the **Address Book** navigator, click External > X12.
The **Address Book for External X12** opens.
2. Click the **Add** button that is located to the right of the **Trading Partner** list box (upper panel).
The **Add X12 Trading/Application Partner In External Folder X12** window opens.

The window contains the tabs listed below. Refer to the following sections for a description of the fields displayed in each tab view:

- ["Trading Partner Tab - External X12"](#)
 - ["ISA Options Tab"](#)
 - ["TA1 Post Offices Tab"](#)
 - ["Authorization/Security Tab"](#)
 - ["Application Partner Tab"](#)
3. Complete all required fields, then complete any optional fields.
Note: The **Trading Partner Name** and **Interchange ID** field are required.
 4. Click Save.
The window closes and the new Trading Partner is added to the listbox in the Address Book for External X12 window.
- [Trading Partner tab - External X12](#)
 - [ISA Options tab](#)
 - [TA1 Post Offices tab](#)
 - [Authorization/Security tab](#)
 - [Application Partner tab](#)

Trading Partner tab - External X12

The following fields are contained in the Add X12 Trading Partner In External X12 window, **Trading Partner** tab view:

| Field | Description |
|----------------------|---|
| Trading Partner Name | Enter a name for this trading partner. This name appears in the Trading Partner list, Trade Links dialog box, Traffic reports, and so on. This name must be unique for each trading partner defined, even if they are in different folders. |

| Field | Description |
|--|--|
| ISA Address | Select the unique two-character qualifier that identifies the coding structure used to define the Interchange ID. This field corresponds to ANSI ASC X12 data element I05. |
| Interchange ID | Enter the unique identification code for this trading partner to be used to route data to and from it. If the data is outbound from the trading partner, this is the ISA Sender ID. For data inbound to this trading partner, this is the ISA Receiver ID. This field corresponds to ANSI ASC X12 data elements I06 and I07. The combination of the ISA address and Interchange ID must be unique across any system of maps you implement. |
| Interchange | When checked, enables checking of duplicate interchange control numbers for this Trading Partner. If a duplicate interchange is found it is rejected. |
| Functional Group within an Interchange | When checked enables checking of duplicate Functional Group control numbers within the interchange for this partner. If a duplicate functional group is found it is then rejected. |
| Transaction within a Functional Group | When checked, enables checking of duplicate transaction control numbers within the functional group for this partner. If a duplicate transaction is found it is then rejected. |
| Last Interchange Control # | Resets the Interchange control number. |

ISA Options tab

The fields in the Add X12 Trading Partner In External X12 window, **ISA Options** tab view are described in ["ISA Options Tab"](#).

TA1 Post Offices tab

The following fields are contained in the Add X12 Trading Partner In External X12 window, **TA1 Post Offices** tab view::

Get

Allows override of the TA1 Get Post Office that was set in Message Manager Configuration. Get Post Office fields can be disabled depending on the setting in the Message Manager configuration.

Put

Allows override of the TA1 Put Post Office that was set in Message Manager Configuration.

Authorization/Security tab

The fields in the Add X12 Trading Partner In External X12 window, **Authorization/Security** tab view are described in the following table:

| Field | Description |
|--|--|
| Authorization Qualifier (Inbound and Outbound) | Enter the two-character qualifier that identifies the coding structure used to define the inbound or outbound Authorization ID. The default value is 00, indicating no authorization information present. This field corresponds to ANSI/ASC/X12 data element I01. |
| Authorization ID (Inbound and Outbound) | Enter the inbound or outbound authorization information for this trading partner to be used for additional identification or authorization of data sent by this trading partner. By default, no authorization information is defined. This field corresponds to ANSI ASC X12 data element I02. |
| Security Qualifier (Inbound and Outbound) | Select the two-character qualifier that identifies the type of information in the Security ID. The default value is 00, indicating there is no security information present. This field corresponds to ANSI ASC X12 data element I03. |
| Password (Inbound and Outbound) | Enter a password if you selected 01 in the inbound or outbound Security Qualifier field. |
| Validate Authorization | Indicates whether or not ISA Authorization information is validated against the Partner Manager database. |
| Validate Security | Indicates whether or not security elements are validated against the Partner Manager database. |

Application Partner tab

The following fields are contained in the Add X12 Trading Partner In External X12 window, **Application Partner** tab view:

| Field | Description |
|--------------------------|---|
| Application Partner Name | Enter a name for an application partner, such as a division, department or function within the trading partner. This name appears in the Application Partner List, Trade Links dialog box, Traffic reports, and so on. This name must be unique within a trading partner. |
| GS Address | Enter a code identifying the application partner, used to route data to and from it. If the data is outbound from the application partner, GS Address is the GS Sender ID. For data inbound to this application partner, GS Address is the GS Receiver ID. It corresponds to ANSI ASC X12 data elements 142 and 124. This address must be unique within a trading partner. |
| Alias Name | Enter an alias by which the application partner might be known, for instance an abbreviation of the application partner name, and so on. This alias is not used by Partner Manager or the Message Manager but is available for user application maps. Trading Manager validates this information to ensure that the alias name is unique. See the Trading Manager Installation documentation for more information about these files and the alias.mtt type tree. |

| Field | Description |
|--------------------------|---|
| User Defined 1 | Enter any user-defined information for this application partner; for example, a department number, an internal customer number, and so on. Partner Manager or the Message Manager does not use this data but it is available for user application maps. See the Trading Manager Installation documentation for more information about these files and the alias.mtt type tree. |
| User Defined 2 | |
| HIPAA Type 7 Institution | Allows selection of HIPAA Institution. |

Adding Internal EDIFACT Trading Partners

About this task

Use the following instructions to add an internal EDIFACT Trading Partner:

Procedure

- In the **Address Book** navigator, click Internal > EDIFACT.
The **Address Book for Internal EDIFACT** opens.
- Click the **Add** button that is located to the right of the **Trading Partner** list box (upper panel).
The **Add EDIFACT Trading/Application Partner In Internal Folder EDIFACT** window opens.

The window contains the tabs listed below. Refer to the following sections for a description of the fields displayed in each tab view:

- [Identification Tab](#)
 - [Syntax/Separators Tab](#)
 - [Application Partner Tab](#)
- Complete all required fields, then complete any optional fields.
Note: The **Trading Partner Name** and **Interchange ID** field are required.
 - Click Save.
The window closes and the new Trading Partner is added to the listbox in the Address Book for Internal EDIFACT window.
- [Identification tab](#)
 - [Syntax/Separators tab](#)
 - [Application Partner tab](#)

Identification tab

The following fields are contained in the Add EDIFACT Trading Partner In Internal EDIFACT window, in the **Identification** tab view:

| Field | Description |
|---|--|
| Trading Partner Name | Enter a name for this trading partner. This name appears in the Trading Partner list, Trade Links dialog box, Traffic reports, and so on. This name must be unique for each trading partner defined, even if they are in different folders. |
| Interchange ID | Enter the unique identification code for this trading partner to be used to route data to and from it. If the data is outbound from the trading partner, this is the Sender ID. For data inbound to this trading partner, this is the Receiver ID. The combination of the Partner Code ID Qualifier and Interchange ID must be unique across any system of maps you implement. |
| Partner ID Code Qualifier | Select the unique two-character qualifier that identifies the coding structure used to define the Interchange ID. |
| Routing Address | Enter a code identifying the application partner, used to route data to and from it. |
| Comm Agreement ID | Enter a code identifying the commercial agreement ID. |
| Enveloping Option (these buttons only appear when you are adding internal partners) | Identify the format of the data that is provided by the application maps that is sent to this trading partner.
Partially Enveloped - the version type and other information have limited information and are to be completed by Trading Manager before routing the EDI data to the trading partner.
Fully Enveloped - The fully enveloped option is provided to support legacy translator support. It allows the user to use existing translators in order to preserve existing interface code, while using Trading Manager for communications and audit purposes. |

Syntax/Separators tab

The following fields are contained in the Add EDIFACT Trading Partner In Internal EDIFACT window, in the **Syntax/Separators** tab view:

| Field | Description |
|------------------------|--|
| Use UNA Segment | Check box if you use the UNA Segment for this Trading Partner. |
| Syntax ID | Select the Syntax ID used for this Trading Partner. Values are: UNOA; UNOB; UNOC; UNOD; UNOE; UNOF; UNOG; UNOH |
| Syntax Version | Enter a number greater than zero and less than 10 denoting the EDIFACT version syntax in use with this Trading Partner. The default is 1. |
| Component Data Element | Select the delimiter used to separate component data elements within a composite data structure. A default value of : (colon) is provided. |
| Data Element | Select a delimiter used to separate data elements within a segment. |
| Segment Terminator | Select the character used to terminate a segment. A default value of line feed is provided. You can add other separators to these lists using the Standards Maintenance utility. |
| Release Indicator | Select the character used to restore separator and terminator signs within the interchange. A default value of ? (question mark) is provided. |
| Decimal Notation | Select the character used for the decimal point within the interchange. A default value of , (comma) is provided. |

| Field | Description |
|--------------------|--|
| Reserved Character | Spare character reserved for future use. A space is provided as the default. |

Application Partner tab

The following fields are contained in the **Add EDIFACT Trading Partner In Internal EDIFACT** window, in the **Application Partner** tab view:

| Field | Description |
|---------------------------------------|---|
| Application Partner Name | [Optional] Enter a name for an application partner, such as a division, department or function within the trading partner. This name appears in the Application Partner List , Trade Links dialog box, Traffic reports, and so on. This name must be unique within a trading partner. |
| UNG Application ID | Enter a UNG segment header application ID identifying the application partner. |
| Application Partner ID Code Qualifier | [Optional] Identify the format of the data that is provided by the application maps that is sent to this trading partner. |
| Alias Name | [Optional] Enter an alias by which the application partner might be known. For example, an abbreviation of the application partner name, and so on. This alias is not used by Partner Manager or the Message Manager but is available for user application maps. Trading Manager validates this information to ensure that the alias name is unique. |
| User Defined 1 | [Optional] Enter any user-defined information for this application partner; for example, a department number, an internal customer number, and so on. Partner Manager or the Message Manager does not use this data but it is available for user application maps. |
| User Defined 2 | |

Adding External EDIFACT Trading Partners

About this task

Use the following instructions to add an external EDIFACT Trading Partner:

Procedure

- In the **Address Book** navigator, click External EDIFACT. The **Address Book for External EDIFACT** opens.
- Click the **Add** button that is located to the right of the **Trading Partner** list box (upper panel). The **Add EDIFACT Trading/Application Partner In External Folder EDIFACT** window opens.

The window contains the tabs listed below. Refer to the following sections for a description of the fields displayed in each tab view:

- [Identification Tab](#)
 - [Syntax/Separators Tab](#)
 - [Application Partner Tab](#)
- Complete all required fields, then complete any optional fields.
Note: The **Trading Partner Name** and **Interchange ID** field are required.
 - Click Save.
The window closes and the new Trading Partner is added to the listbox in the Address Book for External EDIFACT window.
- [Identification tab](#)
 - [Syntax/Separators tab](#)
 - [Application Partner tab](#)

Identification tab

The following fields are contained in the Add EDIFACT Trading/Application Partner In External Folder EDIFACT window in the **Identification** tab view:

| Field | Description |
|---------------------------|--|
| Trading Partner Name | Enter a name for this trading partner. This name appears in the Trading Partner list, Trade Links dialog box, Traffic reports, and so on. This name must be unique for each trading partner defined, even if they are in different folders. |
| Interchange ID | Enter the unique identification code for this trading partner to be used to route data to and from it. If the data is outbound from the trading partner, this is the Sender ID. For data inbound to this trading partner, this is the Receiver ID. The combination of the Partner Code ID Qualifier and Interchange ID must be unique across any system of maps you implement. |
| Partner ID Code Qualifier | Select the unique two-character qualifier that identifies the coding structure used to define the Interchange ID. |
| Routing Address | Enter a code identifying the application partner, used to route data to and from it. |
| Comm Agreement ID | Enter a code identifying the commercial agreement ID. |
| Password | Allows you to enter a password for the Trading Partner. |
| Password Qualifier | Allows you to select a two character qualifier for the password. |
| Duplicate Control | Allows you to select either None , for no duplicate control, or Interchange in order to have duplicate control conducted at the Interchange level. |

| Field | Description |
|--------------------|--|
| Last UNB Control # | Resets the last UNB Control number to a new value. |

Syntax/Separators tab

The following fields are contained in the **Add X12 Trading/Application Partner In External Folder EDIFACT, Syntax/Separators** tab view:

| Field | Description |
|------------------------|--|
| Use UNA Segment | Check box if you use the UNA Segment for this Trading Partner. |
| Syntax ID | Select the Syntax ID used for this Trading Partner. Values are: UNOA; UNOB; UNOC; UNOD; UNOE; UNOF; UNOG; UNOH |
| Syntax Version | Enter a number greater than zero and less than 10 denoting the EDIFACT version syntax in use with this Trading Partner. The default is 1. |
| Component Data Element | Select the delimiter used to separate component data elements within a composite data structure. A default value of : (colon) is provided. |
| Data Element | Select a delimiter used to separate data elements within a segment. |
| Segment Terminator | Select the character used to terminate a segment. A default value of line feed is provided. You can add other separators to these lists using the Standards Maintenance utility. |
| Release Indicator | Select the character used to restore separator and terminator signs within the interchange. A default value of ? (question mark) is provided. |
| Decimal Notation | Select the character used for the decimal point within the interchange. A default value of , (comma) is provided. |
| Reserved Character | Spare character reserved for future use. A space is provided as the default. |

Application Partner tab

The following fields are contained in the **Add X12 Trading/Application Partner In External Folder EDIFACT, Application Partner** tab view:

| Field | Description |
|---------------------------------------|---|
| Application Partner Name | [Optional] Enter a name for an application partner, such as a division, department or function within the trading partner. This name appears in the, Application Partner List Trade Links dialog box, Traffic reports, and so on. This name must be unique within a trading partner. |
| UNG Application ID | Enter a UNG segment header application ID identifying the application partner. |
| Application Partner ID Code Qualifier | [Optional] Identify the format of the data that is provided by the application maps that is sent to this trading partner. |
| Alias Name | [Optional] Enter an alias by which the application partner might be known. For example, an abbreviation of the application partner name, and so on. This alias is not used by Partner Manager or the Message Manager but is available for user application maps. Trading Manager validates this information to ensure that the alias name is unique. |
| User Defined 1
User Defined 2 | [Optional] Enter any user-defined information for this application partner; for example, a department number, an internal customer number, and so on. Partner Manager or the Message Manager does not use this data but it is available for user application maps. |
| Password | Allows you to enter a password for the Application Partner. |

Adding Internal TRADACOMS Trading Partners

About this task

Use the following instructions to add an internal TRADACOMS Trading Partner:

Procedure

- In the **Address Book** navigator, click Internal > **TRADACOMS**.
The **Address Book for Internal TRADACOMS** opens.
- Click the **Add** button that is located to the right of the **Trading Partner** list box (upper panel).
The **Add TRADACOMS Trading Partner In Internal Folder TRADACOMS** window opens.

The window contains the tabs listed below. Refer to the following sections for a description of the fields displayed in each tab view:

- ["Trading Partner Tab"](#)
 - ["Application Partner Tab"](#)
- Complete all required fields, then complete any optional fields.
Note: The **Trading Partner Name** and **Trading Partner ID** fields are required.
 - Click Save.
The window closes and the new Trading Partner is added to the listbox in the Address Book for Internal TRADACOMS window.
- [Trading Partner tab](#)
 - [Application Partner tab](#)

Trading Partner tab

The following fields are contained in the **Add X12 Trading Partner In Internal Folder TRADACOMS** window in the **Trading Partner** tab view:

| Field | Description |
|----------------------|---|
| Trading Partner Name | Enter a name for this trading partner. This name appears in the Trading Partner list, Trade Links dialog box, Traffic reports, and so on. This name must be unique for each trading partner defined, even if they are in different folders. |
| Trading Partner ID | Enter the unique identification code for this trading partner to be used to route data to and from it. If the data is outbound from the trading partner, this is the Sender Reference or Name. For data inbound to this trading partner, this is the Receiver Reference or Sender Name. |

Application Partner tab

The following fields are contained in the **Add X12 Trading Partner In Internal Folder TRADACOMS** window in the **Application Partner** tab:

| Field | Description |
|----------------------------------|---|
| Application Partner Name | Enter a name for an application partner, such as a division, department, or function within the trading partner. This name appears in the Application Partner List, Trade Links dialog box, Traffic reports, and so on. This name must be unique for each trading partner defined, even if they are in different folders. |
| Application Partner ID | Determined form the SDT (supplier ID composite) and CDT (customer ID composite). |
| Alias Name | Enter an alias by which the application partner might be known. For example, an abbreviation of the application partner name, and so on. This alias is not used by Partner Manager or the Message Manager but is available for user application maps. Trading Manager validates this information to ensure that the alias name is unique. |
| User Defined 1
User Defined 2 | Enter any user-defined information for this application partner; for example, a department number, an internal customer number, and so on. Partner Manager or the Message Manager does not use this data but it is available for user application maps. |

Adding External TRADACOMS Trading Partners

About this task

Use the following instructions to add an external TRADACOMS Trading Partner:

Procedure

1. In the **Address Book** navigator, click External > **TRADACOMS**.
The **Address Book for External TRADACOMS** opens.
2. Click the **Add** button that is located to the right of the **Trading Partner** list box (upper panel).
The **Add TRADACOMS Trading Partner In External Folder TRADACOMS** window opens.

The window contains the tabs listed below. Refer to the following sections for a description of the fields displayed in each tab view:

- ["Trading Partner Tab"](#)
 - ["Application Partner Tab"](#)
3. Complete all required fields, then complete any optional fields.
Note: The **Trading Partner Name** and **Trading Partner ID** field are required.
 4. Click Save.
The window closes and the new Trading Partner is added to the listbox in the Address Book for External TRADACOMS window.

- ["Trading Partner tab"](#)

Trading Partner tab

The following fields are contained in the **Add TRADACOMS Trading Partner In External Folder TRADACOMS** window in the **Trading Partner** tab view:

| Field | Description |
|----------------------|---|
| Trading Partner Name | Enter a name for this trading partner. This name appears in the Trading Partner list, Trade Links dialog box, Traffic reports, and so on. This name must be unique for each trading partner defined, even if they are in different folders |
| Trading Partner ID | Enter the unique identification code for this trading partner to be used to route data to and from it. If the data is outbound from the trading partner, this is the Sender Reference or Name. For data inbound to this trading partner, this is the Receiver Reference or Sender Name. |
| Duplicate Control | Allow duplicate control checking to be set to None , or to be set to Interchange when duplicate control checking is to be performed at the interchange level. |

Editing a Trading Partner

About this task

The Edit Trading Partner window enables you to edit the EDIFACT, TRADACOMS, and X12 fields.

To edit a trading partner

Procedure

1. In the **Address Book** navigator, click Internal > *folder* or External > *folder* to display the **Address Book** form, where *folder* is **EDIFACT**, **TRADACOMS**, or **X12**.
2. Select a trading partner from the **Trading Partner** list.
3. Click **Edit**.
Note: You can also double-click the trading partner name to display the Edit Trading Partner dialog box.
The Edit Trading Partner dialog box appears.
4. Edit any field for a trading partner.
5. Click **Save**.

Copying a Trading Partner

You can copy existing trading partner information to a new trading partner.

Enable the **Copy Application Partners** check box to copy associated application partners information to the new trading partner.

The following information is not copied:

- Application partner **Alias Name**; this record is unique to each application partner
- Contact information associated with the trading and application partners
- Trade links associated with the trading and application partners

To copy a trading partner

1. In the **Address Book** navigator, click **Internal** > *folder* or **External** > *folder* to display the **Address Book** form, where *folder* is **EDIFACT**, **TRADACOMS** or **X12**.
2. Select a trading partner from the **Trading Partner** list.
3. Click **copy**.
The **Copy Trading Partner** window opens.
4. Enter a new trading partner name in the **Trading Partner Name** field. The other required fields vary depending upon whether you are copying X12, EDIFACT or TRADACOMS.
 - For X12 - You must select the new ISA Address
 - For EDIFACT - You must change either the Interchange ID or the Routing Address
 - For TRADACOMS - You must change the Trading Partner ID
5. Change any information in the remaining fields, if required.
6. Enable the **Copy Application Partners** check box if you want the existing application partner information copied to the new trading partner.
7. Click **Save**.
The information is copied to a new trading partner and appears in the Address Book.

Deleting a Trading Partner

About this task

The following procedures describe how to delete a trading partner.

To delete a trading partner

Procedure

1. In the **Address Book** navigator, click Internal > *folder* or External > *folder* to display the **Address Book** form, where *folder* is **EDIFACT**, **TRADACOMS**, or **X12**.
 2. Select a trading partner from the **Trading Partner** list.
 3. Click **Delete**.
The **Trading Partner Delete** form displays.
- Note: This function may delete many records from the database and should be used with care. The **Trading Partner Delete** form displays the impact of this deletion, and after accepted, deletes all records associated with the Trading Partner.
4. Click **Delete** to confirm the removal of the records.
A confirmation appears that asks if it is okay to remove the Trading Partner.
 5. Click **OK**.
A message box opens that tells you the delete was performed successfully.
 6. Click **OK** to close the message box.
The Trading Partner has been successfully deleted and is removed from the Address Book.

Application partners

Application partners are associated with a particular trading partner or application, such as the Billing department. Application partners provide a means of grouping transactions by functional area within an organization.

You can add, edit, and delete application partners.

Application partners are managed in the Internal and External address book for the data type (EDIFACT, TRADACOMS, or X12).

For all application partner management, select the data type (EDIFACT, TRADACOMS, or X12) in the Internal or External address book.

Note: Application partners are required for X12 but are optional for EDIFACT and TRADACOMS.

- [Adding an X12 Application Partner](#)
- [Adding an EDIFACT Application Partner](#)
- [Adding TRADACOMS Application Partners](#)
- [Editing an Application Partner](#)
- [Deleting an Application Partner](#)

Adding an X12 Application Partner

About this task

Use the following instructions to add an internal or external X12 Application Partner

Procedure

1. In the **Address Book** navigator:
 - Click Internal...X12 to add an Application Partner to an Internal Trading Partner.
 - Click External...X12 to add an Application Partner to an Internal Trading Partner.
2. Select a Trading Partner in the **Trading Partner** list box (upper panel).
3. Click the **Add** button that is located to the right of the **Application Partner** list box (lower panel).
The **Add Internal Application Partner for Trading Partner** window opens.
4. Enter information in the fields of this window. See the next section, "[Add X12 Application Partner Window Fields](#)" for field descriptions.
Note: The **GS Address** must be unique within the associated trading partner.
Note: The **Application Partner Name** and **GS Address** fields are required.
5. Click Save.
The new application partner is added and appears in the Application Partner list in the Address Book.

- [Add X12 Application Partner Window fields](#)

Add X12 Application Partner Window fields

The following fields are contained in the X12 Add Internal Application Partner for Trading Partner window:

| Header | Header |
|----------------------------------|---|
| Application Partner Name | Enter a name for an application partner, such as a division, department or function within the trading partner. This name appears in the, Application Partner List , Trade Links dialog box, Traffic reports, and so on. This name must be unique within a trading partner. |
| GS Address | Enter a code identifying the application partner, used to route data to and from it. If the data is outbound from the application partner, GS Address is the GS Sender ID. For data inbound to this application partner, GS Address is the GS Receiver ID. It corresponds to ANSI ASC X12 data elements 142 and 124. This address must be unique within a trading partner. |
| Alias Name | Enter an alias by which the application partner might be known, for instance an abbreviation of the application partner name, and so on. This alias is not used by Partner Manager or the Message Manager but is available for user application maps. Trading Manager validates this information to ensure that the alias name is unique. See the Trading Manager Installation documentation for more information about these files and the alias.mtt type tree. |
| User Defined 1
User Defined 2 | Enter any user-defined information for this application partner; for example, a department number, an internal customer number, and so on. Partner Manager or the Message Manager does not use this data but it is available for user application maps. See the Trading Manager Installation documentation for more information about these files and the alias.mtt type tree. |
| HIPAA Type 7
Institution | Note: This field only appears when you are adding an Application Partner to an X12 External Trading Partner. Allows you to select a type of Healthcare Institution to associate with the Application Partner (for example Medicare). |

Adding an EDIFACT Application Partner

About this task

Use the following instructions to add an internal or external EDIFACT Application Partner

Procedure

1. In the **Address Book** navigator:
 - Click Internal...EDIFACT to add an Application Partner to an Internal Trading Partner.
 - Click External...EDIFACT to add an Application Partner to an Internal Trading Partner.

- Select a Trading Partner in the **Trading Partner** list box (upper panel).
 - Click the **Add** button that is located to the right of the **Application Partner** list box (lower panel).
The **Add Internal Application Partner for Trading Partner** window opens.
 - Enter information in the fields of this window. See the next section, "[Add EDIFACT Application Partner Window Fields](#)" for field descriptions.
Note: The **Application Partner Name** and **UNG Application ID** fields are required.
 - Click Save.
The new application partner is added and appears in the Application Partner list in the Address Book.
- [Add EDIFACT Application Partner window fields](#)

Add EDIFACT Application Partner window fields

The following fields are contained in the Add Internal Application for Trading Partner EDIFACT window in the **Application Partner** tab view:

| Field | Description |
|---------------------------------------|---|
| Application Partner Name | Enter a name for an application partner, such as a division, department or function within the trading partner. This name appears in the , Application Partner List Trade, Links dialog box, Traffic reports, and so on. This name must be unique within a trading partner. |
| UNG Application ID | Enter a UNG segment header application ID identifying the application partner. |
| Application Partner ID Code Qualifier | Identify the format of the data that is provided by the application maps that is sent to this trading partner. |
| Alias Name | Enter an alias by which the application partner might be known. For example, an abbreviation of the application partner name, and so on. This alias is not used by Partner Manager or the Message Manager but is available for user application maps. Trading Manager validates this information to ensure that the alias name is unique. |
| User Defined 1
User Defined 2 | Enter any user-defined information for this application partner; for example, a department number, an internal customer number, and so on. Partner Manager or the Message Manager does not use this data but it is available for user application maps. |
| Password (external EDIFACT only) | This field appears only when you are adding an Application Partner to an external EDIFACT Trading Partner. Allows you to enter a password for the Application Partner. |

Adding TRADACOMS Application Partners

About this task

Use the following instructions to add an internal or external EDIFACT Application Partner

Procedure

- In the **Address Book** navigator:
 - Click **Internal...TRADACOMS** to add an Application Partner to an Internal Trading Partner.
 - Click **External...TRADACOMS** to add an Application Partner to an External Trading Partner.
 - Select a Trading Partner in the **Trading Partner** list box (upper panel).
 - Click the **Add** button that is located to the right of the **Application Partner** list box (lower panel).
The Add Internal Application Partner for Trading Partner window opens.
 - Enter information in the fields of this window. See the next section, "[Add TRADACOMS Application Partner Window Fields](#)" for field descriptions.
 - Click Save.
The new application partner is added and appears in the Application Partner list in the Address Book.
- [Add TRADACOMS Application Partner window fields](#)

Add TRADACOMS Application Partner window fields

The following fields are contained in the Add Internal Application for Trading Partner TRADACOMS window in the **Application Partner** tab view:

| Field | Description |
|----------------------------------|---|
| Application Partner Name | Enter a name for an application partner, such as a division, department, or function within the trading partner. This name appears in the Application Partner List Trade Links dialog box, Traffic reports, and so on. This name must be unique for each trading partner defined, even if they are in different folders. |
| Application Partner ID | Determined from the SDT (supplier ID composite) and CDT (customer ID composite). |
| Alias Name | Enter an alias by which the application partner might be known. For example, an abbreviation of the application partner name, and so on. This alias is not used by Partner Manager or the Message Manager but is available for user application maps. Trading Manager validates this information to ensure that the alias name is unique. |
| User Defined 1
User Defined 2 | Enter any user-defined information for this application partner; for example, a department number, an internal customer number, and so on. Partner Manager or the Message Manager does not use this data but it is available for user application maps. |

Editing an Application Partner

About this task

You can edit application partner definitions.

To edit an application partner

Procedure

1. In the **Address Book** navigator, click Internal > *folder* or External > *folder* to display the **Address Book** form, where *folder* is **EDIFACT**, **TRADACOMS**, or **X12**.
2. Select a trading partner in the **Trading Partner** list to view the associated application partners.
3. Select an application partner in the **Application Partner** list.
4. Next to the **Application Partner** list, click the **Edit** button.
The Edit Application Partner dialog box appears.
5. Edit any field for the application partner.
6. Click Save.

Deleting an Application Partner

About this task

Note: Associated trade links must be deleted from the trading partner **Address Book** before the application partner can be deleted.

To delete an application partner

Procedure

1. In the **Address Book** navigator, click Internal > *folder* or External > *folder* to display the **Address Book** form, where *folder* is **EDIFACT**, **TRADACOMS**, or **X12**.
2. Select a trading partner from the **Trading Partner** list.
3. Select the application partner from the **Application Partner** list.
4. Click Delete and confirm the delete action when prompted.

Managing Administrative Contacts

About this task

Use the **Address Book** to define contacts that are associated with a particular trading partner or particular application partner. You can also define administrative contacts that are not associated with a trading partner. Contact information is useful for defining the person to contact in your organization or your trading partner's organization about issues involving EDI traffic.

Contact information includes identifying individuals or groups of individuals and how to reach them. Providing contact information allows Trading Manager to send error notifications.

These contact types are not associated with a trading or application partner; therefore, the **Alert Options** are not available. Typically, they are people in your organization to contact.

You can add, edit, and delete administrative contacts in the **Admin** address book.

- [Adding a Contact](#)
- [Editing a Contact](#)
- [Deleting a Contact](#)

To add an administrative contact

Procedure

1. In the **Address Book** navigator, select the Internal > Admin folder.
The Contacts for Administration dialog box appears, listing any defined administrative contacts.
2. Click Add.
The Add a Contact dialog box appears.
3. Enter the contact information.
4. Click Save.

To edit an administrative contact

Procedure

1. In the **Address Book** navigator, click the Internal > Admin folder.
2. Select the contact from the **Contacts** list.
3. Click Edit in the Contact List dialog box. The **Edit a Contact** dialog box appears.
4. Edit the contact information.
5. When your contact information is complete, click Save.

To delete an administrative contact

Procedure

1. In the **Address Book** navigator, click the **Internal** Admin folder.
2. Select the contact from the list.
3. Click Delete.
A message box opens that prompts you to confirm the delete.
4. Click Yes.
The contact is deleted.

Adding a Contact

About this task

Use the following procedures to define contacts that are associated with a particular trading partner or particular application partner. You can also define administrative contacts that are not associated with a trading partner.

This information is defined for your convenience and may be used to format e-mail message headers, compose letters using your default word processor, or route data to appropriate people when Alert conditions are detected.

To add a contact

Procedure

1. In the **Address Book** navigator, click Internal > *folder* or External > *folder* to display the **Address Book** form, where *folder* is **EDIFACT**, **TRADACOMS**, or **X12**.
2. Select a partner from the **Trading Partner** list or **Application Partner** list.
3. Click Contacts in the area for the partner selected.
The **Contact List** appears, listing any contacts defined for the selected partner.
4. Click Add in the Contact List dialog box.
The **Add a Contact** dialog box appears.
5. Enter the contact information.
6. Select an alert option from the **Alert Options** group box. (This option is only available for contacts associated with internal trading/application partners).
7. Click OK.
8. Click Save.
 - [E-Mail Check box and Field](#)
 - [Monitor Directory Check box and Field](#)

E-Mail Check box and Field

About this task

If the alert is to be sent by e-mail, enable the **E-Mail** check box and specify the e-mail address to which the alert is to be sent. If you specified an e-mail address on the **Contact** dialog box, that information is automatically displayed in this field.

Monitor Directory Check box and Field

About this task

If the alert is to be sent to a directory location, enable **Monitor Directory**, and specify the directory to which the alert is to be written.

Editing a Contact

About this task

Note: Alert Options are only available when editing information for a contact associated with an internal trading or application partner.
To edit a contact

Procedure

1. In the **Address Book** navigator, click Internal > *folder* or External > *folder* to display the **Address Book** form, where *folder* is **EDIFACT**, **TRADACOMS**, or **X12**.
 2. Select a partner from the **Trading Partner** list or **Application Partner** list.
 3. Click Contacts in the area for the partner selected.
The trading or application partner **Contact** form appears, listing the contacts defined for the selected partner.
 4. Select a contact in the contact list.
 5. Click Edit in the Contacts dialog box.
The Edit Contact dialog box appears.
 6. Edit the contact information including **Alert Options**.
 7. Click Save.
-

Deleting a Contact

About this task

Use the following procedure to delete a contact for a trading or application partner.

To delete a contact

Procedure

1. In the **Address Book** navigator, click Internal > *folder* or External > *folder* to display the **Address Book** form, where *folder* is **EDIFACT**, **TRADACOMS**, or **X12**.
2. Select a partner from the **Trading Partner** list or **Application Partner** list.
3. Click Contacts in the area for the partner selected.
The trading or application partner **Contacts** form appears, listing the contacts defined for the selected partner.
4. Select the contact from the **Contacts** list.

Results

Click Delete and confirm the delete action when prompted

Post Offices

Use the **Post Offices** navigator to define and maintain the routing instructions for inbound and outbound data for Trading Manager. The **Post Offices** navigator is used to define post offices that represent sources and destinations for data flowing to and from Trading Manager.

- [Overview](#)
 - [Viewing Post Offices](#)
 - [Adding Post Offices](#)
 - [Copying Post Offices](#)
 - [Editing Post Offices](#)
 - [Deleting Post Offices](#)
 - [VAN Post Offices Overview](#)
-

Overview

This chapter describes the **Post Offices** navigator and provides instructions for creating and modifying post office information.

Procedures described in this chapter enable you to define:

- Where data from trading partner relationships is transferred
- How data is transferred to and from Trading Manager

A **Get** post office is the source for data that is processed by the Message Manager. A **Put** post office is the destination for data from the Message Manager.

- A **Get** post office is always used to pick up data coming into the Message Manager
 - A **Put** post office is always used to drop off data coming out of the Message Manager
- [File Post Offices](#)
 - [E-Mail post offices](#)
 - [FTP Post Offices](#)
 - [VAN post offices](#)
 - [HTTP post offices](#)
 - [JMS post offices](#)
 - [MessageQ client post offices](#)
 - [MessageQ server post offices](#)
 - [MQSeries client post offices](#)
 - [MQSeries server post offices](#)

- [MSMQ post offices](#)
 - [OracleAQ post offices](#)
-

File Post Offices

Use a File Post Office when data is transferred to or from local directory structures. Configuring a file route post office requires only that you identify a local directory where e-commerce data is **Get** or **Put** without specifying how the file is physically transferred.

- **Get Post Offices** - The Message Manager is configured to execute automatically based on new files created or existing files changed in the directory defined for the File Post Office. This creation or change is known as a file event. Unless Message Manager has been customized, there is only one Get File Directory: the one specified in **mmgr_mail** value. For example, in Windows this is: `install_dir\tmgr_vn.n\mmgr\mail`. Any other get file directory is ignored unless the Launcher and **stampandsortfile** systems have been customized. See the Trading Manager documentation for further information about **stampandsortfile** map systems.
Note: Get post office fields may be disabled depending upon the setting in the Message Manager configuration.
 - **Put Post Offices** - The data resulting from Message Manager processing is written to files in the directory defined for the File Post Office.
-

E-Mail post offices

Use an E-mail Post Office when the source of or destination for e-commerce data is an e-mail message using a MAPI, cc:Mail, Lotus Notes, or Internet mail system. When you define an e-mail post office, Trading Manager uses the e-mail adapter to route mail into or out of the Message Manager. For more information about the E-mail adapter, see the E-mail Adapter documentation.

- **Put Post Offices** - The data resulting from Message Manager processing is routed using the e-mail adapters and the settings defined for the e-mail post office.
-

FTP Post Offices

Use File Transfer Protocol (FTP) Post Offices to get and put files in remote directories using the Internet's FTP communications support. Trading Manager uses the FTP adapter to route files using FTP. For more information about the FTP adapter, see the FTP Adapter documentation.

- **Get Post Offices** - The Message Manager is configured to execute at particular time intervals to retrieve any available data from the remote directories using the FTP adapters and the settings defined for the FTP post office.
Note: Get Post Office fields may be disabled depending upon the setting in Message Manager configuration.
 - **Put Post Offices** - The data resulting from Message Manager processing is routed to the remote location using the FTP adapters and the settings defined for the FTP post office.
 - [Examples of FTP commands](#)
The following are examples of typical FTP commands. In these examples, replace `company` with your company name.
-

Examples of FTP commands

The following are examples of typical FTP commands. In these examples, replace `company` with your company name.

```
-t -URL FTP://abfed1:adiabf@ftp2.company.com/outgoing/$thread$.xxx  
-t -URL FTP://abfed1:adiabf@ftp2.company.com/outgoing/$thread$.Internal_test  
-t -URL FTP://abfed1:adiabf@ftp2.company.com/outgoing/$thread$_internal_test.xxx  
-t -URL FTP://abfed1:adiabf@ftp2.company.com/outgoing/Test_$thread$_somevalue.xxx  
-t -URL FTP://abfed1:adiabf@ftp2.company.com/outgoing/abc.xxx
```

In the first example in the above list, xxx indicates the specific Post Office nickname. In the last example in the list, any file for this Post Office will have the unique name abc.txt and will override any previous one.

VAN post offices

Use VAN Post Offices to send and receive mail asynchronously through a commercial Value Added Network (VAN). VAN adapters must be used to send or receive mail from a Value Added Network by using a scripted session. For more information about the VAN adapter, see the VAN Adapter documentation.

- **Put Post Offices** - The data resulting from Message Manager processing is routed to the remote VAN using the VAN adapters and the settings defined for the VAN post office.
-

HTTP post offices

Use HTTP Post Offices when e-commerce data is being transferred to and from an HTTP(S) web server. If you have a Command Server, a Launcher, or Platform API on one platform, you can use the HTTP adapter to transfer data to an HTTP(S) web server on another platform. For more information about the HTTP adapter, see the HTTP Adapter documentation.

JMS post offices

Use JMS Post Offices when e-commerce data is being transferred to a Java Messaging Service (JMS) compliant system provider. JMS is part of the Java 2 Platform Enterprise Edition (J2EE) standards and defines a standard API for accessing enterprise-messaging systems. Trading Manager uses the Java Message Service Adapter to transfer data to and from JMS systems. For more information about the JMS adapter, see the Java Message Service Adapter documentation.

MessageQ client post offices

Use MessageQ Client Post Offices when BEA MessageQ adapters are used to perform message transformation in the MessageQ environment, including the translation of data between different formats. The BEA MessageQ adapters also control the flow of messages between MessageQ applications. For more information about the BEA MessageQ adapter, see the BEA MessageQ Adapter documentation.

MessageQ server post offices

Use Message Q Server Post Offices when BEA MessageQ adapters are used to perform message transformation in the MessageQ environment, including the translation of data between different formats. The BEA MessageQ adapters also control the flow of messages between MessageQ applications. For more information about the BEA MessageQ adapter, see the BEA MessageQ Adapter documentation.

MQSeries client post offices

Use MQSeries Client Post Offices when transferring e-commerce data between applications using IBM WebSphere applications. IBM WebSphere MQ can move data in heterogeneous computer networks, allowing systems of dissimilar computers to move data to and from each other. The IBM WebSphere MQ client adapter controls the data. For more information about the IBM WebSphere MQ Adapter, see the IBM WebSphere MQ Adapter documentation.

MQSeries server post offices

Use MQ Series Server Post Offices when transferring e-commerce data between applications using IBM WebSphere applications. IBM WebSphere MQ can move data in heterogeneous computer networks, allowing systems of dissimilar computers to move data to and from each other. The IBM WebSphere MQ client adapter controls the data. For more information about the IBM WebSphere MQ Adapter, see the IBM WebSphere MQ Adapter documentation.

MSMQ post offices

Use MSMQ Post Offices when transferring e-commerce data to and from Microsoft Message Queue (MSMQ) servers. For more information about the MSMQ Adapter see the MSMQ Adapter documentation.

OracleAQ post offices

Use OracleAQ Post Offices when e-commerce data is being transferred to and from an Oracle database. See the OracleAQ Adapter documentation.

Viewing Post Offices

About this task

When you select a post office view from the **Post Offices** navigator, you can add, copy, edit, and delete post offices of that type.

To view all post offices

Procedure

1. Open Partner Manager.
2. From the File menu, select Post Offices.
The **Post Offices** navigator opens.
3. Click All Post Offices in the navigator.
The All Post Offices dialog box appears with both **Get** and **Put** post offices displayed.
4. View **Get** or **Put** post offices only by clicking the associated option button in **Post Office Type**.

To view specific post offices

Procedure

Select one of the post office types displayed in the navigator.

For example, when you select E-Mail from the navigator, a list showing only the e-mail post offices opens.

Adding Post Offices

About this task

The following steps describe how to add a post office.

Note: The following instructions are not applicable to File and VAN post offices. If you are adding a File post office, see "[Adding a File Post Office](#)". For VAN post offices see the [Adding a VAN Post Office](#).

To add post offices

Procedure

1. Select one of the post office types displayed in the navigator.

The list of post offices for that adapter is displayed.

2. Click Add.

The **Add Post Office** dialog box appears.

3. Enter the following information:

- a. **Name** - Name for the post office; for example, X12 Data for Processing. The name displays in the Post Offices list and in the drop-down list boxes when maintaining trade links.
- b. **Nickname** - A three-character nickname. A valid, unique nickname is automatically generated based on the **Name** entry. You may use this nickname or assign one of your own. The nickname is used to identify this post office throughout the Trading Manager system.
- c. **Adapter Type** - Select an adapter type.
- d. **Get/Put Indicator** - Select the post office type option.
- e. **Adapter Command String** - Enter an adapter command string; for example:
-PR MAPI -U mzipper -P srhwec9 -TO pbure -RCPT

4. Click Save.

The new post office appears in the **Post Offices** list.

- [Adding a File Post Office](#)
-

Adding a File Post Office

About this task

To add a file post office

Procedure

1. Click File in the **Post Offices** navigator.

The File Post Offices dialog opens.

2. Click Add.

The Add Post Office dialog box opens.

3. Select File from the **Adapter Type** drop-down menu.

4. The Add Post Office dialog box for **File** adapters opens.

Enter information in this dialog box as described in the following table.

| Field | Information to enter or select |
|-------------------|---|
| Name | Name of the file route post office; for example, X12 Data For Processing. The name displays in the Post Offices list and in the Get File Post Office and Put File Post Office lists in the Trade Links navigator. |
| Nickname | A three-character nickname. A valid, unique nickname is automatically generated based on the Name entry. You may use this nickname or assign one of your own. The nickname is used to identify this post office throughout the Trading Manager system. |
| Get/Put Indicator | Select the post office type option. |
| Directory | Enter the path of the directory or folder where files for the file post office are located. Specify a path relative to the directory where the Launcher is installed. It is recommended that you use the default directory (<i>install_dir\tnmgr_n.n\mmgr\file</i>) for all Get file post offices to simplify Message Manager configuration and processing, instead of creating different physical directories for each file route post office. For Put file post offices, however, you may specify different physical directories. |

5. Click Save.

The new post office appears in the **Post Offices** list.

Copying Post Offices

About this task

You can copy post offices.

To copy a post office

Procedure

1. From the File menu, select Post Offices.
The list of all post offices appears.
2. Select a post office from the **Post Offices** list.
3. Click Copy. The Copy Post Office dialog box appears.
4. Specify a new **Name** and **Nickname** for the copy of the existing post office.
5. Edit the other File post office settings as needed.
6. Click Save.
The additional post office appears in the **Post Offices** list.

Editing Post Offices

About this task

You can edit post offices.

To edit a post office

Procedure

1. Select one of the Post Office types displayed in the navigator.
The list of Post Offices for that adapter is displayed.
 2. Select a post office from the list.
 3. Click Edit.
The Edit Post Office dialog box opens.
- Note: When editing the File Post Office, be aware that changing the nickname may cause a conflict with associated trade links. Also, if you change the physical directory, you need to verify that all files have been moved to the new directory.
4. Specify a new **Name** and **Nickname** for the copy of the existing post office.
 5. Edit the other File post office settings as needed.
 6. Click Save.
The additional post office appears in the **Post Offices** list.

Deleting Post Offices

About this task

Note: You cannot delete a post office that is associated with any trade links. You must change the post office for each affected trade link prior to deleting the post office.

To delete a post office

Procedure

1. Select one of the Post Office types displayed in the navigator.
The list of Post Offices for that adapter is displayed.
2. Select the post office that you want to delete from the list.
3. Click Delete and confirm the delete action when prompted.
The post office is removed from the **Post Offices** list.

VAN Post Offices Overview

This section describes how to configure the VAN adapter as well as how to specify options for the Monitor file, as well as how to add, copy, edit, and delete VAN Post Offices.

- [Supported VAN Types](#)
- [Adding a VAN Post Office](#)

Supported VAN Types

The following is a list of supported VAN types:

| | |
|---------------------------------|-------------------------|
| • Advantis (IBM) | • Kleinschmidt |
| • Advantis (Sears) | • EDINet |
| • AT&T | • Sprint |
| • Commerce | • Transettlements |
| • Compuserve | • User-defined networks |
| • GE Information Systems (GEIS) | |

Adding a VAN Post Office

About this task

For more information about the VAN adapter, see the VAN Adapter documentation.

To add a VAN post office

Procedure

1. Select VAN from the list of Post Office types displayed in the navigator.
The VAN Post Offices dialog box opens with all Post Offices associated with the VAN adapter displayed.
2. Click Add.
The **Add Post Office** form appears.
3. Select VAN from the **Adapter Type** drop-down menu.
The Add Post Office dialog box for the **VAN** adapter type opens.
4. Enter information in this dialog box as described in the following table.

| VAN Post Office Field | Information to Enter or Select |
|------------------------|--|
| Name | Name for the post office; for example, X12 Data For Processing . The name displays in the Post Office list and in the Get Mail Post Office and Put Mail Post Office drop-down list box in the Trade Links navigator. |
| Nickname | A three-character nickname. Partner Manager automatically generates a valid, unique nickname based on the Name entry. You may use this nickname or assign one of your own. The nickname is used to identify this post office throughout the Trading Manager system. |
| Adapter Type | Select the VAN adapter type. |
| Get/Put Indicator | Enable the option button for Get Post Office or Put Post Office . |
| Adapter Command String | The command string specified in the VAN configuration setting automatically displays here. |

5. At this point you can either save the Post Office without configuring it, or configure the Post Office before you save it.
 - To save the Post Office without configuring it, click the **Save** button. A confirmation box will open which will ask you to confirm your save. Click OK to close this box and add the Post Office to the scroll list in the VAN Post Offices dialog box.
 - If you want to configure the Post Office, see the next section *Configuring a VAN Post Office*.
- [Configuring a VAN Post Office](#)
- [VAN source configuration](#)
- [Specifying options for the Monitor File](#)
- [Other VAN source configuration options](#)

Configuring a VAN Post Office

About this task

These steps describe how to configure a VAN Post Office.

To configure a VAN Post Office

Procedure

1. Add a VAN Post Office and complete all fields as described in "[Adding a VAN Post Office](#)".
2. Click Configure.
The **VAN Source Configuration** dialog box appears.

Note: All VANs include the **VAN Name**, **Phone**, **User ID**, **Password**, and **Script** fields; some VANs have additional fields to configure.

3. Select the **VAN Name**.

Each supported VAN destination is configured according to VAN requirements. After you select the **VAN Name** from the drop-down list, the configuration dialog box and required fields are customized for each VAN.

4. Enter a numeric **Phone** number for the modem to dial to connect to the VAN.

The numeric phone number cannot contain any spaces, dashes, or parentheses.

5. Enter the **User ID** for the specific VAN account.

6. Enter the **Password** associated with the **User ID**.

7. Enter the **Script** file name or click **Browse** to navigate your file system and select the **Script** file.

The script file contains all the commands and responses to send to the VAN to satisfy the logon requirements of the specific network. You may want to add your own script, create it, and save the file with a **.job** file name extension.

Scripts used as sources for input map cards have a ***_g.job** designation (indicating a **GET**). Scripts used as targets (destinations) for output map cards have a ***_p.job** designation (indicating a **PUT**).

8. Click **OK**.

The VAN adapter command string is built from the information in the VAN Source Configuration dialog box and appears in the **Adapter Command String** field.

9. Click **Save**.

The new VAN Post Office appears in the **Post Offices** list.

VAN source configuration

In addition to defining VAN source information, you can define configuration files, trace file creation, and monitor options.

Click a tab on the VAN Source Configuration dialog box to define:

- **Options** to specify a configuration file other than the default **aplus.cfg**. Leave this field blank.
- **Trace** to specify a trace file name and append option for a trace file containing diagnostic information.
- **Diagnostics** to specify a modem monitor file and overwrite options.

Specifying options for the Monitor File

About this task

You can specify options for the monitor file that override the default settings. The monitor file name is based on the prefix of the associated data file and the **.mon** file extension.

To specify the options for the monitor file

Procedure

1. Add a VAN Post Office as described in ["Adding a VAN Post Office"](#) and click the **Configure** button.
2. Complete all required fields in Connection tab on the VAN Source Connection dialog box as described in ["VAN Source Configuration"](#).
3. Select the **Diagnostics** tab.
4. Select a **Monitor Mode**:
 - a. **NONE** - no monitor file is produced
 - b. **APPEND** - information is added to the existing monitor file each time the adapter is run
 - c. **OVERWRITE** - any existing monitor file is overwritten each time the adapter is run
- 5.
6. Click **OK**.
7. Click **Save**.

Other VAN source configuration options

Two other tabs on the VAN Source Configuration dialog box allow you to set the following options:

- **Options** - allows you to specify a configuration file other than the default **aplus.cfg**. Leave this field blank.
- **Trace** - allows you to specify a trace file name and append option for a trace file containing diagnostic information.

Trade links

Trade links contain the partner addresses, post offices used, standards information, control and tracking information, and current test or production status for each individual document that you exchange with that partner

• [Trade Links Overview](#)

Use the **Trade Links** navigator to define and maintain trading relationships.

• [Inbound trade links](#)

- [Inbound trade link applications](#)
 - [Outbound trade links](#)
 - [ISA and GS Numbering Schemes](#)
 - [Searching for a trade link](#)
 - [Automatic acknowledgement creation](#)
Partner Manager will display the **Automatic Acknowledgement Creation** dialog if the trade link for that acknowledgement does not already exist and an acknowledgement is expected from your external trading partner, or you are creating an acknowledgement.
 - [Outbound CONTROL trade link](#)
 - [Creating a control number map](#)
-

Trade Links Overview

Use the **Trade Links** navigator to define and maintain trading relationships.

The **Trade Links** navigator enables you to coordinate the trading partners, post offices, and additional information to define a trading relationship. The navigator contains **Inbound** and **Outbound** trade links.

This chapter describes how to use the **Trade Links** navigator to define EDIFACT, TRADACOMS, and X12 trade links.

- [Establishing trade links](#)
Establishing trade links is automated by two wizards that guide you through the process, to ensure that the resulting link provides a logical connection.
 - [Exporting trade links](#)
Trade links may be exported into an XML file. Such exports are useful when communicating issues with support, or for your own information extraction needs.
-

Establishing trade links

Establishing trade links is automated by two wizards that guide you through the process, to ensure that the resulting link provides a logical connection.

The **Trade Link Add Wizard** helps you define the trade link in the database. The **EDI Wizard** is used for X12 and EDIFACT trade links. This wizard operates on messages that you define, to produce the correct type tree for the correct version and release of the standard for the Message Manager. The **EDI Wizard** produces two copies of the e-commerce type tree: one tree used by the Message Manager and another tree that is available for your use in application maps.

Exporting trade links

Trade links may be exported into an XML file. Such exports are useful when communicating issues with support, or for your own information extraction needs.

However, due to the complexity of trade links, importing is not currently supported, except by IBM Support using IBM tools. Exporting a trade link will also export to the opposite direction (inbound vs. outbound) trade link if one exists, so that any acknowledgement link is also included. In addition to the trade links, the current configuration settings, post offices, and applicable standards data are included in the XML file so that it reports a current snapshot of your trade link. For more information about the format of the XML generated by the export process, see the XML Autoload Utilities documentation.

To export a trade link:

1. Display any type of trade link desired.
2. Click the **Export** button.
3. Click **OK**.

Note: Export files can be appended to, if desired. If the file selected already exists, you will be given the option to overwrite, or append.

The following objects are not exported during this process:

- Security information, such as user names or passwords
 - Display options
 - Field defaults
 - Traffic
 - Server directory configuration data, such as FTP accounts/passwords
-

Inbound trade links

Inbound indicates that your company (also known as an internal trading partner) is receiving data from another entity (also known as an external trading partner). A basic inbound trade link consists of the following information:

- The external trading partner from which you receive the data.
- Which internal trading partner receives the data.
- Where Trading Manager gets its data (Get post office) and puts its data (Put post office).
- What type(s) of information the two trading partners are exchanging.

This section describes:

- ["X12 Inbound Trade Links"](#)
- ["EDIFACT Inbound Trade Links"](#)
- ["TRADACOMS Inbound Trade Links"](#)

- [X12 Inbound Trade Links](#)
 - [EDIFACT Inbound Trade Links](#)
 - [TRADACOMS Inbound Trade Links](#)
-

X12 Inbound Trade Links

The section describes how to add, edit, copy, and delete X12 Inbound Trade Links:

- To add X12 Inbound Trade Links see "[Adding an X12 Inbound Trade Link](#)"
 - To edit X12 Inbound Trade Links see "[Editing an X12 Inbound Trade Link](#)"
 - To copy X12 Trade Links see page "[Copying an X12 Inbound Trade Link](#)"
 - To delete X12 Trade Links see page "[Deleting an X12 Inbound Trade Link](#)"
 - [Adding an X12 Inbound Trade Link](#)
 - [Editing an X12 Inbound Trade Link](#)
 - [Copying an X12 Inbound Trade Link](#)
 - [Deleting an X12 Inbound Trade Link](#)
-

Adding an X12 Inbound Trade Link

About this task

An inbound trade link is defined as one in which your (external) partner is sending data to you (the internal partner). You are the receiver of the incoming data.

To add an X12 inbound trade link

Procedure

1. From the File menu, select **Trade Links**.
The **Trade Links** navigator appears.
 2. Click Inbound > **X12** in the **Trade Links** navigator.
The X12 Inbound Trade Links window opens.
 3. Click Add Link.
The **X12 Inbound Trade Links Add Wizard** opens.
 4. Follow the on-screen instructions in the Wizard to add an X12 Trade Link.
 5. When you click Finish to complete adding an inbound trade link using the Wizard, the X12 Trade Link: Add Application window opens.
 6. Select a version and functional ID.
 7. Click OK.
You are returned to the X12 Inbound Trade Links window.
 8. Select either the **Test** or **Production** check box to activate the associated **Test Post Office** or **Production Post Office** field.
 9. Select a post office from the drop-down menu on the selected field.
Note: The **Test Post Office** field, and the **Production Post Office** field are mandatory depending upon which Test/Production check boxes are enabled. One of these fields must be completed before you can save the Trade Link.
 10. If you want to make changes to any of the fields in the window related to acknowledgements and validation, click the Acknowledgements/Validate By tab to display those fields.
 11. If you are adding a HIPAA X12 document, click the HIPAA Validation Types tab to display the HIPAA fields.
Note: In the preceding screen, it should be noted that the fields in the **HIPAA Validation Types** area are active only when a HIPAA Version and a Functional ID Code other than FA is selected. These fields display when the HIPAA Validation Types tab is selected. These fields are described in "[Editing a HIPAA X12 Document](#)".
 12. Click Save.
The Trade Link is saved.
-

Editing an X12 Inbound Trade Link

About this task

The following instructions describe how to edit existing trade links.

To edit an inbound trade link

Procedure

1. From the File menu, select **Trade Links**.
2. In the **Trade Links** navigator, expand **Inbound > X12**.
3. Find the desired trade link by using the search buttons at the top of the window. See "[Searching for an X12 Trade Link](#)" for additional search information.
After making your selection, the Search dialog closes and you are returned to the X12 Inbound Trade Links window.
4. Make your changes.

5. Click Save.
A message box opens that informs you that your updates have been made.

6. Click OK.
The Trade Link is updated.

Copying an X12 Inbound Trade Link

About this task

The following instructions describe how to copy existing trade links.

To copy an inbound trade link

Procedure

1. From the File menu, select **Trade Links**.
2. In the **Trade Links** navigator, expand **Inbound > X12**.
The **X12 Inbound** Trade Links window opens.
3. Click the **Copy** button in the trading partner area of the window. The Copy X12 Inbound Trade Link window opens.
4. Click **Find All** in the **New 'From' Trading Partner** area to open the X12 Partner Lookup window, or click **Search** for a more refined search.
5. Highlight a trading partner and click **Select**.
6. Click **Find All** in the **New 'To' Trading Partner** area, or click **Search** for a more refined search.
7. Highlight a partner and click **Select**.
8. Click **Begin Copy**.
9. If the link being copied has no acknowledgement partners, a message box opens that informs you that the copy was successful.
10. If the selected link contains an acknowledgement partner, the **Select Acknowledgement Partner** dialog opens.

You have the following options:

- **Set to Same as 'From' Partner** - you can use the same Acknowledgement partner as the new From partner
- **Select a New Acknowledgement Partner** - you select a different Acknowledgement partner; the Partner Lookup dialog box appears if you click this button
- **Set This Application to 'No Acknowledgement'** - you can set the application so that it has no acknowledgement

11. Make your selection and click **OK**.
A message box opens that informs you that the copy was successful.
12. Click **OK** to close the message box.
The Trade Link is copied.

Deleting an X12 Inbound Trade Link

About this task

The following instructions describe how to delete existing trade links.

Note: Deleting a link deletes all applications and transactions associated with the trade link. The delete operation cannot be reversed.

To delete an X12 inbound trade link

Procedure

1. From the File menu, select **Trade Links**.
2. In the **Trade Links** navigator, expand **Inbound** and **X12**.
3. Find the trade link that you want to delete by using the search buttons at the top of the window. See "[Searching for an X12 Trade Link](#)" for additional search information.
4. Highlight a link.
5. Click **Select**.
6. Click **Delete**.
The **Trade Link Confirmation** message box opens to ask if you really want to delete the trade link.
7. Click **Yes** to delete the trade link, including all applications and transaction sets associated with it.
A message box opens that informs you that the trade link was successfully deleted.
8. Click **OK** to close the message box.

EDIFACT Inbound Trade Links

The section describes how to add, edit, copy, and delete EDIFACT Inbound Trade Links.

- [Adding an EDIFACT Inbound Trade Link](#)
- [Editing an EDIFACT Inbound Trade Link](#)
- [Copying an EDIFACT Inbound Trade Link](#)

- [Deleting an EDIFACT Inbound Trade Link](#)

Adding an EDIFACT Inbound Trade Link

About this task

An inbound trade link is defined as one in which your (external) partner is sending data to you (the internal partner). You are the receiver of the incoming data. To add an EDIFACT inbound trade link, you use the **Link Wizard**.

To add an EDIFACT Inbound Trade Link

Procedure

1. From the File menu, select Trade Links.
The **Trade Links** navigator opens.
2. Click Inbound > **EDIFACT** in the **Trade Links** navigator.
The EDIFACT Inbound Trade Links window opens.
3. Click Add Link.
The **EDIFACT Inbound Trade Links Add Wizard** opens.
4. Follow the on-screen instruction in the Wizard to add an EDIFACT Trade Link. If the EDIFACT trading partners have an application partner, you will be allowed to define whether a UNG should be absent, required, or optional, the default no longer being NO UNG.
5. When you click Finish to complete adding an inbound trade link using the Wizard, the EDIFACT Trade Link: Add Application window opens.
6. Add applications defining the message types you receive from the external trading partner.
7. The **Version/Message Type** group box includes the four components of the EDIFACT version that you defined for the EDIFACT Standard Version.
 - Click Find Version/Message Types to select an existing EDIFACT version, or
 - Select the components from the **Version/Message Type** lists that define the version or message type.
8. Click OK. The EDIFACT Inbound Trade Links window is displayed with the newly added trade link displayed.
9. Click Save.
The EDIFACT Trading Partner is added.

Editing an EDIFACT Inbound Trade Link

About this task

The following instructions describe how to edit existing EDIFACT inbound trade links.

To edit an EDIFACT inbound trade link

Procedure

1. From the File menu, select **Trade Links**.
2. In the **Trade Links** navigator, expand **Inbound** and select **EDIFACT**.
3. Find the desired trade link by using the search buttons at the top of the window. See ["Searching for an EDIFACT Trade Link"](#) for additional search information.
After you have selected the trading partner that you want to edit, it appears in the EDIFACT Inbound Trade Links window.
4. Make your changes.
5. Click Save.
A message box opens that informs you that your updates have been made.
6. Click OK.
The Trade Link is updated.

Copying an EDIFACT Inbound Trade Link

About this task

The following instructions describe how to copy existing trade links.

To copy an inbound trade link

Procedure

1. From the File menu, select **Trade Links**.
2. In the **Trade Links** navigator, expand **Inbound** > **EDIFACT**.
3. Click the **Copy** button in the trading partner area of the window. The Copy EDIFACT Inbound Trade Link window opens.
4. Click Find All in the **New 'From' Trading Partner** area to open the EDIFACT Partner Lookup window, or click Search for a more refined search
5. Highlight a trading partner and click Select.

6. Click Find All in the **New 'To' Trading Partner** area, or click Search for a more refined search.
7. Highlight a partner and click Select.
8. Click Begin Copy.
A message box will open to inform you that your copy was successful.
9. Click OK.
The EDIFACT Trading Partner is copied.

Deleting an EDIFACT Inbound Trade Link

About this task

The following instructions describe how to delete existing trade links.

Note: Deleting a link deletes all applications and transactions associated with the trade link. The delete operation cannot be reversed.

To delete an EDIFACT inbound trade link

Procedure

1. From the File menu, select **Trade Links**.
2. In the **Trade Links** navigator, select **Inbound > EDIFACT**.
The EDIFACT Inbound Trade Link window opens.
3. Find the trade link to be deleted by using the search buttons at the top of the window. See "[Searching for an EDIFACT Trade Link](#)" for additional search information.
After you select the link, the search window closes and you are returned to the EDIFACT Inbound Trade Link window with the selected link displayed.
4. Click **Delete**.
The **Trade Link Confirmation** message box opens to ask if you really want to delete the trade link.
5. Click **Yes** to delete the trade link, including all applications and transaction sets associated with it.
A message box opens that informs you that the trade link was successfully deleted.
6. Click **OK** to close the message box.

TRADACOMS Inbound Trade Links

The section describes how to add, edit, copy, and delete TRADACOMS Inbound Trade Links:

- To add a TRADACOMS Inbound Trade Links, see "[Adding a TRADACOMS Inbound Trade Link](#)".
- To edit a TRADACOMS Inbound Trade Links, see "[Editing TRADACOMS Inbound Trade Links](#)".
- To copy a TRADACOMS Inbound Trade Links, see "[Editing TRADACOMS Inbound Trade Links](#)".
- To delete a TRADACOMS Inbound Trade Links, see "[Deleting TRADACOMS Inbound Trade Links](#)".
- [Adding a TRADACOMS Inbound Trade Link](#)
- [Editing TRADACOMS Inbound Trade Links](#)
- [Copying TRADACOMS Inbound Trade Links](#)
- [Deleting TRADACOMS Inbound Trade Links](#)

Adding a TRADACOMS Inbound Trade Link

About this task

An inbound trade link is defined as one in which your (external) partner is sending data to you (the internal partner). You are the receiver of the incoming data. The procedure uses the **Link Wizard**.

Note: Because application partners are optional in TRADACOMS, if SDT/CDT segments are expected, select the trading partner that also has an application partner associated with it.

To add a TRADACOMS inbound trade link

Procedure

1. From the File menu, select **Trade Links**.
The **Trade Links** navigator opens.
2. Click **Inbound > TRADACOMS** in the **Trade Links** navigator.
The TRADACOMS Inbound Trade Links window opens.
3. Click **Add Link**.
The **TRADACOMS Inbound Trade Links Add Wizard** opens.
4. Follow the on-screen instruction in the Wizard to add a TRADACOMS Trade Link.
5. When you click **Finish** to complete adding an inbound trade link using the Wizard, the TRADACOMS Trade Link: Add Application window opens.

6. Select a Document Type by selecting a type from the **Document Type** field drop-down menu.
You can access a list of all TRADACOMS document types by clicking the **Find** button.
7. Click OK.
The TRADACOMS Trade Link: Add Application window closes and you are returned to the TRADACOMS Inbound Trade Links window with the Application tab displayed.
8. Select a Get Post Office using the **Get Post Office** field drop-down menu.
Note: The **Get Post Office** field is mandatory and must be entered before the Trade Link can be saved.
9. Select a Put Post Office.
If the **Test** check box is enabled, you must enter a Post Office in the **Test Put Post Office** field

If the **Production** check box is enabled, you must enter a Post Office in the **Production Put Post Office** field.

Note: The **Get Post Office** and **Production Put Post Office** fields are mandatory depending upon which Test/Production check boxes are enabled. One of these fields must be completed before you can save the Trade Link.
10. If you want to make changes to customer or supplier details, click the SDT/CDT & VAT tab.
11. Click Save.
The **Trade Link Saved** message box opens to inform you that the trade link is saved.
12. Click OK.
The message box closes and the Trade Link is saved.

Editing TRADACOMS Inbound Trade Links

About this task

The following instructions describe how to edit existing EDIFACT inbound trade links.

To edit a TRADACOMS inbound trade link

Procedure

1. From the File menu, select **Trade Links**.
2. In the **Trade Links** navigator select **Inbound >TRADACOMS**.
3. Find the trade link to be edited by using the search buttons at the top of the window. See "[Searching for a TRADACOMS Trade Link](#)" for additional search information.
After you have selected the trading partner that you want to edit, it appears in the TRADACOMS Inbound Trade Links window.
4. Make your changes.
5. Click Save.
A message box opens that informs you that your updates have been made.
6. Click OK.
The Trade Link is updated.

Copying TRADACOMS Inbound Trade Links

About this task

The following instructions describe how to copy existing trade links.

To copy a TRADACOMS inbound trade link

Procedure

1. From the File menu, select **Trade Links**.
2. In the **Trade Links** navigator, expand **Inbound > TRADACOMS**.
The TRADACOMS Inbound Trade Link window opens.
3. Click the **Copy** button in the trading partner area of the window.
The Copy TRADACOMS Inbound Trade Link window opens.
4. Click Find All in the **New 'From' Trading Partner** area to open the TRADACOMS Partner Lookup window, or click Search for a more refined search
5. Highlight a trading partner and click Select.
6. Click Find All in the **New 'To' Trading Partner** area, or click Search for a more refined search).
7. Highlight a partner and click Select.
8. Click Begin Copy.
A message box will open to inform you that your copy was successful.
9. Click OK.
The TRADACOMS Trade Link is copied.

Deleting TRADACOMS Inbound Trade Links

About this task

The following instructions describe how to delete existing trade links.

Note: Deleting a link deletes all applications and transactions associated with the trade link. The delete operation cannot be reversed.

To delete a TRADACOMS inbound trade link

Procedure

1. From the File menu, select **Trade Links**.
2. In the **Trade Links** navigator, select **Inbound > TRADACOMS**.
3. Find the desired trade link by using the search buttons at the top of the window. See "[Searching for a TRADACOMS Trade Link](#)" for additional search information.
After you have selected the trade link to be deleted, the search window closes and you are returned to the TRADACOMS Inbound Trade Link window with the selected trade link displayed.
4. Click **Delete**.
The **Trade Link Confirmation** message box opens to ask if you really want to delete the trade link.
5. Click **Yes** to delete the trade link, including all applications and transaction sets associated with it.
A message box opens that informs you that the trade link was successfully deleted.
6. Click **OK** to close the message box.

Inbound trade link applications

One or more business documents are exchanged between an internal and external trading partner, with each business document defined as an application within the trading relationship. The initial applications are added when you run the **Inbound Trade Links Wizard**. You can add more applications for a given external and internal trading partner from the Inbound Trade Links: Partner View dialog box.

- [X12 Inbound Trade Link Applications](#)
- [EDIFACT Inbound Trade Link Applications](#)
- [TRADACOMS Inbound Trade Link Applications](#)

Refer to the following procedures to add, edit, and delete TRADACOMS Inbound Trade Link Applications.

X12 Inbound Trade Link Applications

Refer to the following procedures to add, edit, and delete X12 Inbound Trade Link Applications.

- [Adding an X12 Inbound Trade Link Application](#)
- [Editing an X12 Inbound Trade Link Application](#)
- [Editing a HIPAA X12 document](#)
- [Deleting an X12 Inbound Trade Link Application](#)

Adding an X12 Inbound Trade Link Application

About this task

One or more business documents are exchanged between an internal and external trading partner, with each business document defined as an application within the trading relationship.

The initial applications are added when you run the **Inbound Trade Links Wizard**. You can add more applications for a given external and internal trading partner from the Inbound Trade Links: Partner View dialog box.

Use the following instructions to add a new X12 application for a trade link.

To add a new X12 inbound trade link application

Procedure

1. From the File menu, select **Trade Links**.
2. In the **Trade Links** navigator select **Inbound > X12**.
The X12 Inbound Trade Links window opens.
3. Click the Applications tab.
The Applications tab view opens.
4. Click Add Application.
The X12 Trade Link: Add Application window opens.

5. Select a Version and Functional ID then click OK.
 6. Specify the data state to be exchanged. At least one of the two check boxes must be checked
 - Enable **Test**, if exchanging test data
 - Enable **Production**, if exchanging production data
 7. If you enabled the **Test** or **Production** indicators, choose the **Put Post Office** where the application-ready EDI data received in Test or Production mode is placed.
 8. If desired you can click the Acknowledgements/Validate By tab and enter any information. Refer to the X12 Functional Acknowledgement table following these procedures for a description of the X12 FAs.
 9. If you are adding a HIPAA Inbound Trade Link Application, click the HIPAA Validation Types tab and set the fields as desired. Refer to "[Editing a HIPAA X12 Document](#)" for more information.
 10. If you do not want to perform data validation, but you want to route the data, check the **Route Only; No Validation** check box. See [Route only option](#), for more information.
 11. Click Save.
- The X12 Inbound Trade Links window displays the new trade link application in the list.
- [Route only option](#)
- The route only option does not ignore all possible errors in the data file.

Levels of X12 Functional Acknowledgement (FA)

About this task

The following are levels of X12 functional acknowledgements:

| Level | Description |
|------------------------|--|
| No Acknowledgement | No FA is sent |
| Functional Group Level | An FA is sent acknowledging receipt of the functional group, providing details of the inbound data to only the functional group (AK1) level. |
| Transaction Set Level | An FA is sent acknowledging receipt of the functional group, providing details of the inbound data to the transaction set (AK2) level. |
| Segment Level | An FA is sent acknowledging receipt of the functional group, providing details of the inbound data to the segment (AK3) level. |
| Data Element Level | An FA is sent acknowledging receipt of the functional group, providing details of the inbound data to the data element (AK4) level. |

If you select an FA to be sent, specify the receiving external trading partner. By default, functional acknowledgements are sent to the X12 external trading partner in the trade link.

To send acknowledgements to an address other than the external trading partner

Procedure

1. In the X12 Inbound Trade Link window Application tab view
 2. Click the **Find Ack To** button
 3. The X12 Partner Lookup in Folder window opens.
 4. Enter search criteria in the window, then click Execute to display partners.
 5. Click Select to close the Lookup window.
- The X12 Inbound Trade Links window displays the new application.

Route only option

The route only option does not ignore all possible errors in the data file.

Tradelink errors, envelope errors, and structural errors that do not allow data to be recognized as valid recognizable data are *not* handled by the **Route only** option. These errors are captured in the **stampandsort** map. The function of this map is to create a tradelink, validate the envelope recognized supported type of data to be processed by Trading Manager (X12; EDIFACT; TRADACOMS), and to create the temporary files that the remaining Trading Manager validation process uses.

During each tradelink validation process (maps **sendxx** or **sortxx**), the logic determines if the flag for Route Only was set 'up'. If not, the logic continues to validate each transaction segment to report validation results within the acknowledgement and/or rejection files. If the Route Only option is turned 'on', the acknowledgments/rejects will not get created if there are errors in the data within transaction segments.

A typical example of this scenario would be the presence of BIN segments. A BIN segment consists of two elements, BIN01 and BIN02. BIN01 is a numeric value that should indicate the size of element BIN02 so that it is known where this element ends. The reason is that the BIN02 is a binary element, and it might contain characters that match any of the delimiters or other segment initiators. If the BIN01 element is incorrect, the parsing of this segment will result in a structurally invalid X12 file, thus the data will fail to be recognized during the initial process where the ROUTE only option is still not in effect.

Editing an X12 Inbound Trade Link Application

About this task

To edit an inbound X12 inbound trade link application

Procedure

1. From the File menu, select **Trade Links**.
 2. In the **Trade Links** navigator select **Inbound > X12**.
- The X12 Inbound Trade Links window opens.
3. Click the Applications tab.
 4. Select a trade link application from the list.

5. Make changes to any of the fields in **Post Offices/Routing: Acknowledgements/Validate By**; and **HIPAA Validation Types** (HIPAA users only) tab views.
6. When the information is complete, click **Save**.
The **Trade Link Saved** message box will open to tell you that your changes have been saved.
7. Click **OK** to close the message box.

Results

The following are **Validate By** group box options:

Interchange

If the transaction is bad, the whole interchange will be removed.

Message

EDIFACT only. If a transaction is bad, only that transaction will be removed. All remaining good data will be processed.

Transaction

X12 only. If a transaction is bad, only that transaction will be removed. All remaining good data will be processed.

Functional Group

If a transaction is bad, the functional group that contained the bad transaction will be removed and the remaining good data will be processed.

Editing a HIPAA X12 document

The selections that you make in the **HIPAA Validation Types** area of the Edit X12 Inbound Trade Link window are used by Message Manager to determine the validation level of incoming HIPAA data.

Note: These fields are enabled *only* when the Inbound Group is for a HIPAA Version with a Functional ID code other than FA. Additionally, to enable these drop-down menus, at least one Structure Validation record must be imported, or manually added for levels 1 and 2 and at least one HIPAA Type Qualifiers Import (**hipaa_x12_4010_qualifier.dat**), or maintenance must be performed for levels 3; 4; 6; and 7, with one exception. In this exception, type 5 will be enabled if type 2, 3, or 4 is enabled and a HIPAA Type Qualifiers Import was run. For details related to the HIPAA Type Qualifiers Import, see the Chapter 3 of the *Trading Manager Installation and User Guide*.

You may select one of the following values for these fields:

- Type 1: **Disabled; Enabled; or Reported in 997**
- Type 2: **Disabled; Enabled; Enabled, Reported in 997; or Enabled, Reported in 999**
- Type 3: **Disabled; Enabled; Enabled; Reported in 824; Enabled; Reported in 999; or Enabled, Execute Map**
Note: If **Enabled, Execute Map** is specified in Type 3, the Type 3 Map field must contain a value.
- Type 4: **Disabled; Enabled; Enabled, Reported in 824; Enabled, Reported in 999**
- Type 5: **Disabled; Enabled; Enabled, Reported in 997; Enabled, Reported in 999**
- Type 6: **Disabled; Enabled; Enabled, Reported in 824; Enabled, Reported in 999**
- Type 7: **Disabled; Enabled; Enabled, Reported in 824; Enabled, Reported in 999**

You can only specify Type 7 if the External Application Partner for this link has a "HIPAA Institution" set.

If any 824 reporting was selected, a box will be enabled to allow you to specify the version for the 824 transaction. The default version is 4050, but it can be changed to a value in the drop down list (currently versions 4050 and 4010 are available).

For tradelinks for HIPAA version V4050X151 and functional ID "PI", if a type 4 824 or 999 report was selected, another check box in the HIPAA validation window is enabled to allow you to define how to perform validation for the BIN segments that might be present in those transactions. The default value is to perform no validation (Value=NEVER), but you can also specify the values "X" to perform only format validation, or "S" to perform validation against a Schema for both version and content. It is important in the last two cases to have the required schema file located in the same directory as the Message Manager maps.

Deleting an X12 Inbound Trade Link Application

About this task

The following instructions describe how to delete an existing X12 trade link application.

To delete an X12 inbound trade link application

Procedure

1. From the File menu, choose **Trade Links**.
2. In the **Trade Links** navigator select Inbound > **X12**.
The X12 Inbound Trade Links window opens.
3. Click the Applications tab.
4. Select a trade link application from the list.
5. Click the **Delete Application** button.
The **Trade Links Application Removal Confirmation** box opens. The box tells you that you must **Save** after closing the box to complete removal.
6. Click **OK**.
The message box closes.
7. Click **Save**.
The trade link application is deleted.

EDIFACT Inbound Trade Link Applications

Refer to the following procedures to add, edit and delete EDIFACT Inbound Trade Link Applications

- [Adding an EDIFACT Inbound Trade Link Application](#)
- [Editing an EDIFACT Inbound Trade Link Application](#)
- [Deleting an EDIFACT Inbound Trade Link Application](#)

Adding an EDIFACT Inbound Trade Link Application

About this task

One or more business documents are exchanged between an internal and external trading partner, with each business document defined as an application within the trading relationship.

The initial applications are added when you run the **Inbound Trade Links Wizard**. You can add more applications for a given external and internal trading partner from the Inbound Trade Links: Partner View dialog box.

Use the following instructions to add a new EDIFACT application for a trade link.

To add a new EDIFACT inbound trade link application

Procedure

1. From the File menu, select **Trade Links**.
2. In the **Trade Links** navigator select **Inbound > EDIFACT**.
The EDIFACT Inbound Trade Links window opens.
3. Click the Applications tab.
4. Click the **Add Application** button.
The EDIFACT Trade Link: Add Application window opens. If the No UNG option is set, the two fields in the **Application Partners** area will be grayed out.
5. If this is an optional or mandatory UNG link, and an UNG segment is expected for this application, select the Application Partners associated with this UNG segment in the drop-down list.
6. Complete the **Msg Version No**, **Msg Release No**, and **Controlling Agency** fields.
7. Click the **Find Version Message Types** button to open the **Find All EDIFACT Functional ID Message Types** list box and select a message type from the list.
8. Click OK.
9. With the Post Offices tab displayed select test or production data by enabling one or both **Test** and/or **Production** check boxes.
10. Select a Test Post Office and/or Production Post Office. You must select a test or production Post Office in order to save the trade link application.
11. If data is not to be validated, but is to be routed to the Post Office, enable the **Route Only; No Validation** check box. See [Route only option](#) for details.
12. If validation is desired, click the Acknowledgements/Validate By tab and make changes to any fields in this view.
13. Click Save.
The new Inbound Trade Link Application is now created.

Levels of EDIFACT Acknowledgement (CONTRL)

About this task

The following list identifies the levels of EDIFACT functional acknowledgements. Please note that CONTRL, as used in this section, is *not* a typographical error.

| Acknowledgment | Description |
|------------------------|---|
| No Acknowledgment | No CONTRL is sent |
| Interchange Level | A CONTRL is sent acknowledging receipt, providing details of the inbound data to only the interchange (UCI) level. |
| Functional Group Level | This option is only available if this is a UNG level document, that is, application partners are specified. A CONTRL is sent acknowledging receipt, providing details of the inbound data to only the functional group (UCF) level. |
| Message Level | A CONTRL is sent acknowledging receipt, providing details of the inbound data to the message (UCM) level. |
| Segment Level | A CONTRL is sent acknowledging receipt, providing details of the inbound data to the segment (UCS) level. |
| Data Element Level | A CONTRL is sent acknowledging receipt, providing details of the inbound data to the data element (UCD) level. |

Editing an EDIFACT Inbound Trade Link Application

About this task

To edit an inbound EDIFACT trade link application

Procedure

1. From the File menu, select **Trade Links**.
2. In the **Trade Links** navigator select Inbound > **EDIFACT**.
The EDIFACT Inbound Trade Links window opens.
3. Click the Applications tab.
4. Select a trade link application from the list.
5. Make changes to any of the fields in **Post Offices/Routing; Acknowledgements/Validate By**; and Application Partners tab views.
6. When the information is complete, click **Save**.
The **Trade Link Saved** message box will open to tell you that your changes have been saved.
7. Click **OK** to close the message box.

Deleting an EDIFACT Inbound Trade Link Application

About this task

The following instructions describe how to delete an existing EDIFACT trade link application.

To delete an EDIFACT inbound trade link application

Procedure

1. From the File menu, choose **Trade Links**.
2. In the **Trade Links** navigator select Inbound > **EDIFACT**.
The EDIFACT Inbound Trade Links window opens.
3. Click the Applications tab.
4. Select a trade link application from the list.
5. Click the **Delete Application** button.
The **Trade Links Application Removal Confirmation** box opens. The box tells you that you must **Save** after closing the box to complete removal.
6. Click **OK**.
The message box closes.
7. Click **Save**.
The trade link application is deleted.

TRADACOMS Inbound Trade Link Applications

Refer to the following procedures to add, edit, and delete TRADACOMS Inbound Trade Link Applications.

- [Adding a TRADACOMS Inbound Trade Link Application](#)
- [Editing a TRADACOMS Inbound Trade Link Application](#)
- [Deleting a TRADACOMS Inbound Trade Link Application](#)

Adding a TRADACOMS Inbound Trade Link Application

About this task

One or more business documents are exchanged between an internal and external trading partner, with each business document defined as an application within the trading relationship.

The initial applications are added when you run the **Inbound Trade Links Wizard**. You can add more applications for a given external and internal trading partner from the Inbound Trade Links: Partner View dialog box.

Use the following instructions to add a new EDIFACT application for a trade link.

To add a new TRADACOMS inbound trade link application

Procedure

1. From the File menu, select **Trade Links**.
2. In the **Trade Links** navigator select Inbound > **TRADACOMS**.
The TRADACOMS Inbound Trade Links window opens.
3. Click the Application tab.
4. Click the **Add Application** button.
The TRADACOMS Trade Link: Add Application window opens.
5. Select a document type from the **Document Type** field.

6. Click OK.
7. Add data as desired in the fields in the SDT/CDT & VAT tab view
8. With the Post Offices tab displayed, select test or production data by enabling one or both **Test** and/or **Production** check boxes.
9. Select a **Test Post Office** and/or **Production Post Office**.
10. Click Save.

Editing a TRADACOMS Inbound Trade Link Application

About this task

To edit an inbound TRADACOMS trade link application

Procedure

1. From the File menu, select **Trade Links**.
2. In the **Trade Links** navigator select **Inbound > TRADACOMS**.
The EDIFACT Inbound Trade Links window opens
3. Click the Applications tab.
4. Select a trade link application from the list.
5. Make changes to any of the fields in **Post Offices** and SDT/CDT & VAT tab views.
6. When the information is complete, click **Save**.
7. The **Trade Link Saved** message box will open to tell you that your changes have been saved.
8. Click **OK** to close the message box.

Deleting a TRADACOMS Inbound Trade Link Application

About this task

The following instructions describe how to delete an existing TRADACOMS trade link application.

To delete a TRADACOMS inbound trade link application

Procedure

1. From the File menu, choose **Trade Links**.
2. In the **Trade Links** navigator select **Inbound > TRADACOMS**.
The EDIFACT Inbound Trade Links window opens.
3. Click the Applications tab.
4. Select a trade link application from the list.
5. Click the **Delete Application** button.
The **Trade Links Application Removal Confirmation** box opens. The box tells you that you must **Save** after closing the box to complete removal.
6. Click **OK**.
The message box closes.
7. Click **Save**.
The trade link application is deleted.

Outbound trade links

When you define **Outbound Trade Links**, outbound indicates that your company (also known as an internal trading partner) is sending data to another entity (also known as an external trading partner). A basic outbound trade link consists of the following information:

- Which internal trading partner is sending the data
- Which external trading partner is receiving the data
- Where Trading Manager gets its data (Get post office) and puts its data (Put post office)
- What type(s) of information the two trading partners are exchanging
- **X12 Outbound Trade Links**
X12 Outbound Trade Links can be added, edited, copied, and deleted.
- **EDIFACT Outbound Trade Links**
EDIFACT Outbound Trade Links can be added, edited, copied, and deleted.
- **EDIFACT Outbound Trade Links Applications**
You can add, edit, and delete EDIFACT Outbound Trade Link Applications.

X12 Outbound Trade Links

X12 Outbound Trade Links can be added, edited, copied, and deleted.

- [Adding an X12 Outbound Trade Link](#)
- [Editing an X12 Outbound Trade Link](#)
- [Copying an X12 Outbound Trade Link](#)
- [Deleting an X12 Outbound Trade Link](#)

Adding an X12 Outbound Trade Link

About this task

To add an X12 outbound trade link requires the following steps:

- Select the external and internal trading partners with associated application partners.
- Select the **Get** and **Put** post offices:
- The **Get** post office is where Message Manager retrieves the EDI data.
- The **Put** post office provides the routing instructions for the EDI data produced by Message Manager (or how Message Manager sends the data).
- Select the interchange control (ISA) and functional group (GS) numbering schemes.
- Add applications defining the message types you receive from the external trading partner.

To select external and internal X12 trading partners

Procedure

1. From the File menu, select Trade Links.
2. In the **Trade Links** navigator select Outbound > X12.
3. Click Add Link.
The **Add an X12 Outbound Link Wizard** opens.
4. Follow the on-screen instructions in the Wizard to add an X12 Trade Link.
5. When you click Finish to complete adding an inbound trade link using the Wizard, the X12 Trade Link: Add Application window opens.
6. Select a version and functional ID for the application partner.
7. Select a version and functional ID for this outbound link. Click OK to return to the X12 Outbound Trade Links window.
8. If you want to add another application click the **Add Application** button.
9. When you are finished adding applications, select a Put Post Office for the selected version and functional group (FG). At least one Put Post Office is required per version and FG. You may select a Test and/or Production Post Office. Message manager will route the data to this post office based on the Test/Production indicator in the X12 data. You may also specify if this application should be Route Only. Setting Route Only bypasses all validation for the selected version and functional group. See [Route only option](#) for details.
10. In the X12 Outbound Trade Links window, in the Application tab, and click the Format, GS/ST Control tab.
This tab view contains the **Application Format** options and the **GS Time Stamp Options**.

You can choose one of the following Application Format enveloping options by selecting the associated button:

- **Fully Enveloped** - This option indicates that the data passed to Trading Manager for outbound processing is already fully enveloped with terminators/delimiters, ISA envelope information, and Control numbers. Trading Manager makes no updates, leaving this enveloping information as is.
 - **Control Number Enveloped** - This option indicates that terminator/ delimiter enveloping is set in the data, but Trading Manager should generate a proper control number at the Interchange, Functional Group, and Transaction level.
 - **Partially Enveloped** - Trading Manager envelopes data with terminators/delimiters as specified in Partner Manager, and generates a proper control number at the Interchange level, Functional Group level, and Transaction level. ISA control fields are also set based on information in Partner Manager.
- You can change the **GS Time Stamp Option** by selecting one of the following buttons. **HHMM**; **HHMMSS**; **HHMMSSD**; **HHMMSSDD**.

Time is expressed in 24-hour clock time as follows: HHMM, or HHMMSS, or HHMMSSD, or HHMMSSDD, where H = hours (00-23), M = minutes (00- 59), S = integer seconds (00-59) and DD = decimal seconds; decimal seconds are expressed as follows: D = tenths (0-9) and DD = hundredths (00-99)

The **GS Control By Application** fields may be changed if the **GS Group Control # Schema** is set to Increment By Application. This scheme is set in the main tab labeled **GS Control**.

11. You can select the Format tab view and the Transactions tab view and modify any of the default settings of fields in these tab views.
12. Select the **Identification**, **GS Control**, **ISA Control** and Applications tab and make any desired changes to any of the fields.
13. Click Save.

Editing an X12 Outbound Trade Link

About this task

The following instructions describe how to edit existing X12 Outbound trade links.

To edit an X12 outbound trade link

Procedure

1. From the File menu, select **Trade Links**.
2. In the **Trade Links** navigator, expand **Outbound** and **X12**.
3. Find the desired trade link by using the search buttons at the top of the window. See ["Searching for an X12 Trade Link"](#) for additional search information.

After making your selection, the search window closes and you are returned to the X12 Outbound Trade Links window.

4. Make any desired changes to the fields in the Edit windows, and click Save when you are finished.
You are returned to the X12 Outbound Trade Links window.
5. Click any of the tab views in this window to modify any fields.
6. When you are satisfied that all of your changes have been made, click Save.
The **Trade Link Saved** message box opens to inform you that your changes were successfully saved.
7. Click OK.
8. The message box closes and the edit is complete

Copying an X12 Outbound Trade Link

About this task

The following instructions explain how to copy an X12 outbound trade link.

To copy an X12 Outbound trade link

Procedure

1. From the File menu, select **Trade Links**.
2. In the **Trade Links** navigator, select **Outbound > X12**.
3. Click **Copy Link** in the trading partner area of the window. The Copy X12 Outbound Trade Link dialog box appears.
4. Click **Find All** in the **New 'From' Trading Partner** area (or click Advanced Find for a more refined search).
5. Highlight a partner and click **Select**.
6. Click **Find All** in the **New 'To' Trading Partner** area (or click Advanced Find for a more refined search).
7. Highlight a partner and click **Select**.
8. Click **Begin Copy**.
9. Click **OK** to confirm.

The trade link copied will appear in the **X12 Outbound Trade Links** window.

Deleting an X12 Outbound Trade Link

About this task

Note: Deleting a link deletes all partner relationships, applications, and transactions associated with the trade link. The delete operation cannot be reversed.

To delete an X12 outbound trade link

Procedure

1. From the File menu, select **Trade Links**.
2. In the **Trade Links** navigator, select **Outbound > X12**.
The X12 Outbound Trade Links window opens.
3. Find the trade link to be deleted by using the search buttons at the top of the window. See "[Searching for an X12 Trade Link](#)" for additional search information.
After making your selection, the search window closes and you are returned to the X12 Outbound Trade Links window.
4. Click **Delete**.
5. When the confirmation dialog box appears, click **Yes** to delete the trade link, including all applications and transaction sets associated with it.

EDIFACT Outbound Trade Links

EDIFACT Outbound Trade Links can be added, edited, copied, and deleted.

- [Adding an EDIFACT Outbound Trade Link](#)
- [Editing an EDIFACT Outbound Trade Link](#)
- [Copying an EDIFACT Trade Link](#)
- [Deleting an EDIFACT Outbound Trade Link](#)
- [Adding a TRADACOMS Outbound Trade Link](#)
- [Editing TRADACOMS Outbound Trade Links](#)
- [Copying TRADACOMS Outbound Trade Links](#)
- [Deleting TRADACOMS Trade Links](#)
- [Adding an X12 Outbound Trade Links Application](#)
- [Editing an X12 Outbound Trade Link Application](#)
- [Deleting an X12 Outbound Trade Link Application](#)

Adding an EDIFACT Outbound Trade Link

About this task

Refer to the following procedures to add an EDIFACT Outbound Trade Link

To add an EDIFACT outbound trade link

Procedure

1. From the File menu, select Trade Links.
The **Trade Links** navigator opens.
2. Click Outbound > **EDIFACT** in the **Trade Links** navigator.
The EDIFACT Outbound Trade Links window opens.
3. Click Add Link.
The **EDIFACT Outbound Trade Links Add Wizard** opens.
4. Follow the on-screen instruction in the Wizard to add an EDIFACT Trade Link.
5. When you click Finish to complete adding an inbound trade link using the Wizard, the EDIFACT Trade Link: Add Application window opens.
6. Add applications defining the message types you receive from the external trading partner.
7. The **Version/Message Type** group box includes the four components of the EDIFACT version that you defined for the EDIFACT Standard Version.
 - Click Find Version/Message Types to select an existing EDIFACT version, or
 - Select the components from the **Version/Message Type** lists that define the version or message type.
8. Click OK.
The EDIFACT Outbound Trade Links window is displayed with the newly added trade link displayed.
9. In the Post Offices tab view, select the **Test** or **Production** check box. You can also select both.
10. Select a **Test Put Post Office** and/or a **Production Put Post Office** (depending upon which check boxes that you selected in the **Test/Production Indicators** area).
11. If validation is not desired, enable the **Route Only; no Validation** check box. See [Route only option](#) for details.
12. Click Save.
The EDIFACT Trading Partner is added.

Editing an EDIFACT Outbound Trade Link

About this task

The following instructions describe how to edit an existing EDIFACT outbound trade link.

To edit an EDIFACT inbound trade link

Procedure

1. From the File menu, select Trade Links.
2. In the **Trade Links** navigator, expand **Outbound** and select **EDIFACT**.
3. Click the **Edit From TP** button.
The Edit EDIFACT Trading Partner window opens.
4. Make changes to the fields in this window as desired.
5. Select the Syntax/Separators tab and make any desired changes.
6. Click Save.
The Edit EDIFACT window closes and you are returned to the EDIFACT Inbound Trade Links window.
7. Click the **Edit To TP** button.
The Edit EDIFACT Trading Partner in Internal folder EDIFACT window opens.
8. Make changes to the fields in this window as desired.
9. Select the Syntax/Separators tab and make any desired changes.
10. Click Save.
The Edit EDIFACT window closes and you are returned to the EDIFACT Outbound Trade Links window.
11. Click Save.
A message box opens that informs you that your updates have been made.
12. Click OK.
The EDIFACT Trade Link is updated.

Copying an EDIFACT Trade Link

About this task

The following instructions describe how to copy existing trade links.

To copy an EDIFACT outbound trade link

Procedure

1. From the File menu, select Trade **Links**.
2. In the **Trade Links** navigator, expand **Outbound > EDIFACT**.
3. Find the desired trade link by using the search buttons at the top of the window. See "[Searching for an EDIFACT Trade Link](#)" for additional search information.
Highlight a link and click Select.
4. Click the **Copy** button in the trading partner area of the window. The Copy EDIFACT Outbound Trade Link window opens.
5. Click Find All in the **New 'From' Trading Partner** area to open the EDIFACT Partner Lookup window, or click Search for a more refined search
6. Highlight a trading partner and click Select.
7. Click Find All in the **New 'To' Trading Partner** area, or click Search for a more refined search).
8. Highlight a partner and click Select.
9. Click Begin Copy.
A message box will open to inform you that your copy was successful.
10. Click OK.
The EDIFACT Trading Partner is copied.

Deleting an EDIFACT Outbound Trade Link

About this task

The following instructions describe how to delete existing trade links.

Note: Deleting a link deletes all applications and transactions associated with the trade link. The delete operation cannot be reversed.

To delete an EDIFACT outbound trade link

Procedure

1. From the File menu, select Trade **Links**.
2. In the **Trade Links** navigator, select Outbound > **EDIFACT**.
3. Find the desired trade link by using the search buttons at the top of the window. See "[Searching for an EDIFACT Trade Link](#)" for additional search information.
Highlight a link and click Select.
The selected trade link is displayed in the EDIFACT Outbound Trade Links window.
4. Click Delete.
The **Trade Link Confirmation** message box opens to ask if you really want to delete the trade link.
5. Click Yes to delete the trade link, including all applications and transaction sets associated with it.
A message box opens that informs you that the trade link was successfully deleted.
6. Click OK to close the message box.

Adding a TRADACOMS Outbound Trade Link

About this task

An inbound trade link is defined as one in which your (external) partner is sending data to you (the internal partner). You are the receiver of the incoming data. The procedure uses the **Link Wizard**.

Note: Because application partners are optional in TRADACOMS, if SDT/CDT segments are expected, select the trading partner that also has an application partner associated with it.

To add a TRADACOMS outbound trade link

Procedure

1. From the File menu, select Trade Links.
The **Trade Links** navigator opens.
2. Click Inbound > **TRADACOMS** in the **Trade Links** navigator.
3. Click Add Link.
The **TRADACOMS Outbound Trade Links Add Wizard** opens.
4. Follow the on-screen instruction in the Wizard to add a TRADACOMS Trade Link.
5. When you click Finish to complete adding an inbound trade link using the Wizard, the TRADACOMS Trade Link: Add Application window opens.
6. Select a Document Type by selecting a type from the **Document Type** field drop-down menu.
You can access a list of all TRADACOMS document types by clicking the **Find** button.
7. Click OK.
The TRADACOMS Trade Link: Add Application window closes and you are returned to the TRADACOMS Inbound Trade Links window with the Application tab displayed.
8. Select a Get Post Office using the **Get Post Office** field drop-down menu.
Note: The **Get Post Office** field is mandatory and must be entered before the Trade Link can be saved.
9. Select a Put Post Office.

If the **Test** check box is enabled, you must enter a Post Office in the **Test Put Post Office** field.

If the **Production** check box is enabled, you must enter a Post Office in the **Production Put Post Office** field.

Note: The **Get Post Office** and **Production Put Post** Office fields are mandatory depending upon which Test/Production check boxes are enabled. One of these fields must be completed before you can save the Trade Link.

10. If you want to make changes to customer or supplier details, click the SDT/CDT & VAT tab.

11. Click Save.

The **Trade Link Saved** message box opens to inform you that the trade link is saved.

12. Click OK.

The message box closes and the Trade Link is saved.

Editing TRADACOMS Outbound Trade Links

About this task

The following instructions describe how to edit existing EDIFACT inbound trade links.

To edit a TRADACOMS outbound trade link

Procedure

1. From the File menu, select **Trade Links**.
2. In the **Trade Links** navigator, expand **Outbound** and select **TRADACOMS**.
3. Find the desired trade link by using the search buttons at the top of the window. See "[Searching for a TRADACOMS Trade Link](#)" for additional search information.
Highlight a link and click Select.
After you have selected the trading partner that you want to edit, it appears in the TRADACOMS Inbound Trade Links window.
4. Click the **Edit From TP** button to edit "From" Trading Partner fields, or click the **Edit From AP** button to edit "From" Application Partner fields.
The **Edit TRADACOMS Trading Partner** window opens.
5. Make changes to the fields in this window as desired.
6. When your changes are complete, click Save.
You are returned to the TRADACOMS Inbound Trade Links window.
7. Click the **Edit To TP** button to edit "To" Trading Partner fields, or click the **Edit To AP** button to edit "To" Application Partner fields.
8. Make any desired changes to the fields in these window(s) and save your changes.
9. From the TRADACOMS Inbound Trade Links window, click the Applications tab and make any desired changes to the Get Post Office, Test Post Office, and/or Production Post Office by selecting the Edit button located to the right of each field.
10. Select the STD/CDT & VAT tab and make any desired changes.
11. When your edits are complete, click Save.
12. A message box opens that informs you that your updates have been made.
13. Click OK.
The TRADACOMS Trade Link is updated.

Copying TRADACOMS Outbound Trade Links

About this task

The following instructions describe how to copy existing trade links.

To copy a TRADACOMS outbound trade link

Procedure

1. From the File menu, select **Trade Links**.
2. In the **Trade Links** navigator, expand **Outbound > TRADACOMS**.
3. Find the desired trade link by using the search buttons at the top of the window. See "[Searching for a TRADACOMS Trade Link](#)" for additional search information.
After you select a link it is displayed in the TRADACOMS Outbound Trade Links window.
4. Click the **Copy** button in the trading partner area of the window. The Copy TRADACOMS Outbound Trade Link window opens.
5. Click Find All in the **New 'From' Trading Partner** area to open the TRADACOMS Partner Lookup window, or click Search for a more refined search.
6. Highlight a trading partner and click Select.
7. Click Find All in the **New 'To' Trading Partner** area, or click Search for a more refined search).
8. Highlight a partner and click Select.
9. Click Begin Copy.
A message box will open to inform you that your copy was successful.
10. Click OK.
The TRADACOMS Trade Link is copied.

Deleting TRADACOMS Trade Links

About this task

The following instructions describe how to delete existing trade links.

Note: Deleting a link deletes all applications and transactions associated with the trade link. The delete operation cannot be reversed.

To delete a TRADACOMS outbound trade link

Procedure

1. From the File menu, select **Trade Links**.
2. In the **Trade Links** navigator, select Inbound or **Outbound > TRADACOMS**.
3. Find the desired trade link by using the search buttons at the top of the window. See "[Searching for a TRADACOMS Trade Link](#)" for additional search information.
Highlight a link and click **Select**.
You are returned to the TRADACOMS Outbound Trade Links window with the link that you selected displayed.
4. Click **Delete**.
The **Trade Link Confirmation** message box opens to ask if you really want to delete the trade link.
5. Click **Yes** to delete the trade link, including all applications and transaction sets associated with it.
A message box opens that informs you that the trade link was successfully deleted.
6. Click **OK** to close the message box.

Adding an X12 Outbound Trade Links Application

About this task

To add a new outbound X12 trade link application

Procedure

1. From the File menu, select **Trade Links**.
2. In the **Trade Links** navigator select **Outbound > X12**.
The X12 Outbound Trade Links window opens.
3. Click the Applications tab.
The Applications tab view opens.
4. Click **Add Application**.
The X12 Trade Link: Add Application window opens.
5. Select a Put Post Office.
6. Add details about the Application by selecting the Format, GS/ST Control tab and the Transactions tab views.
7. Click **Save**.
The **Trade Link Saved** message box opens.
8. Click **OK** to close the message box.
The X12 Outbound Trade Link Application is saved.

Editing an X12 Outbound Trade Link Application

About this task

To edit an X12 outbound trade link application

Procedure

1. From the File menu, select **Trade Links**.
2. In the **Trade Links** navigator select **Outbound > X12**.
The X12 Outbound Trade Links window opens.
3. Click the Applications tab.
4. Select a trade link application from the list.
5. Make changes to any of the fields in **PO/Validation/Ack**; **Format GS**; and Transactions tab views.
6. When the information is complete, click **Save**.
The **Trade Link Saved** message box will open to tell you that your changes have been saved.
7. Click **OK** to close the message box.

Deleting an X12 Outbound Trade Link Application

About this task

To delete an X12 outbound trade link application

Procedure

1. From the File menu, choose **Trade Links**.
2. In the **Trade Links** navigator select Outbound > **X12**.
The X12 Outbound Trade Links window opens.
3. Click the Applications tab.
4. Select a trade link application from the list.
5. Click the **Delete Application** button.
The **Trade Links Application Removal Confirmation** box opens. The box tells you that you must **Save** after closing the box to complete removal.
6. Click OK.
The message box closes.
7. Click Save.
The trade link application is deleted.

EDIFACT Outbound Trade Links Applications

You can add, edit, and delete EDIFACT Outbound Trade Link Applications.

- [Adding an EDIFACT Outbound Trade Links Application](#)
- [Editing an EDIFACT Outbound Trade Link Application](#)
- [Deleting an EDIFACT Outbound Trade Link Application](#)
- [TRADACOMS Outbound Trade Link Applications](#)

You can add, edit, and delete TRADACOMS Trade Link Applications.

Adding an EDIFACT Outbound Trade Links Application

About this task

Use the following instructions to add an EDIFACT Outbound Trade Links Application.

To add an EDIFACT outbound trade links application

From the File menu, select **Trade Links**.

Procedure

1. In the **Trade Links** navigator select Outbound > **EDIFACT**.
The EDIFACT Outbound Trade Links window opens.
2. Click the Applications tab.
3. Click the **Add Application** button.
The EDIFACT Trade Link: Add Application window opens. If the No UNG option is set, the two fields in the **Application Partners** area will be grayed out.
4. If this is an optional or mandatory UNG link, and an UNG segment is expected for this application, select the Application Partners associated with this UNG segment in the drop-down list.
5. Complete the **Msg Version No**, **Msg Release No**, and **Controlling Agency** fields.
6. Click the **Find Version Message Types** button to open the **Find All EDIFACT Functional ID Message Types** list box and select a message type from the list.
7. Click OK.
8. With the Post Offices tab displayed, select test or production data by enabling one or both **Test** and/or **Production** check boxes.
9. Select a Test Post Office and/or Production Post Office. You must select a test or production Post Office in order to save the trade link application.
10. Click the **Msg Control No/Version Info** and Application Partners tab and make any changes to any fields in these views.
The Acknowledgements are set in the **Ack Level** field. See [Adding an EDIFACT Inbound Trade Link Application](#).
11. Click Save.
The new Outbound Trade Link Application is now created.

Editing an EDIFACT Outbound Trade Link Application

About this task

To edit an inbound EDIFACT outbound trade link application

Procedure

1. From the File menu, select **Trade Links**.
2. In the **Trade Links** navigator select Outbound > **EDIFACT**.
The EDIFACT Outbound Trade Links window opens.
3. Click the Applications tab.
4. Select a trade link application from the list.
5. Make changes to any of the fields in **Msg Control No/Version Info**, **Post Offices** and Application Partners tab views.
6. When your updates are complete click **Save**.
The **Trade Link Saved** message box will open to tell you that your changes have been saved.
7. Click OK to close the message box.

Deleting an EDIFACT Outbound Trade Link Application

About this task

To delete an EDIFACT outbound trade link application

Procedure

1. From the File menu, choose **Trade Links**.
2. In the **Trade Links** navigator select Inbound > **EDIFACT**.
The EDIFACT Inbound Trade Links window opens.
3. Click the Applications tab.
4. Select a trade link application from the list.
5. Click the **Delete Application** button.
The **Trade Links Application Removal Confirmation** box opens. The box tells you that you must **Save** after closing the box to complete removal.
6. Click OK.
The message box closes.
7. Click **Save**.
The trade link application is deleted.

TRADACOMS Outbound Trade Link Applications

You can add, edit, and delete TRADACOMS Trade Link Applications.

- [Adding a TRADACOMS Outbound Trade Link Application](#)
- [Editing a TRADACOMS Outbound Trade Link Application](#)
- [Deleting a TRADACOMS Outbound Trade Link Application](#)

Adding a TRADACOMS Outbound Trade Link Application

About this task

To add a new TRADACOMS outbound trade link application

Procedure

1. From the File menu, select **Trade Links**.
2. In the **Trade Links** navigator select Outbound > **TRADACOMS**.
The TRADACOMS Outbound Trade Links window opens.
3. Click the Application tab.
4. Click the **Add Application** button.
The TRADACOMS Trade Link: Add Application window opens.
5. Select a Document Type.
6. Click OK.
7. Add data as desired in the fields in the SDT/CDT Control No VAT RSG tab view.
8. With the Post Offices tab displayed, select test or production data by enabling one or both **Test** and/or **Production** check boxes.
9. Select a **Test Post Office** and/or **Production Post Office**.
10. Click **Save**.

Editing a TRADACOMS Outbound Trade Link Application

About this task

To edit an outbound TRADACOMS trade link application

Procedure

1. From the File menu, select **Trade Links**.
2. In the Trade Links navigator select Outbound > **TRADACOMS**.
The **TRADACOMS Outbound Trade Links** window opens
3. Click the Applications tab.
4. Select a trade link application from the list.
5. Make changes to any of the fields in **Post Offices** and SDT/CDT Control No VAT RSG tab views.
6. When the information is complete, click **Save**.
7. The **Trade Link Saved** message box will open to tell you that your changes have been saved.
8. Click **OK** to close the message box.

Deleting a TRADACOMS Outbound Trade Link Application

About this task

To delete a TRADACOMS outbound trade link application

Procedure

1. From the File menu, choose **Trade Links**.
2. In the **Trade Links** navigator select Outbound > **TRADACOMS**.
The TRADACOMS Outbound Trade Links window opens.
3. Click the Applications tab.
4. Select a trade link application from the list.
5. Click the **Delete Application** button.
The **Trade Links Application Removal Confirmation** box opens. The box tells you that you must **Save** after closing the box to complete removal.
6. Click **OK**.
The message box closes.
7. Click **Save**.
The TRADACOMS Outbound Trade Link Application is deleted.

ISA and GS Numbering Schemes

The functional group control number reset does not occur until after the next refresh (the next time the Message Manager's **Export** map is executed). See the Trading Manager documentation for information on scheduling the execution of the **Export** map.

The following list defines the ISA and GS Numbering Schemes.

Increment

This numbering scheme is the default. A perpetually-incrementing interchange control number is used.

The first interchange sent for this relationship has a control number of 1, the second has a control number of 2, and so on.

If you select Increment, you can also reset the number to your preference by enabling **Reset** and specifying a control number.

User Defined

Use a user-defined numbering scheme. Enter the full name and path to a compiled map file (.mmc) that provides this custom control number.

The compiled map file is required when you select this option. The full file name and path of the compiled map file must be specified relative to the path of the Launcher.

You can also reset the number to your preference by enabling **Reset** and specifying a control number.

Increment by Trading Partner

A perpetually incrementing GS number within this partner link (internal application partner to external trading partner) is used.

The first interchange sent for this relationship has a control number of 1 the second has a control number of 2, and so on.

You can also reset the number to your preference by enabling **Reset** and specifying a control number.

Increment by Application Partner

A perpetually incrementing functional group control number within this partner link (application partner and application partner) is used.

The first functional group sent for this application partner within this trading partner has a control number of 1, the second has a control number of 2, and so on.

You can also reset the number to your preference by enabling **Reset** and specifying a control number.

Increment by Functional ID Code

A perpetually incrementing functional group control number within the functional ID code, regardless of trading partner is used.

The first PO functional group sent has a control number of 1; the second has a control number of 2 (regardless of its partner link), and so on.

Relative

This selection is the default.

A functional group control number that is relative to the functional group's sequence within its interchange.

The first functional group in the first interchange has a control number of 1, the second has a control number of 2, the first functional group in the second interchange has a control number of 1, and so on.

Searching for a trade link

When entering search criteria, case sensitivity for the search depends on the database. By default, Microsoft Access and Microsoft SQL Server are not case sensitive (for example, Acme is the same as ACME), while Oracle is case sensitive. For more information, contact your database administrator to determine if SQL LIKE searches are case sensitive.

- [Searching for an X12 Trade Link](#)
 - [Searching for an EDIFACT Trade Link](#)
 - [Searching for a TRADACOMS Trade Link](#)
-

Searching for an X12 Trade Link

About this task

Refer to the following steps to search for an X12 Trade Link.

To search all X12 trade links

Procedure

1. Click the File menu and select Trade Links.
The **Trade Links** navigator opens.
2. Select Inbound or **Outbound > X12**.
The **Inbound** or Outbound X12 Trade Links window opens. The Search buttons are located at the top of the window. These buttons allow you to search for Trade Links based upon various search criteria.
3. To search all available trade links, click the **All Links** button.
The X12 Inbound Trade Link Lookup: All Links window opens.
4. Sort the names in the list box by selecting one of two **Display** buttons.
 - Select Names to sort the list by Trading Partner name. This causes names to be sorted in alphanumeric order.
 - Select ISA/GS to sort in ISA order:
5. Select the desired Trade Link from the list and click Select.

To search X12 trade links by partner

Procedure

1. Click the File menu and select Trade Links.
The **Trade Links** navigator opens.
2. Select Inbound or **Outbound > X12**.
3. Click the **Search by Partner** button at the top of the window.
The X12 Inbound Trade Link Lookup: Search By Partner window opens.
4. Define your search criteria by enabling or disabling (checking or unchecking) the check boxes in the Search the following window area and by selecting and deselecting a button in the ISA/GS Search (Exact) window area.
5. When you are satisfied that your search criteria are properly defined, click the **Begin Search** button to populate the list box with trade links that match your search criteria
6. Select the desired trade link from the list box and click Select.

To select an X12 trade link by document type

Procedure

1. Click the File menu and select Trade Links.
2. Select Inbound or **Outbound > X12**.
3. Click the **Search by Document** button at the top of the window.
The X12 Inbound Trade Link Document Search window opens.
4. Select a document from the top list box. When you click on the document, the trade links associated with that document display in the bottom list box.
5. Click on the desired trade link then click Select.

Searching for an EDIFACT Trade Link

About this task

Refer to the following steps to search for an EDIFACT Trade Link.

To search all EDIFACT trade links

Procedure

1. Click the File menu and select Trade Links.
2. Select Inbound or **Outbound > EDIFACT**
3. Click the **Search All** button at the top of the window.
The EDIFACT Inbound or Outbound Trade Link Lookup: All Files window opens.
4. Scroll through the files in the list until you locate the desired file.
5. Click Select.

To select EDIFACT trade links by partner

Procedure

1. From the File menu select Trade Links.
2. Select Inbound or **Outbound > EDIFACT**
3. Click the **Search By Partner** button at the top of the window.
The EDIFACT Inbound or Outbound Trade Link Lookup: Search By Partner window opens.
4. Define your search criteria by enabling or disabling (checking or unchecking) the check boxes in the Search the following window area and by selecting and deselecting a button in the Interchange ID Search (Exact) window area.
 - Search for a specific Folder or Trading Partner by entering specific data into the **Search For** field.
 - Search for a specific Interchange ID by entering specific data into the **Interchange ID** field.
5. When you are satisfied that your search criteria are properly defined, click the **Begin Search** button to populate the list box with trade links that match your search criteria
6. Select the desired trade link from the list box and click Select

To select an EDIFACT trade link by document

Procedure

1. From the File menu select Trade Links.
2. Select Inbound or **Outbound > EDIFACT**
3. Click the **Search by Document** button at the top of the window.
The EDIFACT Inbound or Outbound Trade Link Document Search window opens.
4. Select a document from the top list box. When you click on the document, the trade links associated with that document display in the bottom list box.
5. Click on the desired trade link then click Select.

Searching for a TRADACOMS Trade Link

About this task

Refer to the following steps to search for an EDIFACT Trade Link.

To search all TRADACOMS trade links

Procedure

1. Click the File menu and select Trade Links.
2. Select Inbound or **Outbound > TRADACOMS**
3. Click the **Search All** button at the top of the window.
4. The **TRADACOMS Inbound** or Outbound Trade Link Lookup: All Files window opens.
5. Sort the files in the list by selecting one of two buttons in the Display area at the top of the window.
 - Select the **Names** button to sort the list by Trading Partner name.
 - Select **ID** to sort by Trading Partner ID.
6. Click Select.

To select TRADACOMS trade links by partner

Procedure

1. From the File menu select Trade Links.

2. Select Inbound or **Outbound > TRADACOMS**
3. Click the **Search By Partner** button at the top of the window.
The **TRADACOMS Inbound** or Outbound Trade Link Lookup: Search by Partner window opens.
4. Define your search criteria by entering starting letters for a trading partner, application partner or from application partner ID in the fields in the Enter Starting Letters window area. You can also select whether or not you want to sort using names, or IDs by selecting one of the two **Display** buttons.
5. When you are satisfied that your search criteria are properly defined, click the **Begin Search** button to populate the list box with trade links that match your search criteria
6. Select the desired trade link from the list box and click Select.

To select a TRADACOMS trade link by document

Procedure

1. From the File menu select Trade Links.
2. Select Inbound or **Outbound > TRADACOMS**
3. Click the **Search by Document** button at the top of the window.
The **TRADACOMS Inbound** or Outbound Trade Link Document Search window opens.
4. Select a document from the top list box. When you click on the document, the trade links associated with that document display in the bottom list box.
5. Click on the desired trade link then click Select.

Automatic acknowledgement creation

Partner Manager will display the **Automatic Acknowledgement Creation** dialog if the trade link for that acknowledgement does not already exist and an acknowledgement is expected from your external trading partner, or you are creating an acknowledgement.

This may occur when you:

- add a trade link
- copy a trade link
- add an application to an existing trade link

This dialog will create or modify the trade link for the opposite direction so that you do not have to manually make those changes. The dialog may display multiple times if there are more than version/functional group on the trade link.

The instructions presented at the top of the dialog will show the version and transaction set that will be created. Follow the instructions carefully as they will change when multiple versions are being processed.

Outbound CONTRL trade link

Please note that CONTRL, as used in this section, is *not* a typographical error.

Partner Manager displays the Automatic Outbound CONTRL Creation dialog box when you:

- Add a trade link
 - Copy a trade link
 - Add an application to an existing trade link
 - The trading partner link requires an outbound CONTRL acknowledgement (an outbound acknowledgement does not already exist for that trading/application partner and version)
- The Automatic Outbound CONTRL Creation dialog box appears only when you are creating EDIFACT trade links.

If you select No Acknowledgement in the **Ack Level** drop-down list, this dialog box does not appear.

If you want to create an outbound CONTRL acknowledgement, select the **Put and Get Post Offices** from the drop-down list and click Create. This dialog box may appear more than once if the link contains multiple versions and/or acknowledgement partners. In this case, the version appears in the dialog box.

The Control Message Type will default to : Version=4, Release=1, Controlling Agency=UN; the sole CONTRL Message Type provided by IBM in the EDIFACT Standards (see Utilities > Standards). This section of the form will remain disabled until you create additional CONTRL Message Types in the EDIFACT Standards definitions. At such a time, this section will be enabled and you can select which CONTRL Message Type to use.

The UNG Options (No UNG vs. Mandatory UNG) is disabled if an Outbound Trade Link already exists for this CONTRL creation, or if the Inbound Trade Link does not contain an UNG-level definition. If an Outbound Trade Link already exists, the UNG Option of that link will be used for the CONTRL document. Note Optional UNG CONTRL Trade Links are not supported.

Note: The Automatic CONTRL form will not be displayed if an Outbound Trade Link already exists, and contains a CONTRL Message Type.

Creating a control number map

Trading Manager supports user-defined numbering schemes in maps for interchange, functional group, and transaction set control numbers. This section describes the requirements for your maps that provide a number and are executed from the Message Manager using the RUN function.

- [Exchanging information with Message Manager](#)

You implement user-defined control numbering schemes through control number maps. When building envelopes for those application partners that provide

- partially wrapped X12 data, the compiled map providing the control number level and specified in the Partner Manager is executed by the `RUN` function.
- [Map requirements and features](#)

Exchanging information with Message Manager

You implement user-defined control numbering schemes through control number maps. When building envelopes for those application partners that provide partially wrapped X12 data, the compiled map providing the control number level and specified in the Partner Manager is executed by the `RUN` function.

The Message Manager passes input card 1 to the control number map the last used control number for this particular trading partner, application partner, or transaction set. The control number map returns the control number value to be used as output card 1 to the Message Manager; this value is placed in the appropriate EDI envelope.

This control number returned by the control number map is used to update the last used control number within the Message Manager for the appropriate trading partner, application partner, and transaction set.

Map requirements and features

The control number map must have at least one input card and one output card. You must define input card 1 and output card 1 to accommodate a control number character size of 1-9 characters. For example, define a text or number item with a minimum length of 1 and a maximum length of 9.

Alternately, you could define a control number Group in a type tree with Items that recognize component parts of the one- to nine-character control number. Include an Item where the control number is defined as `YYMMDD`.

Note: Although the string that is transferred to and from the control number map can be defined as text, the interchange and functional group control numbers must be numeric.

- [Example: increments other than 1](#)
- [Example of transaction set control number by date](#)

Example: increments other than 1

The map may include other input cards and output cards as needed. For example, you may have established an increment other than one for the ISA control number that you exchange with your partner:

- Input card 1 represents the most recently used control number (`LastUsedControl Number`)
- Input card 2 represents data from a Customer Master Database (`CustomerMasterDB`)
- Output card 1 contains a rule with the `SEARCHUP` function to find the increment for the ABC Rock customer; the increment is added to the most recently used control number and the result is returned to the Message Manager. The map rule is:

```
=LastUsedControlNumber + SEARCHUP ( Increment Column:::CustomerMasterDB ,  
CustID Column:::CustomerMasterDB , "ABC Rock" )
```

Example of transaction set control number by date

For example, a map may concatenate a sequence number with other information such as the current date:

- Input card 1 represents the most recently used control number (`LastUsedControl Number`)
- Output card 1 contains a rule with the `CURRENTDATE` function. In this example, the first six characters of the control number are the current date in YYMMDD format. The last three characters represent the sequence number of the current data set. If the date from the most recently used control number is the same date as the current date, the sequence number is incremented by one. If the date is not the same as the current date, the sequence number is reset to 1. The map rule is:

```
=IF Shift 9 =( RENTDATE() = YYMMDD:LastUsedControlNumber ,  
SeqNum:LastUsedControlNumber +1 , 1 )
```

Reports

About this task

Use the Reports window to view Message Manager transmission and audit report information. The Reports window enables you to view and generate detailed information about transmissions and transactions.

To access the Reports window

Procedure

Click on Reports the toolbar. Or, select Reports from the File menu.
The Reports window displays.

Results

The following four types of reports are available from the **Report Type** drop-down box:

- ["Traffic Reports"](#)
- ["Alerts Report"](#)
- ["Additional Reports Overview"](#)
- ["Partner Manager Log Report"](#)

You can run Partner Manager reports on the current or archive database. After reports are run, they can be printed or exported.

Reports are presented in views, which provide different levels of detail and enable resend interchange capability. Each report has a default view.

Note: If you do not have any traffic for a data type, that data type will not be displayed in the data type's drop-down box. For example, if you do not have X12 traffic, **X12** will not be listed in the data type's drop-down box.

- [Traffic reports](#)
 - [Additional reports overview](#)
 - [Printing Partner Manager reports](#)
 - [Exporting reports](#)
 - [Value Added Tax \(VAT\) reports](#)
 - [Partner Manager editor](#)
-

Traffic reports

Traffic reports contain information on transmissions that have been processed by the Message Manager system in the specified date range.

- Traffic reports can be refreshed at any time.
- The default date selection for traffic reports is configured in the Display Options utility.
- Traffic reports can be run on inbound and outbound directions on a selected report level.
- Traffic reports can be exported to file formats for viewing and printing.
- Traffic reports allow you to edit and resend interchanges.
- Traffic Reports allow you to optionally view X12 997 content, and to jump between X12 997 transmissions and the corresponding Functional Group.
- Traffic Reports allow you to review your X12 Acknowledgement activity. For example, displaying Overdue Acknowledgements, All Unacknowledged, Acknowledged Only, or manually Acknowledged traffic.

Navigate in traffic reports with right-click command access or by clicking the buttons on the toolbar.

- [Drill down for details](#)
 - [Report specific search](#)
 - [Specify a date range](#)
 - [Report status icon colors and codes](#)
 - [Partner Manager log report](#)
 - [Alerts report](#)
 - [Resend requests](#)
 - [Transmission report](#)
 - [Interchange report](#)
 - [Functional group report](#)
 - [Message/transaction report](#)
 - [X12 TA1 status reports](#)
-

Drill down for details

You can "drill down" to view finer levels of detail from the default report view in traffic reports. The "drill down" options include:

| View Type | Drill Down To |
|------------------|---|
| All | Display Detail |
| Transmission | Display Data Type |
| Data Type | View Interchanges |
| Interchange | View Functional Groups
Resend the interchange immediately
Edit/Resend the interchange |
| Functional Group | Display Transactions |
| Transaction | Display Segment Errors |

Icon colors in the reports provide visual status indicators for each transmission.

For EDIFACT reporting, the drill down may skip functional group views if there are no functional groups in the data.

- [View Interchanges in Reports](#)
- [View Functional Groups in Reports](#)
- [Display detail in reports](#)
- [Display Transactions in Reports](#)
- [Display Segment Errors in Reports](#)
- [Display Data Types](#)

View Interchanges in Reports

About this task

In a data type report view, you can view interchanges for the selected transmission.

To view interchanges

Procedure

1. In a report, select a single transmission.
2. Click the **Interchanges** button.
The **Interchanges View** appears.

Results

From the **Interchanges View**, you can view Functional Groups, resend that interchange immediately, and edit/resend the interchange. If X12 TA1 records exist, a drill-down from **Interchange** to **Functional Groups** will also display those X12 TA1 records in a separate window.

View Functional Groups in Reports

About this task

In an interchange report view, you can view functional groups for the selected interchange.

To view functional groups of a specific interchange

Procedure

1. In a report, select a single functional group.
2. Click the **Functional Groups** button.
The Functional Groups View appears.

Display detail in reports

For each report view, the **Display Detail** screen provides detailed information on the selected interchange, transmission, functional group, transaction, alert, or log.

Display Transactions in Reports

About this task

In a functional group view report, you can view transaction details.

To view transaction details

Procedure

1. Select a single transmission.
2. Click the **Display Transactions** button.
The **Transactions View** of that transaction is displayed.

Display Segment Errors in Reports

About this task

For each transmission in a transactions view report, you can display segment errors if they exist.

To view segment errors of a transmission

Procedure

1. Select a single transmission in a transaction view of a report.

2. If segment errors exist for the selected transmission, click the **Display Segment Errors** button.

Note: The **Display Segment Errors** button is available only if the selected transmission contains segment errors. If no segment errors exist, the button is not available.

Display Data Types

About this task

In a report **Transmission View**, you can view data type details in the **Interchanges View**.

To view data type details

Procedure

1. Select a single transmission.
2. Click the **Display Data Type View** button.

Results

Double-click a transmission in the **Data Type View** to display functional group details, and then transaction details.

Report specific search

The **Report Specific Search** field is specific to each report level.

Report levels in traffic reports include:

| | |
|--------------------|-----------------------|
| • Transmission | • Message/Transaction |
| • Interchange | • Traffic Alert |
| • Functional Group | • Partner Manager Log |

Specify a date range

For traffic reports, the **Date Range** can be adjusted as required. Click the option button to choose your date range.

| Option | Header |
|------------|---|
| All dates | All dates are reported |
| Date Range | When enabled, the From Date and To Date fields can be populated. Click Calendar to choose the From Date and To Date. For Traffic Alerts, the From Time and To Time can be entered in HHMM format. |
| Today | Data with current (today's) date reported. |

Report status icon colors and codes

The colors of the icons in the Partner Manager traffic reports provide visual indicators of status. The status description, **Red**, **Yellow**, **Green** or **Blue** can also be included in the exported traffic reports by enabling the **Include Traffic Status Color** check box).

| Color | Description |
|---------------------------|---|
| Red | Indicates an error in transmission at that level |
| Red (TA1 reports only) | Rejected |
| Yellow | Error at a level below displayed transmission level |
| Green | Transmission completed without errors |
| Green/Yellow | X12 Functional Acknowledgement is in error |
| Green (TA1 reports only) | Accepted |
| Yellow (TA1 reports only) | Partially accepted |
| Blue | In process |

The following table describes each code and its icon color.

| Code | Description | Icon Color |
|------|--|------------|
| 2 | Syntax version or level not supported | Red |
| 7 | Interchange recipient not actual recipient | Red |
| 12 | Invalid value | Red |
| 13 | Missing | Red |
| 14 | Value not supported in this position | Red |

| Code | Description | Icon Color |
|-------------|---|-------------------|
| 15 | Not supported in this position | Red |
| 16 | Too many constituents | Red |
| 17 | No agreement | Red |
| 18 | Unspecified error | Red |
| 20 | Character invalid as service indicator | Red |
| 21 | Service character(s) | Red |
| 22 | Invalid service character(s) | Red |
| 23 | Unknown Interchange sender | Red |
| 24 | Too old | Red |
| 25 | Test indicator not supported | Red |
| 26 | Duplicate detected | Red |
| 28 | References do not match | Red |
| 29 | Control count does not match number of instances received | Red |
| 30 | Groups and messages/packages mixed | Red |
| 32 | Lower level empty | Red |
| 33 | Invalid occurrence outside message, package, or group | Red |
| 35 | Too many data element or segment repetitions | Red |
| 36 | Too many segment group repetitions | Red |
| 37 | Invalid type of character(s) | Red |
| 39 | Data element too long | Red |
| 40 | Data element too short | Red |
| 45 | Trailing separator | Red |
| 46 | Character set not supported | Red |
| 47 | Envelope functionality not supported | Red |
| 48 | Dependency condition violated | Red |
| D01 | Mandatory element missing | Red |
| D02 | Data element has incorrect size | Red |
| D03 | Invalid code for identifier | Red |
| D04 | Invalid date format | Red |
| D05 | Invalid time format | Red |
| E00 | Unknown data element | Red |
| E01 | Code list error | Red |
| E02 | Data element has incorrect format | Red |
| E03 | Data element has incorrect size | Red |
| E04 | Missing segment | Red |
| E05 | Missing segment terminator | Red |
| E06 | Mandatory element missing | Red |
| E07 | Error in data | Red |
| E08 | Unknown element found | Red |
| E09 | Failed business rule | Red |
| E10 | Missing element | Red |
| E11 | ISA/IEA format or content error | Red |
| E12 | GS/GE format or content error | Red |
| E13 | Invalid because of errors elsewhere | Red |
| E14 | Missing tradelink | Red |
| E15 | Unknown transaction/Version release | Red |
| E16 | Failed to route to PUT post office | Red |
| E17 | Valid but not delivered because of errors elsewhere | Red |
| E18 | Functional group does not contain any transaction | Red |
| E19 | Contains transaction(s) that do not belong in this functional group | Red |
| E20 | Duplicate transaction | Red |
| E21 | Duplicate functional group | Red |
| E22 | UNB/UNZ format content error | Red |
| E23 | UNH/UNT format content error | Red |
| F01 | FA in error | Green/ Yellow |
| F02 | Rejected via TA1 | Green/ Yellow |
| S01 | Segment out of sequence or unknown | Red |
| S02 | Missing mandatory segment/loop | Red |
| S03 | Loop exceeded limit | Red |
| S04 | Segment exceeded limit | Red |
| S05 | Segment has element error | Red |
| V00 | Data valid | Green |
| V01 | Error data fount | Yellow |
| V02 | Contains functional group with unknown tradelink | Yellow |
| V03 | Duplicate interchange | Red |
| V04 | Failed to deliver some or all functional groups (inbound) | Yellow |
| V05 | In process | Blue |

| Code | Description | Icon Color |
|------|---|------------|
| V06 | Failed to deliver some or all interchanges (outbound) | Yellow |
| V07 | Accepted with errors via TA1 | Red |
| V08 | Failed Interchange Authorization Check | Red |
| V09 | Failed Interchange Security Check | Red |

Partner Manager log report

Access to the Partner Manager Log report is granted or denied in the **Security Groups** utility. The Partner Manager Log report identifies error messages and reports Resend requests/processing. In addition, if Partner Manager security is enabled and then later disabled, a warning will appear in the log. This report is run for a specified date range. You can export the report. For information on how to purge the log, see the "Purge" section in Chapter 8.

Click an options button in the **Reports Options** to dynamically populate the report for **All Messages, Errors Only, Warnings Only, and Information Only**.

Information in the Partner Manager Log report includes:

| Field | Description |
|-----------|--|
| Source | Source of the log message. |
| User | The current User ID if Security Groups are being used. If Partner Manager security is not used, User defaults to the database user name that user or the current user. |
| Date Time | The date and time of the log event in the YYYYMMDD HH:MM:SS format. |
| Message | Log message that describes the log event. |

Note: The Partner Manager Log report replaces the Resend Requests report.

- [Source of Partner Manager log message](#)

Source of Partner Manager log message

The source of the log message includes incident type and status level.

Possible log message sources include:

| Log Message Type | Description | Status Type | Icon |
|------------------|---|-------------|--------|
| Program Error | Run-time errors generated by the Partner Manager application. | Error | Red |
| SQL Errors | Errors generated by the database system you are using. | Error | Red |
| User Maintenance | In the event security is disabled (by deleting the last user). | Warning | Yellow |
| Login | In the event the user with the password overwrote the database-version check. | Warning | Yellow |
| Resend | When a resend request is made. Also when the resend item is processed by Message Manager. | Information | Green |

Program and SQL errors are useful for communicating with IBM Software Customer Support if problems arise.

Alerts report

The Alert report shows a history of e-mail traffic alert notifications of rejected data for the specified date range and database (current or archive). The dates specified are for the Message Manager map execution dates (when traffic occurs in Partner Manager).

You can specify reporting on traffic alerts that match the following selection requirements:

- Specified Date Range
- Specified Database (Current or Archive)
All traffic in the Message Manager system is reported, regardless of data type.

Information in the Traffic Alert report includes:

| Field | Description |
|-----------------|---|
| PO Nickname | Post office nickname as defined in Partner Manager. |
| Transmission ID | Unique Transmission ID assigned by Message Manager to the interchange. |
| Data Type | EDIFACT, X12, or TRADACOMS. |
| Contact | The e-mail address specified in the Address Book for the trading partner. |
| Start Time | Time when the traffic alert was sent. |

Resend requests

The **Resend Requests** report has been replaced with the Partner Manager Log Report, which logs resend requests.

Transmission report

The **Transmission** report is a standard-independent report that displays a summary of inbound or outbound transmissions.

Note: All transmissions are reported (EDIFACT, TRADACOMS, X12, and any other standards).
You can specify reporting on transmissions that match the specified selection requirements.

Information is displayed for the specified transmission direction (inbound or outbound). The information displayed includes:

| Field | Description |
|-----------------------|---|
| Transmission ID | A component of the ThreadID assigned by Message Manager system to the transmission. |
| Date | Date when the Message Manager transmission processing began. |
| Time | Time when the Message Manager transmission processing began. |
| Size | Size (in bytes) of the transmission data. |
| Return Code | Report status code. |
| O/R | Original or Resend indicator. An O indicates an original transmission. An R indicates a resent transmission. |
| Input Thread | Source of the transmission data. If the data source is a file, this column shows the file name. Otherwise, it shows the resource adapter (FTP, E-mail, VAN) command string. |
| Original Transmission | If this is a resent transmission, the Original Transmission ID is displayed. Blank if this is not a resent transmission. |
| PO Nickname | Post Office associated with the transmission. |

Note: Double-click a transmission, or select a single transmission and click the **Data Type View** button, to the display the **Data Type View**.

Interchange report

The Interchange report displays trading partner interchange information for the specified interchange date range.

The **Inbound or Outbound Interchanges View** displays information for interchanges and allows you to:

- Resend that interchange immediately
- Edit/Resend the interchange

You can specify reporting on interchanges that match the following selection requirements:

- Data Type** (select EDIFACT, TRADACOMS, or X12, or other standard from the drop-down list)
- Specific Post Office Nickname
- Specified Date Range
- Direction (Inbound or Outbound)
- Database (Current or Archive)
- Record **Status of All Records, No Errors**, records with **Errors Only**, or a Specific Status Code
- A specific Trading Partner
- A Report Specific Search (enter a Control Number)

The information displayed includes:

| Field | Description |
|-----------------------|--|
| Transmission ID | Unique Transmission ID assigned by Message Manager to the interchange. |
| Sender Ichg ID/ Qual | ISA Sender ID qualifier and address of the trading partner that sent the interchange. |
| Sender Name | Trading partner name as defined in the Address Book . |
| Receiver Ichg ID/Qual | ISA Receiver ID qualifier and address of the trading partner that received the interchange. |
| IC Ctrl No. | ISA Control number of the interchange. |
| IC Date | YYYYMMDD format of the date the interchange was created by the sender. (UNB/ISA Envelope) |
| IC Time | HHMM format of the time the interchange was created by the sender. (UNB/ISA Envelope) |

- [Drill down tips](#)
- [Specific trading partner for the interchange report](#)
- [Specific status codes for the interchange report](#)
- [Edit and/or resend an interchange](#)
- [Resend an interchange Immediately](#)

Drill down tips

About this task

- Double-click a transmission in the interchange view-to-view functional group information.
- Double-click a transmission in the functional group view to display transaction information.
- Double-click a transmission in the transaction view to display segment errors (if any).

Specific trading partner for the interchange report

About this task

In the Interchange report, you can specify a specific trading partner.

To specify a specific trading partner

Procedure

1. In the **Trading Partner** area, enable the **Sending** or **Receiving** option button.
2. Click the **Find** button.
The specified data type Trading Partners dialog box appears.
3. Select a trading partner.
4. Click **Select**.
The selected trading partner appears in the **Name** field.

Specific status codes for the interchange report

About this task

In the Interchange report, you can specify a specific status code.

Procedure

1. To specify a specific status code
2. For the **Specific Status Code** field, enter the status code.
Or click the **Status Codes** button and select a status code from the **Find All Status Codes** dialog box. Click **Select**.
3. Click the **Run Report** button to run the Interchange report for the specific status code.

Edit and/or resend an interchange

About this task

In the Interchange report, you can select a single interchange and edit and resend that interchange immediately.

Note: The Display Options,>Editor/Viewer setting controls which text editor is used to edit the interchange.

To edit and resend an interchange

Procedure

1. Select a single interchange in the Interchange report.
2. Click the **Edit/Resend Interchange** button in the toolbar.
The Partner Manager Editor window appears.
3. Edit the interchange as desired.
4. If using a text editor other than the Partner Manager Editor:
 - Save the file and close that editor.
 - Click Revert on the Partner Manager Editor toolbar to view the interchange in the Partner Manager Editor window.
5. Click Resend on the Partner Manager Editor toolbar.
A confirmation dialog box appears.
6. Click OK to resend the data as displayed.
A resend status confirmation dialog box appears. The location of the resend data and the interchange file name is displayed.

Results

The resend, user, date and time stamp, and message appear in the Partner Manager Log report.

Note: The Security Group Maintenance,>Transmission Data setting controls edit and resend capability.

Resend an interchange Immediately

About this task

In the Interchange report, you can select a single interchange and resend that interchange immediately.

To resend an interchange immediately

Procedure

1. Select a single interchange.
2. Click Resend Interchange Immediately button on the toolbar.
A resend confirmation dialog box appears.
3. Click OK to send the interchange.
A sent confirmation dialog box appears.
4. Click OK.

Results

The resend, user, date and time stamp, and message appear in the Partner Manager Log report.

Note: The Security Group Maintenance > Transmission Data setting controls edit and resend capability.

Functional group report

The **Functional Group** report displays a summary of functional groups received (**Direction = Inbound**) or sent (**Direction = Outbound**) for sending or receiving trading partners.

For a complete description of all fields in the report display, see the section "[Functional Group View](#)".

- [Jumping to X12 997](#)
- [X12 acknowledgement reporting options](#)
- [Manual override of acknowledgement settings](#)
- [Functional group view](#)

Jumping to X12 997

About this task

The **X12 Functional Group** Report allows you to jump directly to the corresponding 997 from a functional group.

To jump to a X12 997

Procedure

1. Select a functional group.
2. Click the **Jump to Corresponding 997 group** icon in the toolbar.
The display "jumps" to the corresponding 997 from the functional group.
3. Use the report's **Back** button to return to the original report.
Note: Two conditions must exist for this function to be available. The selected functional group must have had its Functional Acknowledgment Received (that is, the FG Reconciliation has processed this record), and if user security is in force, the user's Security Group setting for "X12 997 Data: View Data" must be checked.

X12 acknowledgement reporting options

When an **Outbound X12 Functional Group** report is selected in the Reports dialog box, five options become available in the **Report Specific Search** area on the **Acknowledgement Reporting** drop-down menu:

| Option | Description |
|--------------------------|---|
| All Records | The default; displays all data regardless of Acknowledgement Status. |
| Overdue Acknowledgements | Only functional groups that have not been acknowledged within the time frame specified on the trade link will be included. See "Specifying Acknowledgement Timeframe". This time calculation is based on the number of hours specified on the trade link, the map execution date and time, and current date and time of the PC running Partner Manager. |
| All Unacknowledged | Only Unacknowledged Functions Groups will be included. |
| Acknowledged Only | Opposite of "All Unacknowledged"; only those functional groups that have been acknowledged will be included. |
| Manual Acknowledged Only | Only those functional groups that have been manually acknowledged will be included. See Manual Override of Acknowledgement Settings for additional information. |

Note: Functional groups that have a Functional Acknowledgement in Error will display a green and yellow arrow pointing right in this report. This arrow and unique color is used to distinguish between a good functional group (green circle), a Functional Group containing an error at a lower level (yellow diamond), and a Bad Functional Group (red stop sign).

Manual override of acknowledgement settings

About this task

The **X12 Outbound Functional Group** report allows you to manually set the acknowledgement flag. Normally, the maps set this acknowledgement flag automatically during functional group reconciliation when the 997 is received. However, there may be times where you want to set this flag yourself, perhaps after manual intervention with your trading partner.

To manually set the Acknowledgement flag

Procedure

1. Select the functional group record you wish to change in this report.

2. Click the **Manual Acknowledgement Override** icon in the toolbar.

The Manual Acknowledgement Override dialog box appears.

The Manual Acknowledgement Override dialog box displays a summary of the Functional Group you selected and two areas:

- **FA Received.** Select Yes or No

- **FA Status Code.** Select one of the four options presented from the **FA Status Code** drop-down menu Accepted, Partial, Rejected, or Error.

3. Click the **Save** button to save your changes, or click Cancel to bypass.

Changes are recorded in the **Partner Manager Log** report for auditing purposes. Additionally, the last manual override change is displayed in the **Acknowledgements** section of the **X12 Functional Group Detail** report.

Note: Partner Manager Security can be used to restrict access to the manual override function. The **Allow Override** check box in the Manual FA Override parameter in Security Groups controls this function. See ["Utilities"](#) for more information.

Functional group view

About this task

The **Functional Group View** displays information about functional groups that meet the specified report selection requirements. The information displayed includes:

| Field | Description |
|------------------|---|
| Transmission ID | A component of the ThreadID assigned by Message Manager system to the transmission. |
| Func'l ID Code | Functional ID code for the functional group from the GS. |
| GS Sender ID | GS address of the trading partner that sent the functional group. |
| Sender TP Name | Name of sending trading partner. |
| Sender AP Name | Name of sending application partner. |
| GS Receiver ID | GS address of the trading partner that received the functional group. |
| Receiver TP Name | Name of receiving trading partner. |
| Receiver AP Name | Name of receiving application partner. |
| FG Control No. | GS Control number of the functional group. |
| FG Version | Functional group version number. |
| FG Date | YYYYMMDD format of the date the functional group was created by the sender. (GS/UNG Envelope) |
| FG Time | HHMMSS format of the time the functional group was created by the sender. (GS/UNG Envelope) |
| Ack Status Cd | Functional acknowledgement status code. |
| Ack Recvd Flag | Flag that indicates receipt (Y or N) of a functional acknowledgement for the functional group sent. |

Procedure

1. To report on functional acknowledgements

2. Specify a **Functional Group Code of FA** in the **Report Specific Search** field.

Message/transaction report

The Message/Transaction report displays information on transactions. You can specify reporting on messages or transactions that match the following selection requirements:

- **Data Type** (select EDIFACT, **TRADACOMS**, or **X12**, or other standard from the drop-down list)
- Specific Post Office Nickname
- Specified Date Range
- Direction (Inbound or Outbound)
- Database (Current or Archive)
- Record **Status of All Records, No Errors**, records with **Errors Only**, or a Specific Status Code
- A Report Specific Search (enter a Message/Transaction Number)

The information displayed includes:

| Field | Description |
|-----------------|--|
| Transmission ID | A component of the ThreadID assigned by Message Manager system to the transmission. |
| TS ID | Transaction set code of the transaction. |
| TS Control No | ST control number of the transaction. |

| Field | Description |
|------------------|---|
| Status | Status code of the transaction set: V indicates Valid, E indicates Error. |
| Sender TP Name | Name of sending trading partner. |
| Sender AP Name | Name of sending application partner. |
| Receiver TP Name | Name of receiving trading partner. |
| Receiver AP Name | Name of receiving application partner. |

The information displayed for TRADACOMS includes:

| Field | Description |
|-------------------|--|
| Transmission ID | A component of the ThreadID assigned by the Message Manager system to the transmission. |
| Message Type | The name of the message in this document, for example INVHD . |
| Doc Body Index | The index of the body in this document. |
| STX Sender Info | The identifier specified in this message for the STX Sender. |
| STX Sender Name | The STX Sender's name, as found in Trading Manager database via the STX Sender Info field, above. |
| STX Receiver Info | The identifier specified in this message for the STX Receiver. |
| STX Receiver Name | The STX Receiver name, as found in Trading Manager database via the STX Receiver Info field, above. |
| SDT Info | The identifier specified in this message for the SDT Partner. |
| SDT Name | The SDT name, as found in Trading Manager database by way of the SDT Info field, above. |
| CDT Info | The identifier specified in this message for the CDT Partner. |
| CDT Name | The CDT name, as found in Trading Manager database by way of the CDT Info field, above. |
| Body Ref | The index of the body in this interchange. |
| Doc Type | The Document Type of this message, for example INVOIC . |
| Map Date | The date the map created this traffic record. |

- [Viewing 997 content](#)
- [Jumping to a functional group](#)

Viewing 997 content

About this task

The **X12 Transaction Report** allows you to view the content of the selected 997 transaction. This is useful to see how your partner responded to your transmission.

To view 997 content

Procedure

1. Select a 997 transaction.
2. Click on the **Display Report Detail** icon on the toolbar.

Results

If 997 content exists, it will be displayed in the right hand side of the **X12 Transaction Detail** window.

Note: The following conditions must be true in order to view 997 content. If user security is in force, the user's Security Group setting for **X12 997 Data: View Data** must be checked. Additionally, to have 997 content written to a database so it can be viewed here, the **Store 997 Content in Database** check box, found in Utilities>Message Manager Configuration, must be enabled.

Jumping to a functional group

About this task

The **X12 Transaction** report allows you to jump directly to the corresponding Functional Group from a 997 transaction.

To jump to a functional group from a 997 transaction

Procedure

1. Select a 997 transaction.
2. Click on the **Jump to Corresponding Functional Group** icon on the toolbar.
The display "jumps" to the functional group from the 997 transaction.
3. Use the report's **Back** button to return to the original report.
Note: Any traffic data converted from releases prior to Trading Manager V7.0 will not have this option available.

X12 TA1 status reports

The X12 TA1 Status reports TA1 status information for the specified date range.

The information displayed includes:

| Field | Description |
|-----------------------|--|
| Transmission ID | A component of the ThreadID assigned by the Message Manager system to the transmission. |
| Sender Ichg ID/Qual | ISA sender ID qualifier and address of the trading partner that sent the interchange. |
| Receiver Ichg ID/Qual | ISA sender ID qualifier and address of the trading partner that received the interchange. |
| Orig IC Ctrl No | Original ISA control number |
| Original IC Time | Original HHMM format of the time the interchange was created by the sender |
| IC Ack Code | ISA control number acknowledgement code. |
| IC Note Code | ISA control number note code |

Additional reports overview

The **Additional Reports** view enables you to generate detailed reports that you can print or export.

Additional reports include:

- Application Partner Detail Report
- EDIFACT Inbound Trade Link Application Report
- EDIFACT Outbound Trade Link Application Report
- Post Office Report
- TRADACOMS Inbound Trade Link
- TRADACOMS Outbound Trade Link
- Trading Partner Detail Report
- X12 Inbound Trade Link Application Report
- X12 Outbound Trade Link Application Report
- Post Office Where Used Report
- [Running additional reports](#)

Running additional reports

About this task

Additional reports are selected from the **File** menu.

To run additional reports

Procedure

1. Click on the **Reports** button on the toolbar.
The Reports window displays.
2. Select Additional Reports from the **Report Type** drop-down box.
3. Select a report from the **Reports** drop-down box.
4. Click the **Run Report** button.
The report-specific form appears.
5. Make your selections.
6. Click the **Run Report** button.

Printing Partner Manager reports

Traffic reports must be exported to xml, html, or text formats prior to being printed. You can then print the exported file from a browser, the Partner Manager Editor, or any text editor.

Exporting reports

Traffic reports use the Traffic Report Export feature.

Additional reports use the export feature.

- [Exporting traffic reports](#)

Exporting traffic reports

About this task

The **Traffic Report Export** feature enables you to export traffic reports to html, xml, and plain text formats.

You can specify the action to take place after the export is completed:

| Action | Description |
|-----------------------------|--|
| Launch Browser | The default browser is launched. |
| Launch Editor | The Partner Manager editor or the user-specified editor (defined in the Display Options utility) is launched. |
| Exit without Launching File | No editor or browser is launched. |

You can specify the number of rows to export and whether to include the traffic status color in the exported file.

If reports are exported to an existing file, the existing file is overwritten.

Note: The file that was overwritten is moved to the recycle bin.

To export a traffic report

Procedure

1. Click the **Export** button.
The Traffic Report Export dialog box appears.
 2. In the **Report Column**, select the columns to export.
 3. Select the **Export Method**.
 4. Select the **After Export Completes** action.
 5. Select the **Number of Rows to Export**.
 6. Enter the full path and filename of the export file or click **Browse** to navigate your file system to **Select/Enter Export File Destination**.
The **File name** defaults to the *reportview.reporttype* name.
 7. Specify the **File name**.
 8. Click **Save**.
If the export file already exists, an overwrite confirmation dialog box appears. Files that are overwritten are moved to the recycle bin.
 9. Click **Begin Export**.
The report is exported.
- [Default traffic report export settings](#)

Default traffic report export settings

About this task

- All report columns are selected for export by default.
- Where applicable, the **Include Traffic Status Color** check box is enabled by default.
- The default export method is **HTML Page**.
- The default behavior to occur after export is **Launch Browser**.
- The default number of Rows to Export is **All**.

Value Added Tax (VAT) reports

About this task

The TRADACOMS subsystem (inbound and outbound) will create VAT reports for any invoice type messages. For example: INVOIC, CREDIT, and UTLBIL. The VAT report will automatically be created in the default *install_dir\tnmgr_vn.n\mmgr\VAT_Reports* directory.

If you wish to change the default directory, you can change the location in the **Edit** Resource dialog box in the **Resource Registry**.

To change the default directory

Procedure

1. Open the **Resource Registry**.
2. Select **Open** from the File menu and navigate to the *install_dir\tnmgr_vn.n\mmgr\mmgr.mrm*
3. Expand the **mmgr** file.
4. Expand the **Resources** file.
5. Double-click the **mmgr_trc_VatReport** file.
The **Edit** Resource dialog box appears.
6. Edit the **Resource Value** to the desired location.
7. Click **OK**.

Partner Manager editor

This editor extracts data from the archives and presents the data to enable access to the original EDI data as presented to Trading Manager.

The Partner Manager Editor is used to:

- Edit and resend interchanges
- View reports exported in html or xml format.
- The Editor can be used to print the exported reports. Other features of the Editor enable you to:
 - Save the exported file
 - Save the exported file in a different format
 - Revert to the saved file (brings in the user-specified editor changes)
 - Launch a user specified editor
 - Find/Replace/Highlight text in the exported file
 - Format the exported file

Note: If you use the **Find/Replace/Highlight** feature and then resize the Editor window, the color changes are not saved.

Note: Use care when using the Editor as the user-specified editor may have multiple files open.

Resending E-commerce data

Inbound and outbound data that has been processed by Trading Manager can be viewed and edited. Run the Interchange traffic report to resend the interchange immediately or edit/resend the interchange data.

The resend interchange data feature extracts data from the Message Manager **archive** folder.

- The resend immediately feature allows you to resend a message immediately, without viewing or editing the original e-commerce data, inbound or outbound.
- The edit/resend interchange feature allows you to edit the interchange, and then resend it.

You must run the Interchange traffic report to access these features.

Note: The Security Group Maintenance, Transmission Data setting controls view, edit, and resend capability.

The ability to resend inbound or outbound data allows you to resubmit e-commerce data that had previously been transformed correctly, but may have encountered transmission errors during processing by the receiving application.

You can view and edit the original e-commerce data prior to resending to your trading partner or internal application. This allows you to update a code that did not pass EDI validation, or create a file containing only a single transaction set that for resubmission.

The **Message Manager** picks up and reprocesses the file that was resent. A new ThreadID reflects this resend activity.

The Server Directory Configuration Maintenance utility configures the location of the archived transmission data and the location where resent data is placed.

Note: For more information on the **Update** and **Resend** components of the Message Manager, see the Trading Manager documentation.

- [**Resend ThreadID**](#)

The **Message Manager System Resend** map creates a new **ThreadID** for the data, changing the origin indicator from original (**o**) to resend (**r**).

Resend ThreadID

The **Message Manager System Resend** map creates a new **ThreadID** for the data, changing the origin indicator from original (**o**) to resend (**r**).

For example:

- Original **ThreadID** - edi00001199809150001**o**.app
- Resent **ThreadID** - edi**00002**199809150001**r**.app

Partner Manager maintains a complete audit trail of the resent data based on its new **ThreadID**.

Utilities

This chapter describes Partner Manager options that are set using the Utilities navigator.

The navigator can be opened by clicking the **Utilities** icon on the Partner Manager toolbar, or by selecting Utilities from the File menu.

Options that can be set using the **Utilities** navigator include the following:

- Configuration
- Database Utilities
- Folders
- Security
- [**Configuration**](#)
- [**Field defaults**](#)

To speed data entry, use this form to enter default values for many of the fields seen in Partner Manager.
- [**Database utilities**](#)

- [Folders](#)
 - [Security](#)
-

Configuration

The **Configuration** utilities contains the following:

- Database Settings
 - Display Options
 - Message Manager Configuration
 - Post Office Search and Replace
 - Post Office Types
 - Server Directory Configuration
 - Standards
 - Field defaults
 - [Database settings](#)
 - [Display options](#)
 - [Message Manager configuration maintenance](#)
 - [Post Office search and replace](#)
 - [Post Office types](#)
 - [Server directory configuration](#)
 - [Standards](#)
 - [Adding and deleting TRADACOMS standards](#)
-

Database settings

In the **Database Settings** utility, you can:

- Change the database you are currently using; see "[Importing a Database Profile](#)"
 - Create a new database profile; see "[Creating a New Database Profile](#)"
 - Import a database profile; see "[Importing a Database Profile](#)"
 - Export a database profile; see "[Exporting a Database Profile](#)"
 - Edit an existing profile; see "[Editing an Existing Database Profile](#)"
 - Delete an existing profile; see "[Deleting an Existing Database Profile](#)"
 - [Database selection](#)
 - [Creating a new database profile](#)
 - [Importing a database profile](#)
 - [Exporting a database profile](#)
 - [Editing an existing database profile](#)
 - [Deleting an existing database profile](#)
 - [Selecting a different database profile](#)
-

Database selection

About this task

Partner Manager lets you change the current database through the Database Selection dialog box.

To change the current database

Procedure

1. From the File menu, select Database **Settings**.
The Partner Manager Database Selection window opens:
2. Select the database vendor from the drop-down menu.
3. Select the Data Source Name (**DSN**) from the drop-down menu.
Note: You create DSNs from the **ODBC** applet in the Windows **Control Panel**.
4. Enter a user ID and password, if required.
5. Enter the database file name.
Note: For Access and Oracle databases, you can leave the **Database** field blank. For all other database types, enter the name of the database, as it exists on the server.
6. (Optional) Enter a description of the database (as it will appear in the title bar).
7. Click **Test Connection**.
The **Connection Test OK** message box opens to inform you that the test has been satisfactorily completed.
8. Click **OK**.
The **Text Connection** message box closes.
9. Click **OK** in the Partner Manager Database Selection window.

A message box opens that informs you that Partner Manager parameters are being updated.

10. Click OK.
Partner Manager closes.
11. Restart Partner Manager to use the new database.

Creating a new database profile

About this task

To create a new database profile

Procedure

1. From the File menu, select Database **Settings**.
The Partner Manager Database Selection dialog box appears.
2. Click Profile Maintenance.
The Back-End Profile Maintenance dialog box opens.
3. Enter a name for the new profile and press **Tab** or **Enter**.
A dialog box appears asking you to confirm the creation of the new profile.
4. Click Yes to create the new database profile.
5. (Optional) Enter a description for the profile.
6. Select the Data Source Name (**DSN**) from the drop-down menu.
7. Enter a user name and password, if required.
8. Enter the database file name.
For Access and Oracle databases, leave the **Database** field blank. For all other database types, enter the name of the database, as it exists on the server.
9. Select a database vendor from the drop-down list.
10. Click **Test Connection** to confirm the database connection.
11. Click **Save** and **OK** to confirm the new profile.
12. Click **Cancel** to close the dialog box.

Results

You can now select the profile from the **Profile** drop-down list.

Importing a database profile

About this task

To import a database profile

Procedure

1. Open the Back-End Profile Maintenance window as described in "[Creating a New Database Profile](#)".
2. Click the **Import** button.
The Profile Import window opens.
3. Select an Import Source:
 - If you are importing from a file, click the **Browse** button and navigate to the file that you want to import. The path will be displayed in the **File to import** field.
 - If you are pasting XML from an existing file, select the **Paste from Clipboard** button to activate the **Paste Export File Here** text entry box. Click in the **Paste Export File Here** field and use the windows copy feature to copy the XML from the clipboard.
4. Select an Import Option:
 - If you want over-write the existing profile, select the **Over-write Existing** button.
 - If you do not want to over-write the existing profile (skip it), select the **Skip Existing** button.
5. Click the **Import** button.

Exporting a database profile

About this task

To export a database profile

Procedure

1. Open the Back-End Profile Maintenance window as described in "[Creating a New Database Profile](#)".
2. Click the **Export** button.
The **Profile Export** window opens.
3. Select a Profile ID by placing a check beside the desired profile in the list, or if you want to select all of the profiles, click the **Select All** button.
4. Choose the export method:
 - If you want to export the profile to a file, click the Export to File checkbox, then use the **Browse** button to navigate to the file you want.
 - If you want to export to clipboard, check the Export to Clipboard checkbox.
5. Click the **Export** button.
 - If you selected Export to File, the profile is copied to the file that you selected.
 - If you selected Export to Clipboard, the profile is copied to the clipboard.

Editing an existing database profile

About this task

To edit an existing database profile.

Procedure

1. From the File menu, select Database **Settings**.
The Partner Manager Database Selection dialog box appears.
2. Click Profile Maintenance.
The Back-End Profile Maintenance dialog box appears.
3. Select an existing profile from the **Profile** drop-down list.
4. Edit the settings.
Note: For Access and Oracle databases, leave the **Database** field blank. For all other database types, enter the name of the database, as it exists on the server.
5. Click Test **Connection** to confirm the database connection.
6. Click Save.
A dialog box opens that confirms successful database connection.
7. Click OK.
8. Close Partner Manager.
9. Restart Partner Manager.

Deleting an existing database profile

About this task

To delete an existing database profile

Procedure

1. From the File menu, select Database **Settings**.
2. The **Partner Manager Database** Selection dialog box appears.
3. Click Profile Maintenance.
The Back-End Profile Maintenance dialog box appears.
4. Select a profile from the **Profile** drop-down list.
5. Click Delete.
A confirmation box opens that asks you to confirm the delete.
6. Click Yes to delete the profile.
7. Click Cancel to close the Back-End Profile Maintenance dialog box.
8. Click OK to close the Partner Manager.
9. Restart Partner Manager.

Selecting a different database profile

About this task

To select a different database profile

Procedure

1. From the File menu, select Database **Settings**.
2. The **Partner Manager Database** Selection dialog box appears.

3. Select an existing profile from the **Profile** drop-down list.
 4. Click OK to close the Partner Manager.
 5. Restart Partner Manager.
-

Display options

About this task

The **Display Options** utility allows you to modify the display according to your personal preferences. Display options are set using the Display Options dialog.

Opening the Display Options dialog

Procedure

1. From the Tools menu, select Display Options.
The Display Options dialog box opens.
 2. Refer to the following sections to specify display options.
 - [Specifying processing options](#)
 - [Specifying transmission data options](#)
 - [Change the look and feel of the display](#)
-

Specifying processing options

About this task

The following procedures describe how to set processing options for your display.

To specify processing options

Procedure

1. Open the Display Options dialog as described in the Display Options section. Make certain that the Processing Options tab is selected.
 2. Select the most recently used trade links to display when you open Partner Manager by enabling or disabling the **Display Last Inbound Trade Link** check box, and/or the **Display Last Outbound Trade Link** check box.
 3. Specify the default date selection for all traffic reports. Select Today, or **All Dates**.
 4. Select Standards (**X12**, **EDIFACT** or **TRADACOM**).
Note: The **Standards** check boxes allow you to restrict display of EDI Standard-Specific options. For example, if your company does not use EDIFACT or TRADACOMS, you can disable these options and the screens specific to these two standards will not appear. You can always restore access to these standards by enabling the check box. This setting only applies to the PC running Partner Manager; it does not affect other users
 5. When you are satisfied with your selections, click Save.
-

Specifying transmission data options

About this task

The following procedures describe how to specify the editor and temporary directory to be used for viewing and editing transmission data in the traffic reports and during the traffic export.

To specify the default editor/viewer

Procedure

1. Open the Display Options dialog as described in the Display Options section.
2. Select the Transmission Options tab.
3. For the **Editor/Viewer**, enable an option button:
Default to Built-in Editor or
Default to User-Specified Editor
 4. If you choose to default to a user-specified editor, enter the full path of that editor application executable program file in the **User Specified Editor** field.
(For example, **C:\WINNT\Notepad.exe**).
Note: The user-specified editor must accept a file name as a parameter. The user-specified editor can be launched by default or launched from the Partner Manager Editor.
 5. Specify the **Temporary Directory**.
Note: While an interchange is being edited, it is placed in this temporary directory until the editing/resend operation is complete.
6. When you are satisfied with your selections, click Save.

Change the look and feel of the display

About this task

The following procedures describe how to make changes to the look and feel of the display based upon your personal viewing preferences.

To make changes to the Display

Procedure

1. Open the Display Options dialog as described in the Display Options section.
2. Select the Look and Feel tab.
3. If you want to hide the IBM banner that appears at the top of the Partner Manager application, disable the **Display IBM Banner** check box.

4. Select Startup Options:

If you want Partner Manager Windows to display in standard Windows default mode, enable the **Windows Default** check box.

If you would prefer the active window always be displayed on top of all others, enable the **Display at Top** check box.

Note: You may find that the Display at Top mode of operation is useful if you are using a lower resolution screen display (for example 800 x 600).

5. When you are satisfied with your selections, click Save.

Message Manager configuration maintenance

About this task

The Message Manager Configuration Maintenance window is used to specify the run-time parameters used by the system of maps in the Message Manager portion of Trading Manager.

These settings apply to all users of the current database (the current database is displayed in **Help > About**). If you have multiple Trading Manager databases (for example, Test or Production), you may want to set these parameters to all of them.

To specify Message Manager parameters

Procedure

1. From the Tools menu, select Message Manager Configuration.
The Message Manager Configuration Maintenance window opens.

2. Make changes to the fields in this window by selecting one of six tab views.

For a detailed description of fields contained in the tab views, refer to the tab view discussed in the following sections:

- ["Optimization Tab"](#)
- ["Mapping Options Tab"](#)
- ["Database Tab"](#)
- ["X12 Acknowledgements Tab"](#)
- ["Archive Options Tab"](#)
- ["Traffic Batch Archiving Tab"](#)
- ["Alerts tab"](#)
- ["Post Office Options tab"](#)

3. Click Save.

The information that you have entered using this form is saved.

- [Optimization tab](#)
- [Mapping Options tab](#)
- [Database tab](#)
- [Acknowledgements tab](#)
- [Archive Options tab](#)
- [Traffic batch archiving tab](#)
- [Alerts tab](#)
- [Post Office Options tab](#)

Optimization tab

Open the Message Manager Configuration Maintenance window with the Optimization tab selected.

Optimization Tab Field Descriptions

The Message Manager Configuration Maintenance window displays the fields described in the following table when the Optimization tab is selected:

| Field | Description |
|-------|-------------|
|-------|-------------|

| Field | Description |
|--|--|
| Maximum X12 FG Validation Size | The value entered represents the maximum size of a functional group that will be validated in a single pass. If a functional group is larger than this size, then each transaction will be validated individually. Best performance is achieved when validating the functional group in a single pass, but it will consume that much more memory and might exceed the capacity of the server and cause Trading Manager to fail. Setting this value allows for smaller functional groups to be processed as fast as possible while larger functional groups will not cause undue crashes or performance degradation. |
| Work File Options | The options available from this drop-down menu are used by Message Manager to determine the work file setting for all RUN maps used in the system. These options are:
-WM - places all work files in memory, which increases performance, but uses memory.
-W - places work files as physical files on the disk, using default work file names, which results in slower performance but reduces memory usage.
Note This option should only be used in specific debugging cases as it forces all RUN maps to be single threaded. If work files on disk are desired, use option -WU (see below).
-WD - indicates default work files on disk, delete when done.
-WU - indicates uniquely named work files on disk, delete when done. |
| One File per Put PO | For outbound files, creates one file containing all the interchanges destined for that specific put Post Office. This eliminates the need to develop a custom system to append all the data into a single file for delivery. |
| One File per Put PO per Interchange | For outbound files, this option allows this version of Trading Manager to run as it did in previous versions and creates one output file for each interchange. |
| X12 Inbound Data Containing Multiple Functional Groups | Check this box to route one file for each functional group. Uncheck to route one file consolidating all functional groups. |

Mapping Options tab

Open the Message Manager Configuration Maintenance window with the Mapping Options tab. Open the window as described previously. Refer to the following table for field descriptions.

Mapping Options Tab Field Descriptions

Open the Message Manager Configuration Maintenance window. The following fields are displayed when the Mapping Options tab is selected.

| Field | Description |
|---------------------------------|--|
| Maximum Reject File Size | This text entry field allows you to enter a value that restricts the amount of EDI data that can be shown in the reject cards. The Message Manager subsystems produce error reports for any EDI data content that fails validation. If the size of the reject data is greater than the value shown, then data will not be present in the error report. The data will still be archived and can be re-sent. A value of '0' means that the Reject data will always be shown in the Error Report. |
| Minimum Reject Data Size | This text entry field allows you to set a value that will cause Message Manager to ignore meaningless EDI data between interchanges or attached at the end, such as extra "new line" characters or messages from a VAN. The value should be set to less than the smallest size of an interchange, or data in general, that you expect to receive. |
| Optional X12/EDIFACT Validation | When enabled, this check box performs: For X12: IEA/GE segment count comparisons; IEA/ISA and GE/GS control number comparison and ST01/GS01 Validation. For EDIFACT: UNB/UNZ and UNG/UNE control number comparison and segment count comparison. |

Database tab

Open the Message Manager Configuration Maintenance window with the **Database** tab selected. Open the window as described previously in this section. Refer to the following table for field descriptions.

This tab view also provides the **Run Export** button. For details see the [Run Export](#).

Database Tab Field Descriptions

The Message Manager Configuration Maintenance window displays the fields described in the following table when the Database tab is selected.

| Field | Description |
|-------------------------------|--|
| Use Stored Procedures | When checked, stored procedures will be used for outbound control number generation and thread id generation on Oracle, SQL Server, Sybase, and DB2 (Non-z/OS). Not applicable for Access or DB2 running on z/OS. |
| Sort TSV Files During Extract | If this check box is enabled, Message Manager will re-sort the tables as they are being created in the export process. This is required when non-alphanumeric or lower/uppercase characters are used to uniquely identify Trading or Application Partners, and your database sort order does not match IBM's sort order. |
| Current DB Vendor | This is a read-only field used by Message Manager to generate SQL statements when they are database dependent. |

Run Export

The **Run Export** button allows you to trigger the Export process on demand, without having to cycle the Launcher. By default, Trading Manager is configured to run Export to synchronize the tsv files with the database at Launcher startup and every day at midnight. You can use the **Run Export** button if you want the tsv files to be synchronized immediately without having to wait for the scheduled time.

Note: Running unnecessary Exports adds overhead to Message Manager and can interfere with the processing of EDI data. You can review the results of a successful export by viewing the Partner Manager log file.

Acknowledgements tab

Open the Message Manager Configuration Maintenance window with the **X12 Acknowledgements** tab selected. Open the window as described previously. Refer to the following table for field descriptions.

Acknowledgements tab field descriptions

The Message Manager Configuration Maintenance window displays the fields described in the following table when the **Acknowledgements** tab is selected.

| Field | Description |
|---|--|
| Store 997 Content in Database | When enabled, this check box allows the content of all 997's received to be stored in the database so they can be viewed in traffic reports. Storing 997 data in the database increases the size of the database and can have an impact on Message Manager performance. |
| Ignore Missing Trade Links for Inbound X12 997's and EDIFACT CONTRL | If you do not want to custom process inbound 997/CONTRL, you can skip entering Trade Links for them and thus prevent Message Manager from generating Trade Link errors. Message Manager will still reconcile these acknowledgements against the outbound transactions recorded in the Partner Manager database regardless of this setting. |
| Create one FG for each 997 Transaction | By default Trading Manager will wrap as many 997 transactions in a single functional group as possible. When enabled, this option forces each 997 transaction to be wrapped in its own functional group. |
| No TA1s | Disables Trading Manager TA1 processing. |
| Create TA1 in TA1 Folder | Enables creating error TA1s and places them in the x12\TA1 folder. |
| Create TA1 and Route to Partner | Enables creating error TA1s and routes them to the external partner. |
| Create TA1 in TA1 Folder and Route to Partner | Enables creating error TA1s and places them in the x12\TA1 folder and also routes them to the external partner. |
| Create 000 TA1's | When enabled, this check box allows Message Manager to build TA1s for interchanges that do not fail the TA1 envelope check., as Message Manager, by default, only creates TA1s for interchanges that fail the check. These TA1s have a code of 000. |
| Default TA1 Post Offices | If TA1 routing is selected, these options have to be entered to properly route TA1. They define the default GET and PUT PO for any TA1 created. These default POs can be overridden for each partner in the Address Book. |

Archive Options tab

Open the Message Manager Configuration Maintenance window with the Archive Options tab selected. Open the window as described previously. Refer to the following table for field descriptions.

Archive Options tab field descriptions

The Message Manager Configuration Maintenance window displays the fields described in the following table when the **Archive Options** tab is selected.

| Field | Description |
|---|--|
| Inbound Transmission as delivered to Trading Manager (zip/tar) | Archive all inbound transmissions into Trading Manager in a zip or tar format. The archived files are located in <code>install_dir\tmgr_n.n\mmgr\archive\stampfil</code> folder and are not used by Trading Manager. Users that already have archiving processes somewhere else can disable this feature for better performance. |
| Outbound Transmission as delivered to Trading Manager (zip/tar) | Archive all outbound transmissions into Trading Manager in a zip or tar format. The archived files are located in <code>install_dir\tmgr_n.n\mmgr\archive\stampfil</code> folder and are not used by Trading Manager. Users that already have archiving processes somewhere else can disable this feature for better performance. |
| Inbound Interchange (Text) | Archives all inbound interchanges in text format. These files are used by Partner Manager's edit/resend feature. Feature can be disabled if Edit/Resend is not used. |
| Outbound Interchange before Trading Manager re-enveloping (Text) | Archives all outbound interchanges before re-enveloping in text format. These files are used by Partner Manager's edit/resend feature. Feature can be disabled if the Edit/Resend features are not used or not required. |
| Outbound Interchange after Trading Manager re-enveloping (Text) | Archives all outbound interchanges after re-enveloping in text format. These files are used by Partner Manager's edit/resend feature. Feature can be disabled if the Edit/Resend features are not used or not required. |
| Outbound Transmission after Trading Manager re-enveloping (zip/tar) | Archive all outbound transmissions out from Trading Manager in a zip or tar format. The archived files are located in the <code>install_dir\tmgr_n.n\mmgr\archive\sendx12</code> , <code>install_dir\tmgr_n.n\mmgr\archive\senddef</code> , or <code>install_dir\tmgr_n.n\mmgr\archive\sendtrc</code> folder (based on the data format) and are not used by Trading Manager. Users that already have archiving processes somewhere else can disable this feature for better performance. |

Traffic batch archiving tab

The Traffic Batch Archiving option automatically moves traffic to the on-line archive using Message Manager. Data will be kept for the number of days specified. The results of this archiving is reported in the Partner Manager log.

Open the Message Manager Configuration Maintenance window with the **Traffic Batch Archiving** tab selected. Open the window as described previously. Refer to the following table for field descriptions.

Traffic batch archiving tab field descriptions

The Message Manager Configuration Maintenance window displays the fields described in the following table when the **Traffic Batch Archiving** tab is selected.

| Field | Description |
|--------------------------------|--|
| Enable Batch Traffic Archiving | This flag enables the batch traffic archiving. |
| Days to Keep | Specifies how many days worth of traffic is to be left in the traffic tables. If you enter a 0 in this field, all traffic will be moved to the online archive. |

If you enable this option, any control numbers currently used for duplicate interchange control number checking are removed at the point that traffic is being archived, not at the point the option is enabled. The utility displays a warning to notify you of this before the setting is saved.

Alerts tab

When an alert is created by Message Manager, it is usually sent to the alert email address associated with the trading partner. There are cases, however, where an alert is generated before the trading partner can be determined. Set the fields on this tab to send an alert in such cases. You may also attach the data causing the alert using the checkbox Attach Data.

Post Office Options tab

The Post Office options tab contains a checkbox that allows you to enable the Fixed Post Office. If you check the **Use One GET Post Office per Standard** checkbox to disable all Get post office functionality in Partner Manager and the XML Autoload, and in its place use 3 Get post offices, one for each standard.

The Post Office options tab contains a checkbox that allows you to enable the Fixed Post Office. If you check the **Use One GET Post Office per Standard** checkbox to disable all Get post office functionality in Partner Manager and the XML Autoload, and in its place use 3 Get post offices, one for each standard. This will simplify usage by eliminating the need to continually specify where data is received.

If you uncheck this checkbox, the Launcher functionality method is restored and all original Get post offices are re-populated.

When using "One GET Post Office per Standard", the post office nicknames will be "X00" for X12, "E00" for EDIFACT, and "T00" for TRADACOMS, unless you already have a post office nickname with those values. If you already are using these nicknames, the final two numbers will be incremented until a unique value is reached. For example, if you previously have a "X00" and a "X01" post office nickname, the new post office nickname will be "X02".

Note: Close all Partner Manager windows before changing this functionality. In addition, all Message Manager functions must be stopped.
If you select the single Get post office option, the following functionality is affected in Partner Manager:

- The ability to change, "jump to" or "get info" on a Get post office on all inbound and outbound Trade Links is disabled.
- Post office maintenance on Get PO's is disabled
- Get post office entry in the Automatic Acknowledgements windows for X12 and EDIFACT is disabled
- You cannot delete any Get post offices
- The Get Post Offices in the Utilities...Field Defaults window is disabled
- The Get post offices in existing traffic report data are not converted as they represent historical data. You will still be able to select previously used Get post offices as selection criteria in the Reports window.
- You will not be able to save a Put post office having the same name or nickname as a Get post office that existed before you selected the "Use One GET Post Office per Standard". This is required in order to enable a restore of original Get PO's if the single Get PO option has been enabled, and then subsequently disabled.
- To maintain database consistency, the merge function is disabled if merging to a database where the value of this setting differs from the source value.
- In the initial Post Office view, the copy options on the Get PO are disabled.

Post Office search and replace

About this task

The Post Office Search and Replace function is used to replace Post Office Directory/Adapter command strings when deploying Message Manager on a different system.

To search and replace Post Offices

Procedure

1. Click the **Utilities** icon on the toolbar.
The **Utilities** navigator opens.
2. Click Configuration > **Post Office Search and Replace**.
The Post Office Search and Replace window opens.
3. Select a Post Office Type to be replace. You can choose to replace File Post Offices, All Post Offices, or you can choose a specific Post Office Type from the drop-down list.

4. Enter the existing Post Office Directory/Adapter Command string in the **Find What** text entry field.
 5. Enter the string that you want to replace the existing string with in the **Replace With** field.
 6. Click Replace.
-

Post Office types

The Post Office types window allows you to mark post offices that are not used by your organization as inactive. This window also allows you to add and delete post offices.

- [Marking a Post Office type inactive/active](#)
 - [Adding a Post Office type](#)
 - [Deleting a Post Office type](#)
-

Marking a Post Office type inactive/active

About this task

You can mark any post office type in the list as inactive as long as there are no post offices assigned to it. After the post office is marked as inactive, it will not appear in the list of post offices in the **Post Office** navigator.

To inactivate/activate Post Offices

Procedure

1. Click the **Utilities** icon on the toolbar.
The **Utilities** navigator opens.
 2. Click Configuration > **Post Office Types**.
The Post Office Types dialog opens.
 3. Select the Post Office that you want to make inactive and click **Edit**.
The Post Office Types Maintenance dialog opens.

Note: You cannot mark any post offices types as inactive if they currently have post offices assigned to them.
 4. Enable the **Inactive** check box.
You are returned to the **Post Office Types** form. Notice that a **Y** (Yes) now appears in the **Inactive** column of the form on the line associated with the post office that you deactivated.
 6. To make the Post Office active again, perform steps 1 - 3 of this procedure. When the Post Office Types Maintenance window opens, deactivate the Inactive check box and click **Save**.
An **N** (No) is displayed in the **Inactive** column of the Post Office Types dialog on the line associated with the post office that you activated.
-

Adding a Post Office type

About this task

You can add a Post Office Type.

To add a post office type

Procedure

1. Click Utilities icon on the toolbar.
The **Utilities** menu opens.
 2. Click Configuration > **Post Office Types**.
The Post Office Types dialog opens.
 3. Click **Add**.
The Post Office Types Maintenance dialog opens.
 4. Enter the post office type in the **Post Office Type** field.
 5. Enter a description of the post office type in the **Description** field.
 6. Click **Save**.
You are returned to the Post Office Types window. The new post office is displayed in the list.
-

Deleting a Post Office type

About this task

You can delete a Post Office Type.

Note: Supplied Post Offices cannot be deleted. These post offices include: OracleAQ; MessageQ Client; MessageQ Server; E-mail; File; FTP; HTTP; JMS; MQSeries Client; MQSeries Server; MSMQ; and VAN.

To delete a post office type

Procedure

1. Click the **Utilities** icon on the toolbar.
The **Utilities** menu opens.
2. Click Configuration > **Post Office Types**.
The Post Office Types dialog opens.
3. Select the post office to be deleted.
4. Click Delete.
A message box opens that asks you to confirm the delete.
5. Click Yes.
You are returned to the Post Office Types dialog. The post office is removed from the list.

Server directory configuration

About this task

The **Server Directory Configuration Maintenance** utility is used to:

- Specify the archive access method
- Specify FTP Connection Information
- Specify the **Archive** and **Resend** transmission data folder locations
- Specify the archive audit information
- Allow you to test the archive access and folder configuration

To access the Server Directory Configuration Maintenance utility

Procedure

1. From the Tools menu, select Server Directory Configuration.
The Server Directory Configuration Maintenance window opens.
2. Select the access method as either **FTP** or **Mapped Drive** by enabling the check box.
If **FTP** is selected, enter information in the fields in the **FTP Connection Information** area of the form.
3. Define the location of the archive and resend data folders.
4. Define the location of the share data folders. This initialization is needed for specifying the location of alias files and for use of the manual export function in the Message Manager Configuration utility.
See "[Managing Traffic Data](#)" for detailed information.

Standards

Note: **Incorrect use of this feature may result in invalidation of your trade links.** Ensure that you back up your Partner Manager database files before attempting the procedures described in this section.

The **Standards Maintenance** utility enables you to maintain X12, EDIFACT, and TRADACOMS standards such as versions, ID codes, and related standards components. You can add, edit, and delete versions.

The following procedures describe how to perform maintenance on X12, EDIFACT and TRADACOMS Standards.

- [Adding an X12 standard version](#)
- [Deleting an X12 standard version](#)
- [Adding X12 interchange standards or ID qualifiers](#)
- [Adding an X12 hex value](#)
- [X12 Structure Validation:Import](#)
- [X12 Structure Validation:Maintenance](#)
- [Deleting X12 interchange standards or ID qualifiers](#)
- [Selecting HIPAA institutions](#)
- [Importing HIPAA type qualifiers](#)
- [Adding an EDIFACT standard version](#)
- [Adding an EDIFACT partner ID code qualifier](#)
- [Adding an EDIFACT functional ID or message type](#)
- [Adding an EDIFACT hex value](#)
- [Deleting an EDIFACT standard](#)

Adding an X12 standard version

About this task

When you add a new X12 standard version, you must also add a functional ID and transaction set associated with that version. Partner Manager enables you to copy functional ID and transaction sets from an existing version. For example, you can create copies of versions with new version names.

To perform maintenance on X12 standards

Note: To exit from the Standards Maintenance window without adding a version or completing an add task, you must click Cancel. If you attempt to exit the dialog box by closing Partner Manager or the Partner Manager process, you may damage your Partner Manager database.

Procedure

1. From the Tools menu, select Standards.
The Standards Maintenance window opens.
2. Select X12 Versions from the **X12 Standards** drop-down menu.
3. Enter a version number in the **Version** field.
4. Click Save.
The Copy Transactions/Func ID's? dialog box appears.
5. Perform one of the following:
 - To copy functional IDs and transaction sets from another version, enter that version number and click OK.
 - To search for and select a functional ID, click Cancel.
6. Select X12 Functional ID from the **X12 Standards** drop-down menu.
The Standards Maintenance X12 Functional ID window appears.
7. Perform one of the following:
 - Enter a functional ID code in the **Functional ID Code** field.
 - Click Display Functional ID's to select a functional ID from the list box.
8. Click Save.
9. Select X12 Transaction Set from the **X12 Standards** drop-down menu.
The Standards Maintenance X12 Transaction Set window opens.
10. Perform one of the following:
 - Enter a transaction set ID in the **Transaction Set ID** field, or
 - Click Display Transaction Sets to select a transaction set from the scroll box.
11. Complete the **Transaction Set Description** field.
12. Click Save.

Deleting an X12 standard version

About this task

To delete an X12 standard version with associated transaction sets and functional IDs, delete its associated transactions sets and functional IDs first.

To delete an X12 standard version

Procedure

1. From the Tools menu, select Standards.
The Standards Maintenance form opens.
2. Select X12 Transaction Set from the **X12 Standards** drop-down menu.
The Standards Maintenance X12 Transaction Set window appears.
3. Click the **Display Transaction Sets** button.
The Find All Transaction Sets window opens.
4. Choose the transaction set associated with the version and functional ID and click Select.
You are returned to the Standards Maintenance X12 Transaction Set window.
5. Click Delete.
6. Select X12 Functional ID from the X12 Standards drop-down menu.
7. Click the **Display Functional IDs** button.
8. Select the functional ID associated with the version and click Select.
9. Click the **Delete** button.
10. Select X12 Versions from the X12 Standards drop-down menu.
11. Enter a version number in the **Version** field or click Display Versions to select a version number.
12. Click Delete.

Adding X12 interchange standards or ID qualifiers

About this task

You can add standards and ID qualifiers.

To add an interchange standard or ID qualifier

Procedure

1. From the Tools menu, select Standards
2. Select one of the following from the **X12 Standards** drop-down menu:
 - **X12 Interchange Standards**
 - **X12 Interchange ID Qualifiers**The **Standards Display X12 Interchange Standards**, or the Standards Display X12 Interchange ID Qualifiers window opens.
3. Enter a qualifier number in the **Qualifier** field.
4. Or click **Display Interchange Standards** or **Display Interchange ID Qualifiers** to select a number.
5. Click **Save**.
6. Click **Cancel**.

Adding an X12 hex value

About this task

Each X12 hexadecimal value supports up to two individual values, for example, 0D 0A (which is CR LF).

To add an X12 hex value

Procedure

1. From the Tools menu, select Standards.
2. Select X12 Hex Values from the **X12 Standards** drop-down menu.
The Standards Maintenance X12 Hex Values window appears.
3. Enter a hexadecimal value in the **Hex Value** field.
4. Enter a description in the **Description** field.
5. Click **Save**.

X12 Structure Validation:Import

About this task

Imports a file that is used by Message Manager to validate X12 Structure. This is a required step in the installation process for all databases other than Access.

To import an X12 Structure Validation file

From the Tools menu, select Standards.

Procedure

1. Select X12 Structure Validation:Import from the **X12 Standards** drop-down list.
The Standards Maintenance X12 Structure Validation: Import window appears.
2. Click the **Find File** button to browse for the file you wish to import.
The **StructureValidation.detl** file, located in the *install_dir\tmgr_vn.n\pmgr* directory, must be imported during installation.
3. Click **Begin Import**.

X12 Structure Validation:Maintenance

About this task

Allows additions, updates, and deletions of X12 Structure information.

Note: This is useful when new/custom standards are used.

To maintain X12 Structure Validation data

Procedure

1. From the Tools menu, select Standards.

2. Select X12 Structure Validation: Maintenance from the **X12 Standards** drop-down list.

The Standards Maintenance X12 Structure Validation: Maintenance window appears.

3. Enter information in the **Version**, **Transaction Set ID** and **Structure Text** fields.

4. Click Save.

Clicking Save will add or update the structure text.

Note: When you click the **Delete** button to delete a version, a dialog box appears instructing you to first delete the transaction set.

Deleting X12 interchange standards or ID qualifiers

About this task

You can delete standards and qualifiers.

To delete an interchange standard or ID qualifier

Procedure

1. From the Tools menu, select Standards.

2. Select one of the following from the **X12 Standards** drop-down list:

- **X12 Interchange Standards**
- **X12 Interchange ID Qualifiers**

The Standards Maintenance X12 Interchange Standards window, or the Standards Maintenance X12 Interchange Qualifiers window opens.

3. Enter a qualifier number in the **Qualifier** field.

- Or click Display Interchange Standards (if you opened the Standards Maintenance X12 Interchange Standards window) or
- **Display Interchange ID Qualifiers** (if you opened the Standards Maintenance X12 Interchange Qualifiers window) to select a number.

4. Click Delete.

Selecting HIPAA institutions

About this task

The **HIPAA Institutions** option allows you to populate the HIPAA Type 7 Institutions field in the X12 External Application Partners window.

To select HIPAA Institutions

Procedure

1. From the Tools menu, select Standards.

2. Select HIPAA Institutions from the **X12 Standards** drop-down menu.

3. Click the **Display Institutions** button.

The Find All HIPAA Institutions window opens.

4. Click on the **Medicare** option and click the **Select** button.

You are returned to the Standards Maintenance window.

5. Click Save.

Importing HIPAA type qualifiers

The HIPAA Type Qualifiers: Import menu item is available from the X12 Standards menu in the Standards Maintenance window. The use of this option is described in the Trading Manager Installation documentation.

Adding an EDIFACT standard version

About this task

An EDIFACT version label consists of three required fields and one optional field:

- Message Version Number
- Message Release Number
- Controlling Agency
- Association Assigned Code (optional)

To add an EDIFACT standard version

Procedure

1. From the Tools menu, select Standards.
2. Select EDIFACT Message Version Numbers from the EDIFACT Standards menu.
The Standards Maintenance EDIFACT Message Version Numbers window opens.
3. Enter a version number in the **Message Version No.** field.
4. Enter a description in the **Description** field.
5. Click Save.
6. Select EDIFACT Message Version Numbers from the EDIFACT Standards menu.
7. Enter a release number in the **Message Release No.** field.
8. Enter a description in the **Description** field.
9. Click Save.
10. Select EDIFACT Controlling Agency from the EDIFACT Standards menu.
11. Enter an abbreviation representing the version agency (such as the **UN**) in the **Controlling Agency** field.
12. Enter a description in the **Description** field.
13. Click Save.
14. (Optional) Select EDIFACT Association Assigned Code from the EDIFACT Standards menu.
15. Enter an abbreviation representing the association in the **Assoc Assign Cd** field.
16. Enter a description in the **Description** field.
17. Click Save.

Adding an EDIFACT partner ID code qualifier

About this task

You can add partner ID code qualifiers.

To add an EDIFACT partner ID code qualifier

Procedure

1. From the Tools menu, select Standards.
2. Select EDIFACT Partner ID Code Qualifiers from the EDIFACT Standards menu.
3. The Standards Maintenance EDIFACT Partner ID Code Qualifiers window opens.
4. Enter a code in the **ID Code Qualifier** field.
5. Enter a description of the partner in the **Description** field.
6. Click Save.

Adding an EDIFACT functional ID or message type

About this task

Adding an EDIFACT Functional ID/Message Type requires that you first add an EDIFACT standard version, as described in ["Adding an EDIFACT Standard Version"](#).

To add an EDIFACT functional ID or message type

Procedure

1. From the Tools menu, select Standards.
2. Select EDIFACT FuncIDMsgType from the EDIFACT Standards menu.
The Standards Maintenance EDIFACT Functional ID and Message Types window appears.
3. Select a message version number from the **Msg Version No.** list.
4. Select a message release number from the **Msg Release No.** list.
5. Select a controlling agency number from the **Controlling Agency** list.
6. (Optional) Select an associate assigned code from the **Assoc Assign Code** drop-down list.
7. Enter an abbreviation representing a functional ID or message type in the **FuncID/MsgType** field.
8. Enter a description in the **FuncID/MsgType Description** field.
9. Click Save.

Adding an EDIFACT hex value

Each EDIFACT hex value supports up to three individual values, for example, 27 0D 0A, as shown in the following table. The EDIFACT hex values delivered with Partner Manager are as follows:

| Hex | Value |
|-----|-----------------------|
| 1C | IS4 (File separator) |
| 1D | IS3 (Group separator) |

| Hex | Value |
|----------|--|
| 1E | IS2 (Record separator) |
| 1F | IS1 (Unit separator) |
| 20 | Space character |
| 27 | Single quote character (') |
| 27 0D 0A | Single quote-carriage return-line feed (` CR LF) |
| 2B | Plus sign (+) |
| 2C | Comma character (,) |
| 3A | Colon character (:) |
| 3F | Question mark character (?) |

To add an EDIFACT hex value

1. From the Tools menu, select Standards.
2. Select EDIFACT Hex Values from the **EDIFACT Standards** drop-down menu.
The Standards Maintenance EDIFACT Hex Values window opens.
3. Enter a hexadecimal value in the **Hex Value** field.
4. Enter a description in the **Description** field.
5. Click Save.

Deleting an EDIFACT standard

About this task

The following procedure describes how to delete the EDIFACT standards:

- Version
- Partner ID code qualifier
- Functional ID or message type

To delete EDIFACT standards

Procedure

1. From the **Tools** menu, select **Standards**.
2. Select a standard type from the **EDIFACT Standards** drop-down menu.
3. Enter the standard type information or click the display button to select a standard.
For example, to delete a message version number:
 - Click the **Display Message Version Numbers** button.
 - Choose a version number and click **Select**.
4. Click the **Delete** button.

Adding and deleting TRADACOMS standards

This section describes how to add and delete X12 Standards.

- [Adding a TRADACOMS document type](#)
- [Editing a TRADACOMS document type](#)
- [Deleting a TRADACOMS document type](#)

Adding a TRADACOMS document type

About this task

The following procedure describes how to add a TRADACOMS document type.

To add a TRADACOMS document type

Procedure

1. From the Tools menu, select Standards.
2. Select Document Types from the **TRADACOMS Standards** drop-down menu.
The Standards Maintenance window displays the TRADACOM information.
3. In the **Document Type** field, enter the document type you wish to add.
4. In the **Segment Versions** group box, enter the **Header**, **Trailer**, and **TAXCON Version** information assigned to this Document Type.
Note: The **TAXCON Version** is only enabled for **Invoice**, **Credit**, and **Utility Bill** types.
5. Enter the description in the **Description** field.

6. Click Save.

Editing a TRADACOMS document type

About this task

The following procedure describes how to edit an existing TRADACOMS document type.

To edit a TRADACOMS document type

Procedure

1. From the **Tools** menu, select **Standards**.
The Standards Maintenance window appears.
2. Select Document Types from the **TRADACOMS Standards** drop-down menu.
The Standards Maintenance window displays TRADACOM fields.
3. Click the **Display Document Types** button.
The **Find All TRADACOMS Document Types** window appears.
4. Choose a document type, and then click **Select**.
5. Edit the data fields as needed.
Note: The **TAXCON Version** is only enabled for **Invoice**, **Credit**, and **Utility Bill** types.
6. Click Save.

Deleting a TRADACOMS document type

About this task

The following procedure describes how to delete a TRADACOMS document type.

To delete a TRADACOMS document type

Procedure

1. From the Tools menu, select Standards.
2. Select Document Types from the TRADACOMS Standards drop-down menu.
3. Click Display Document Types.
4. The Find All TRADACOMS Document Types dialog box displays.
5. Choose a document type and click **Select**.
6. Your selection appears in the **Document Type** field.
Note: The remaining fields will automatically fill in after selecting the document type.
7. Click Delete.

Field defaults

To speed data entry, use this form to enter default values for many of the fields seen in Partner Manager.

Once entered here, these values will display when you are creating a new object such as a partner or trade link. Note that these defaults apply to users of the current database. The following default values can be entered.

- X12 ISA values such as ISA Address (qualifier), control version and separators
- X12 trading partner Security and Authorization settings
- EDIFACT partner qualifier, enveloping and duplicate control settings
- EDIFACT syntax defaults
- Post Office defaults
- **Deleting defaults**
When a field is optional in Partner Manager, you can delete an individual default by setting that field to blank.

Deleting defaults

When a field is optional in Partner Manager, you can delete an individual default by setting that field to blank.

You can also delete all defaults by clicking the **Delete All Defaults** button that appears at the bottom of the screen. This button will be displayed only if default values have been previously saved.

Database utilities

The **Database Utilities** contains the following:

- Merge
- Purge
- [Merge](#)
- [Purge](#)

Merge

About this task

Note: If user security is enabled, the user must have the following privileges to run the merge: **Folder Maintenance, Create Trading Partner, Create Application Partner, Create Folders, Create Inbound and Outbound Links and Create Post Offices.**

Partner Manager enables you to merge one database to another database. For example, this feature enables you to merge selected objects in a source test database into a destination production database.

The source database is the database currently open in Partner Manager and specified in the Data Source Name (DSN) profile. The information you can merge into the destination database includes Trading Partners, Trade Links, and Post Offices.

Note: To verify the current profile information, including DSN information, select About Partner Manager from the Help menu.

To merge a database into another database

Procedure

1. From the Tools menu, select **Merge to Another DB**.
The Merge: Select Destination Database window is displayed.
2. Select an existing profile or DSN.
Note: If you have not entered a profile to select, then enter the User ID, Password, Database (if not Oracle), and Database Vendor.
3. Click Next.
4. Select an option from the Merge Option dialog box:
5. Click Begin Merge.
6. Follow the merge wizard instructions and click Finish.

Results

The wizard varies slightly different depending on the item you select to merge. A message appears upon successful completion.

Note: The next time you perform a database merge, Partner Manager keeps the most-recently selected destination database as the default destination database.

- [Details on merging trade links](#)
- [Post Office and Trading Partner merge details](#)

Details on merging trade links

The merge function is useful for transferring information from a test environment to a production environment.

The **Inbound Trade Link** and **Outbound Trade Link** merge functions for X12 and EDIFACT transfer all Trade Link data to another database.

The functions also merge dependent information, such as Folders, Trading Partners and Application Partners (From, To, and Acknowledgement Partners), Post Offices, Contacts, and any Standards needed (EDIFACT, X12, Functional IDs, and Transaction Sets). Carefully review the initial and final analysis messages to ensure the Trade Link merged as you expected.

- The merge functions do not overwrite existing information; information is appended to destination databases. If the destination database has information in it that is identical to the source database, that information is not included in the merge from the source database. This feature allows you to transfer test Trade Links and related information before implementing them.
- Because the merge functions never overwrite information, you can add information to a copy of a production database, test it, and then **merge the new information only** to the production database.
- Review the initial and final analysis messages carefully on the Select to Merge dialog box to insure the accuracy of the merging databases; this dialog box provides Merge Analysis information about what specifically is to be merged.
- The EDIFACT Trade Link merge does not have the ability to merge a trade link that exists in the destination database.

Post Office and Trading Partner merge details

- Partner Manager does not complete a Post Office merge if the destination database contains a Post Office with the same **Name** or **Nickname**.
- Trading Partners merge their associated Application Partner information in the destination database.

- If the Trading Partner exists in the destination database but the associated Application Partner does not, the Application Partner is merged in the destination database.
- Partner Manager also creates any folders or subfolders you have created in the source database in the destination database. If the folder name exists in the destination database but is a different folder type than the source, Partner Manager merges the folder data in the appropriate folder. For example, data in a source X12 internal folder named **Track** is not merged into the destination X12 external folder named **Track**. Instead, Partner Manager merges the data into the default destination X12 *internal* folder **Internal X12**.

Purge

About this task

Use the following instructions to purge traffic and/or Partner Manager Log data. To purge data, open the Purge/Archive Traffic Reports window as follows

To open the Purge/Archive Traffic Reports window

Procedure

1. Click the **Utilities** icon in the Partner Manager toolbar.
2. Select Database > **Purge** in the **Utilities** navigator to open the window.
 - [Purging traffic or transmission data](#)
 - [Archiving batch traffic](#)

Purging traffic or transmission data

About this task

To purge traffic or transmission data

Procedure

1. With the Purge/Archive Traffic Reports window open, select the **Purge (Permanent Delete) Current Tables** button.
2. Select the type of data to be permanently deleted.
 - To permanently delete traffic data, continue with the next step.
 - To permanently delete transmission data, select the **Purge Transmission Data** check box.
3. All records will be purged before the date displayed in the **Purge/Move All Before this Date** field. If you want to choose a different date than the one displayed, open the calendar by clicking the arrow to the right of the field and select a new date.
Note: All records dated *before* this date will be purged. The purge will also remove any control numbers used for duplicate interchange control number checking, so the utility will display a warning prior to the purge.
4. Click **Begin Purge/Archive**. A message box opens that informs you that the purge is permanent and asks if you want to continue.
5. Click Yes.

Archiving batch traffic

About this task

To archive batch traffic

Procedure

1. With the Purge/Archive Traffic Reports window open, click the **Configure** button located in the **Batch Traffic Archive** area of the window. The Message Manager Configuration window opens with the Traffic Batch Archiving tab view displayed.
2. Check the **Enable Batch Traffic Archiving** check box.
3. Enter the number of days that the archive is to be kept. You can enter any integer between 1 and 365 (inclusive).
If you want to move *all* traffic to the online archive, enter 0 in the **Days to Keep** field. If you do not want to do this, continue with the next step. If you enter 0 the **Days to Keep Warning** will open which asks you to confirm your selection.
4. Click Save to close the Message Manager Configuration window and return to the Purge/Archive Traffic Reports window.
The number of days that the archive is to be kept is displayed in the window.

Results

Enabling this option will also remove any control numbers used for duplicate interchange control number checking (at the point the traffic is archived, not at the point the option is enabled), so that the utility will display a warning prior to saving this setting.

Folders

The **Folder Utilities** facility enables you to organize trading partners in subfolders under the main Address Book **Internal** and **External** folders.

Partner Manager enables you to add, rename, move, and delete internal and external X12 subfolders.

You may not delete or rename main folders. When you delete a subfolder, it deletes all objects (trading and application partners, contacts, transaction sets, and so on) in the subfolder.

- [Adding a subfolder](#)
- [Renaming a subfolder](#)
- [Moving a subfolder](#)
- [Deleting a Subfolder](#)

Adding a subfolder

About this task

To add a subfolder to a main folder

Procedure

1. From the Tools menu, select Folder **Utilities**.
The Folder Utilities window opens.
2. Select a folder.
3. Click Add Sub Folder.
The Folder Maintenance dialog box opens.
4. Enter a name in the **Folder Name** field.
Note: The name can be a maximum of 32 characters.
5. Click Save.
The new subfolder appears in the **Folder Utilities** list and the **Address Book** navigator.

Renaming a subfolder

About this task

To rename a subfolder

Procedure

1. From the Tools menu, select Folder **Utilities**.
The Folder Utilities dialog box appears.
2. Select a subfolder.
3. Click Rename.
The Rename Folder dialog box appears.
- Note: The name can be a maximum of 32 characters.
4. Change the folder name in the **Folder Name** field.
5. Click Save.
The folder is renamed.

Moving a subfolder

About this task

Moving a subfolder moves all objects (trading and application partners, contacts, transaction sets, and so on) in the subfolder.

Note: If the subfolder does not contain any trading partners, it cannot be moved.

To move the contents of one subfolder to another

Procedure

1. From the Tools menu, select Folder **Utilities**.
The Folder Utilities dialog box appears.

2. Select a subfolder.
3. Click Move. The Move Folder dialog box appears.
4. Enter an existing subfolder name in the text box and click OK.
A message informs you that the move was successful.
5. Click OK.

Deleting a Subfolder

About this task

When you delete a subfolder, it deletes all objects (trading and application partners, contacts, transaction sets, and so on) in the subfolder.

To delete a subfolder

Procedure

1. From the Tools menu, select Folder **Utilities**.
The Folder Utilities dialog box appears.
2. Select a subfolder.
3. Click Delete and confirm the delete action when prompted.

Security

The Security utility contains the following:

- ["Security Groups"](#)
- ["Security Parameters"](#)
- ["User Maintenance"](#)
- ["Prior Version Password Remover"](#)
- [Changing passwords](#)
- [Security groups](#)
- [Security parameters](#)
- [User maintenance](#)
- [Prior version password remover](#)

Changing passwords

You can change the password of the current user. This option is only available if security is in force; it is disabled until a valid user logs into Partner Manager.

- [Preventing easily guessed passwords](#)
- [Changing your password](#)
- [Password synchronization](#)

Password synchronization changes your Partner Manager password on all Partner Manager databases if it is changed in any.

Preventing easily guessed passwords

Passwords containing the user name and the words *secret* and *password*, *pass* and *word* will not be accepted as valid passwords. For example, if the user name were Fred, a password of *123Fred123* would be rejected as it contains the user name. This check will only be performed on newly entered passwords. Passwords converted from prior versions will not be subject to this restriction.

Changing your password

About this task

The following steps describe how to change passwords

To change your password

Procedure

1. From the Tools menu, select Security > **Change Password**.
The **Partner Manager Change Password** dialog box opens.

2. Complete the fields in the dialog box. See "[Preventing Easily Guessed Passwords](#)" before changing a password.
3. If you want this password synchronized across all your Trading Manager database profile, check the box labelled **Run Password Synchronization After Save**. See [Password synchronization](#).
4. Click OK.
A message box opens that tells you that the password has been changed.

Password synchronization

Password synchronization changes your Partner Manager password on all Partner Manager databases if it is changed in any.

The databases can be from different vendors, for example, a test Access database and a production DB2 database. This synchronizes the Partner Manager password only, not the password associated with the database. Database passwords are controlled by the database vendor's tools, so they cannot be synchronized by Partner Manager.

Three conditions control this feature:

- Condition 1 - Database profiles must be defined (found on the Database Settings dialog). The passwords are changed in the databases defined in these profiles.
- Condition 2 - The user being changed must exist in those databases defined.
- Condition 3 - The Security Group Parameter labeled **Allow Password Synchronization** must be checked "on" for the current user running Partner Manager. This feature is "off" for new installations or upgrades. This parameter is found in the Utilities, Security Groups section labeled **General Access**.

If condition 1 or 3 (above) are false, the **Run Password Synchronization After Save** checkbox enabling this feature will not be visible.

Passwords can be changed in two places in Partner Manager:

- in User Edit, when a user is created or modified
- through the **Change Password** popup

If a password is changed in either of the two forms, check the checkbox labeled **Run Password Synchronization After Save** to begin this procedure. When a password gets changed and this checkbox is checked, the system will attempt to find that user in all selected Database profiles and then change the password to the value just set.

- [Security parameters that impact passwords](#)

There are additional security parameters that impact passwords, specifically, Minimum Password Length and Password Re-use history in Security Parameters and User Cannot Change Password found on the User record itself.

- [Password synchronization run-time errors/messages](#)

Potential run-time errors may appear during password synchronization.

Security parameters that impact passwords

There are additional security parameters that impact passwords, specifically, Minimum Password Length and Password Re-use history in Security Parameters and User Cannot Change Password found on the User record itself.

These security parameters are set by database, so the possibility exists that the Minimum Password Length, Re-Use History, or ability to change a password differs in the profiled databases. The synchronization routine checks these conditions in all databases against the newly changed password, and will disallow the synchronization if required. The failure reason will be displayed during the synchronization phase, for example "New password has an invalid length in Profile XYZ", or "Password not allowed in Profile XYZ because Re-Use-Rule violated".

In addition, the Security Parameter "Log All Security Maintenance" is honored. If the synchronization changes a password with that setting enabled, a Partner Manager Log record will be created in the same database being changed. An additional Log Record in the current database summarizing the synchronization phase will be written if logging is enabled ("User name syncd password to n profile").

The **PW Expiration Date** field in the Users table will be updated to reflect the new expiration date, and the **User Must Change Password On Next Login** flag will be cleared on all accounts updated across all selected databases.

Failure reasons will display in the list of profiles selected. Because failure reason text can be lengthy, clicking on a Profile will display the complete status of that profile in a text box below the list.

An account that is currently locked out on a database will still have its password changed, but it will not unlock that account.

Note: Database profiles are stored individually by PC, in the Windows Registry, so Profiles can differ based on where partner manager is running. Ensure your Database Profiles reflect their database instances correctly.

Password synchronization run-time errors/messages

Potential run-time errors may appear during password synchronization.

You may encounter the following during synchronization:

- Database Profiles allow you to omit the database password, thus making you enter it during entry to Partner Manager. You will be asked for that database password during the synchronization phase if that is the case. This is due to the fact that Partner Manager passwords are not the same as Database Passwords.
- Database Profiles can be invalid for numerous reasons (for example: if the database user/password is wrong, server has moved, or the database is a different Partner Manager version). Some of these errors can only be determined during a database connection attempt, and are reported back by the database. Such error messages will not have a standard format; they will vary by database vendor and database version. It is not uncommon to see popup messages during the synchronization phase if Profiles are invalid.
- As a database transaction (begin/commit) cannot be performed across different databases, it will be possible that some of the passwords synchronized and others did not, if an error was reported.
- It is possible that no passwords were synchronized if you, as a Partner Manager user, do not exist in the profiles defined, or if all of the profiles are invalid.

Security groups

About this task

Partner Manager enables you to define groups with security permission attributes and assign users to those groups. Assigned users log into Partner Manager.

Note: From the Help menu, select About Partner Manager to view the current **User ID** and **Security Group**.

Defining security groups enables you to configure default sets of permissions that you can then assign to one or more Partner Manager users. You can use the security profiles provided, or adjust the access selections to fit your needs.

You can add, edit, and delete security groups. You then define and add users to security groups.

Partner Manager provides three suggested levels of group permissions. You can use the following as a template for your security needs:

- **Administrator** - Provides full permission to security, purge and archive, EDI wizard, access, create, edit, and resend functions
- **Power User** - Provides **Administrator** permissions, except for security and purge/archive functions
- **Read Only** - Provides access-only permissions for Trading Partner, Application Partner, Post Offices, Inbound Trade Links, and Outbound Trade Links information
- [Adding a new security group](#)
- [Editing security group settings](#)
- [Deleting a security group](#)

Adding a new security group

About this task

Adding a security group defines permissions for users that belong to that security group. At least one user must be added to a new security group before the security group can be used.

To add a new security group

Procedure

1. From the Tools menu, select Security>Security Groups.
The Security Group Maintenance window appears.
2. In the **Group ID** field, enter the new security group name (up to 10 characters).
3. Press **Enter**.
A confirmation dialog box appears.
4. Click Yes to create the new security group.
5. (Optional) In the **Group Desc.** field, enter a description for the group.
6. Define the security group permissions by clicking a **Security Profiles** button: **Administrator**, **Power User**, or **Read Only**.
7. The default settings for the category are enabled.
Note: Click Clear All to clear all permissions currently enabled.
8. To customize the permissions, enable or disable the check boxes for access settings.
9. Click Save to create the group.

Editing security group settings

About this task

Existing security groups can be modified.

To edit security settings for an existing group

Procedure

1. From the Tools menu, select Security>Security Groups.
The Security Group Maintenance dialog box appears.
2. Select the group ID for the existing group from the **Group ID** drop-down list.
Edit the permissions for this security group.
3. Click Save.

Deleting a security group

About this task

All users assigned to the security group must be deleted prior to deleting the security group.

To delete an existing security group

Procedure

1. From the Tools menu, select Security > Security Groups.
The Security Group Maintenance dialog box appears.
2. Select the security group from the **Group ID** drop-down list.
3. Click Delete.
A delete confirmation dialog box appears.
4. Click Yes.

Security parameters

About this task

This section describes how to set the various Partner Manager security parameters using the Security Parameters window.

To set security parameters

Procedure

1. From the Tools menu, select Security > Security Parameters.
The Security Parameters window opens.
2. If you want to enforce use of a password that consists of a minimum number of characters, enable the **Enforce Minimum Password Length** check box.
The **Minimum Length** field will become active and a default value will be displayed in the field. You can change the default value. The system will then detect existing passwords that do not meet the minimum length requirement.

Note: If you leave the field blank (check box disabled) any password length, including zero, will be valid. (See "[Preventing Easily Guessed Passwords](#)" before selecting a password.)
3. If you want to force a user to be locked out after a number of invalid logon attempts, enable the **Enforce Maximum Invalid Logins** check box.
The **Maximum Invalid Attempts** field will become active and a default value will be displayed in the field.
 - You can change the default value. The system will then lock out any user entering invalid logons for the specified number of times.
 - When the **Enforce Maximum Invalid Logins** check box is enabled, the **Log Accounts Locked Out** check box is active (not grayed out). When enabled, this check box will cause accounts logged out data to be written to the Partner Manager Log File. If you do not want the accounts logged out to be logged, disable this check box.
4. If you want to log all security procedures in the Partner Manager log, enable the **Log All Security Maintenance** check box.
Note: The security procedures logged include: creating and maintaining user ids, security groups, security parameters, and changing password.
5. If you want to log all user logon activity in the Partner Manager log, enable the **Log All Logon Activity** check box.
When this option is selected the Partner Manager log will record all user logon and logoffs, as well as record whenever invalid logon attempts occur.
6. If you want to activate the Password History option, enable the **Enforce Password History** check box.
The By user, prevent the following number of passwords from being used field will be enabled.
 - Enter a value in the field. The default value of the field is 4, and must be a value greater than 1. When a user changes his or her password, the system will prevent the new password from being the same as the last number of passwords specified in this field.
7. Click Save.
A message box opens that informs you that Security Parameters have been successfully updated
8. Click Ok.
The message box closes and the parameters are saved.

User maintenance

About this task

User permissions are defined in the security group permissions. Selecting the **Group ID** on the **User Information** dialog box assigns the user to a specific security group.

A user definition consists of:

- User ID (for logon)
- User Name (optional description)
- Password
- Confirm Password
- Group ID
- Password options

You can add, edit, and delete users. The Change Password feature allows users to change their password.

- [Adding a user to a security group](#)
 - [Editing a user](#)
 - [Deleting a user](#)
 - [User login with security](#)
-

Adding a user to a security group

About this task

To create a user, enter his/her name, password, password expiration rules, and select the Security Group to define the user's permissions.

Note: From the Help menu, select About Partner Manager to view the current **User ID** and **Security Group**.

To add a user to a security group

Procedure

1. From the Tools menu, select Security>User Maintenance.
The Security Users window opens.
2. Click Add.
The User Maintenance: Add window opens.
3. Enter the **User ID** for the new user.
4. Enter a **Group ID**.
Note: The **User ID** and **Group ID** fields are required fields. All others are optional. Complete the following steps if desired. If not continue with step 10.
 5. In the **User Name** field, enter a description for the user.
 6. In the **Password** field, enter a password for the user. (See "[Preventing Easily Guessed Passwords](#)" before selecting a password.)
 7. If you entered an optional password, enter the password again in the **Confirm Password** field.
 8. Select a **Group ID** to assign this user to a security group.
 9. Select any desired password options.
 - If you want to force the user to change their password at the next logon, enable the **User Must Change Password at Next Logon** check box.
 - If you want to disable the user's ability to change passwords, enable the **User Cannot Change Password** check box.
 - If you want to disable the account, enable the **Account is Disabled** check box.
 - If you want to select a number of days for password expiration, click Password expires every: and enter a number of days in the **days** field.
10. Click Save.
You are returned to the Security Users dialog box and the new user is added to the list.

Editing a user

About this task

You can edit information about a user.

To edit an existing user

Procedure

1. From the Tools menu, select Security>User Maintenance.
The User Information dialog box appears.
2. Select a User ID from the User ID drop-down list.
3. You can edit the password for this user or select a different security group from the **Group ID** drop-down list.
4. Click Save.

Deleting a user

About this task

You can delete users. You must delete users before deleting the security group they belong to.

To delete an existing user

Note: Deleting the last user in Partner Manager disables all user and group security.

Procedure

1. From the Tools menu, select Security>User Maintenance.
The User Information dialog box appears.
2. Select a **User ID** from the **User ID** drop-down list.

3. Click the **Delete** button.
A confirmation dialog box appears.
4. Click Yes to confirm the delete of the user.

User login with security

About this task

After you configure security groups and add users, exit Partner Manager. When the Partner Manager application is restarted, the Partner Manager User Login dialog box appears.

To start Partner Manager with security

Procedure

1. Start the Partner Manager application.
The Partner Manager User Login dialog box appears.
2. Enter a valid **User ID**.
Note: The next step is valid only if you configured a Password with the User ID.
3. Enter a valid Password.
4. Click OK.

To log in as another user from within Partner Manager

Procedure

1. From the File menu, select Login as another User.
The **Partner Manager** Login dialog box appears.
2. Enter a valid **User ID**.
3. (Option) Enter a valid **Password**.
4. Click OK.

Prior version password remover

About this task

You can remove prior Trading Manager passwords from the Windows registry by using the Prior Version Password Remover.

To remove old passwords

Procedure

1. Choose the **Utilities** icon from the Partner Manager toolbar, then select Utilities > **Prior Version Password Remover** from the **Utilities** navigator.
The Prior Version Password Remover window opens.
2. Click Begin to start the Windows registry search for prior passwords.
Note: This process may take several minutes.
When the process completes, passwords are cleared.
3. Click Cancel.

EDI Wizard

The EDI Wizard creates type trees used by the Message Manager during the processing of your inbound and outbound data, therefore, it must be run before the Message Manager is built the first time. After you have defined your inbound and outbound trade links in your initial implementation, run the EDI Wizard to create the type trees.

Whenever a new transaction set or an existing transaction set for a new version of the standard is added, run the EDI Wizard again to update the definitions of the data to be recognized by the Message Manager.

- [EDI standard type trees](#)
- [Running the EDI Wizard](#)
- [Where the Type Trees are located](#)
- [When the EDI Wizard cannot locate a type tree](#)
- [When more than one type tree is found for a version](#)

EDI standard type trees

The EDI Wizard creates the following EDI standard type trees:

- **x12user.mtt** (X12 and EDIFACT): You can use the type tree in your application maps. If you have both X12 and EDIFACT data, two trees will be generated: **x12user.mtt** and **edifuser.mtt**. These type trees contain all the definitions for the EDI data processed by the Message Manager for all transaction sets for all versions of the standard required for the trade links that are defined.
- **x12mail.mtt** (X12) and **edifmail.mtt** (EDIFACT): Message Manager uses these type trees. They are referenced by a series of the Message Manager maps. When the EDI Wizard generates these type trees, they are placed in the same directory as the Message Manager maps and trees.
- **EDI Mail Trees:** The EDI Wizard will create both **x12mail.mtt** (X12) and **edifmail.mtt** (EDIFACT) trees if you have the trees for these standards. Message Manager uses these type trees. They are referenced by a series of Message Manager maps. When the EDI Wizard generates the type trees, it is placed in the same directory as the Message Manager maps and trees.
- **EDI User Trees:** You can use the type tree in your application maps. The EDI Wizard will create both X12(**x12user.mtt**) and EDIFACT (**edifuser.mtt**) trees if the 'Create User Trees' option is selected. This type tree contains all the definitions for the EDI data processed by the Message Manager for all transaction sets for all versions of the standard required for the trade links that are defined. These trees are created in the same directory as the Message Manager maps and trees.
Note: The EDI Wizard will not create TRADACOMS trees.

Running the EDI Wizard

About this task

The EDI Wizard creates type trees from Trade Link definitions.

To run the EDI Wizard

Procedure

1. Select EDI **Wizard** from the Wizard menu on the toolbar.
The Type Tree Wizard dialog box appears.
2. Click Next.
3. Click Next to begin to create the customized EDI type trees.
4. Follow the wizard instructions to create your type tree.

Where the Type Trees are located

About this task

The wizard prompts you to:

- Search the trading relationships defined in the Partner Manager.
- Enter the location of the type trees used to create the standard EDI type trees and searches for these trees.
- Enter the location and file name for the new type trees. When you first run the wizard, accept the default type tree names.

When prompted to specify a directory for new Message Manager Standard type trees, choose the directory where the Message Manager source maps and type trees are located. If you choose a different directory and attempt to create the type tree, a dialog box displays alerting you to this fact and asking you to confirm that you want to create this type tree in the specified directory.

If you have X12 trade links, the EDI Wizard will create the **x12mail.mtt** type tree in the specified directory.

If you have EDIFACT trade links, the EDI Wizard will create the **edifmail.mtt** type tree in the specified directory.

Note: if you are using a message version 90 (release 1), which contains the INVOICE and ORDERS messages, you must select the **edif90_1_ctrl.mtt** type tree when using the EDI Wizard to create the **edifmail** tree. Users who do not have message version 90 (release 1) messages need to use the **edifiso9735-4.mtt** when using the EDI Wizard to create the **edifmail** tree.

After the Wizard is finished, it displays a message that the customized type trees have been created.

After a new **x12mail.mtt** (X12) and/or **edifmail.mtt** (EDIFACT) type tree is created, you must redeploy both Trading Manager and Message Manager run maps. See Step 13 in Chapter 5 of the *Trading Manager Installation and User Guide* for instructions on this procedure.

When the EDI Wizard cannot locate a type tree

When the EDI Wizard cannot locate a type tree for a version that is needed, it displays a dialog box showing the type trees that it could not find. It is recommended that you cancel the wizard, install or re-locate the requested type trees, and then restart the wizard.

However, you can generate an incomplete type tree and add the other types manually. It is highly recommended that expert users perform the type tree modification only when a version of the requested type tree is not available. If you do select to continue to generate the type tree, a dialog box appears asking you to confirm creation of an incomplete type tree.

It should be noted that if you generate an incomplete type tree, you must manually add the types representing the transaction sets for the versions of the standard type trees that could not be found, before building the Message Manager maps.

When more than one type tree is found for a version

About this task

When the wizard locates more than one type tree for a particular version of the standard, it displays a dialog box listing those trees.

How to continue

Procedure

1. Select the appropriate type tree.
 2. Click Next.
When possible, use the original version of the standard trees that are installed for the type trees.
-

Database copy utility

The Database Copy Utility copies an existing Partner Manager Database to an empty (newly created) database. The newly created database can be a copy of **M4EC_Empty.mdb**, or a database created by ***inst.sql** scripts that ship with Partner Manager. This copy can be run with a transaction or without a transaction.

It should be noted that the default setting is within a transaction.

The **Run Copy Without Transaction Control** check box should only be enabled if the source database is very large and is causing **LogFile** full errors. If an error occurs when running without a transaction, no rollbacks occur. You must start over with an empty database.

The Database Copy Utility is accessed from the Start menu.

To start the Database Copy Utility

1. From the **Start** menu, click **Programs** IBM WebSphere Transformation Extender n.n.2_Trading Manager then **Database Copy Utility**.
The Partner Manager Copy Utility dialog box opens.
 - [Copying databases](#)
 - [Database scripts](#)
-

Copying databases

About this task

The Database Copy Utility enables you to copy an existing Partner Manager database to another empty database.

The empty database can be the **m4ec_empty.mdb** file or a newly created database created by the ***inst.sql** script files (both files are installed as part of Trading Manager).

This copy may be run without transaction control enabled if the source database is very large. If an error occurs when executing the database copy, and transaction control is not enabled, and no rollbacks occur; you must start over and create a new empty database (using the database drop, inst, and load scripts) or copy the **m4ec_empty.mdb** file (this file is valid for Microsoft Access database files only).

Database scripts

Run the ***drop**, ***inst**, and ***load** scripts on the database or for **Access** make another copy of **M4EC_Empty.mdb**.

- [*drop scripts](#)
 - [*inst scripts](#)
 - [*load scripts](#)
-

*drop scripts

These .sql files are installed in the *install_dir\tmgr_vn.n\pmgr\sql* directory:

- **db2dropnn.sql**
 - **oradropnn.sql**
 - **sqldropnn.sql**
 - **sybasedropnn.sql**
-

*inst scripts

These .sql files are installed in the *install_dir\tmgr_vn.n\pmgr\sql* directory:

- **db2instnn.sql**
- **orainstnn.sql**

- [sqlinstnn.sql](#)
- [sybaseinstnn.sql](#)

*load scripts

These .sql files are installed in the `install_dir \tmgr_vn.n\pmgr\sql\` directory:

- [db2loadnn.sql](#)
- [orainstnn.sql](#)
- [sqlinstnn.sql](#)
- [sybaseinstnn.sql](#)
- [sybaseinstnn_2.sql](#)

Managing traffic data

Partner Manager enables you to archive and purge the traffic record data in your Partner Manager database. (These records represent the Partner Manager e-commerce traffic.) Larger databases may slow down processing and result in unmanageable numbers of entries in each traffic report.

Partner Manager provides the following information management options that enable you to:

- Purge data from the Current traffic tables
- Purge data from the Online Archive traffic tables
- Archive data from the Current traffic tables to the Online Archive tables
- Archive data from the Current traffic tables to the zip file archives
- Archive data from the Online Archive tables to the zip file archives
- Restore data from a zip file archive

Partner Manager provides the following information management options:

| Option | Description |
|---|--|
| Purge (Permanent Delete) Current Tables | Permanently deletes traffic records before a specified date from the Partner manager database. These records cannot be restored. |
| Move to On-Line Archive | Places traffic records before a specified date in the Archive tables in the Partner Manager database. The records are then purged from the current Traffic tables. |
| Move to Current to Off-Line Zip File | Stores traffic records before a specified date in a zip file. These records are then purged from the current Traffic tables. |
| Move On-Line Archive to Off-Line Zip File | Stores traffic records before a specified date from the Archive tables in the Partner Manager database in a zip file. These records are then purged from the Archive tables. |
| Purge (Permanent Delete) of On-Line Archive | Permanently deletes traffic records before a specified date from the Archive tables in the Partner Manager database. These records cannot be restored. |
| Restore Zip File | Restore files archived in a compressed zip file using the Move to Current to Off-Line Zip File or Move On-Line Archive to Off-Line Zip File features. |

- [Purging current traffic or online archive data](#)
- [Archiving traffic data](#)

Purging current traffic or online archive data

About this task

This procedure permanently deletes traffic data from the **Current** traffic database tables.

Note: If the **Purge (Permanent Delete) Current Tables** or the **Purge (Permanent Delete) of Online Archives** purge option is enabled, a **Purge Transmission Data** check box is available. Enable the **Purge Transmission Data** check box to permanently delete the transmission archive of interchange data. This transmission archive folder is defined in the Directories > Archive field of the Server Directory Configuration Maintenance utility.

Before purging data, see the Check the Traffic Table Report Count section for information about traffic table counts.

To permanently delete current traffic information

Procedure

1. From the Tools menu, select Purge.
 2. The Purge/Archive Traffic Reports dialog box appears.
 3. Enable one of the following options:
 - Purge (Permanent Delete) Current Tables
 - Purge (Permanent Delete) of Online Archives
 4. Click the **Calendar** button to select a date.
 5. Click a date in the Calendar dialog box.
All traffic records before but not including this date are permanently deleted from the Partner Manager tables.
- Note: To quickly purge all records, add a future date.

6. Click Select.
 7. Click Begin Purge/Archive.
 8. Click Yes to confirm the purge action.
 9. As the traffic records are being deleted from the Partner Manager database, the current task displays in the **Purge Status** area. After the purge process is completed, a dialog box appears identifying the number of records purged.
 10. Click Cancel to close the Purge dialog box.
- [Check the traffic table report count](#)
-

Check the traffic table report count

About this task

This procedure enables you to check the number of traffic reports to determine whether you need to purge data. The Traffic Report Counts window shows the current number of reports by type using the actual database table name.

The following lists the database table name to the report name available from the **Reports** navigator. Any table name that includes the word **Archive** (such as **ECArtArchive**) shows the number of traffic reports in the traffic data archive.

| Database Table Name | View in Traffic Reports |
|---------------------|---|
| ECArt | Traffic Alert |
| ECICStatus | Data Type View |
| EDF_Status_IC | EDIFACT Interchanges |
| EDF_Status_FG | EDIFACT Functional Groups |
| EDF_Status_Msg | EDIFACT Messages |
| EDF_Status_SegErr | EDIFACT Segment Errors |
| EDF_Status_DEErr | EDIFACT Data Element Errors |
| X12DEError | X12 Data Element errors; if these errors exist, they are shown in the traffic reports segment error views. See Traffic Reports. |
| X12FGStatus | X12 Functional Groups |
| X12ICStatus | X12 Interchanges |
| X12SegError | X12 Segment errors; if these errors exist, they are shown in the Reports navigator under Transmissions. See Reports Navigator Overview. |
| X12TSSStatus | X12 Transactions |
| MapStatus | Transmissions |
| TRCICStatus | TRADACOMS interchanges |
| TRCDocHeader | TRADACOMS document header |
| TRCDocBody | TRADACOMS document body |
| TRCSegError | TRADACOMS document segment errors |
| TRCDError | TRADACOMS document data element errors |

To check the traffic table report count

Procedure

1. From the Tools menu, select Purge.
The Purge/Archive Traffic Reports dialog box appears.
 2. Click the **Traffic Counts** button.
The Traffic Report Counts window appears.
 3. Use the scroll bar to show the archived report information.
Click Copy to move the information to your Windows clipboard. You can paste this data into any Windows application that accepts clipboard text.
 4. Click OK to close the window.
-

Archiving traffic data

The following sections describe how to archive Partner Manager traffic data. Partner Manager enables you to:

- Archive data from the Current traffic tables to the Online Archive tables
 - Archive data from the Current traffic tables to the zip file archives
 - Archive data from the Online Archive tables to the zip file archives
 - [Archiving data from the current traffic tables to the online archive tables](#)
 - [Moving data from the current traffic tables to a zip file](#)
 - [Moving data from the online archive tables to a zip file](#)
 - [Restoring traffic data from a zip file](#)
-

Archiving data from the current traffic tables to the online archive tables

About this task

To move current traffic information to an online archive

Procedure

1. From the Tools menu, select Purge.
The Purge/Archive Traffic Reports dialog box appears.
 2. Enable **Move to On-Line Archive**.
 3. Enter a date in any valid date format in the **Purge/Move All Before This Date** field or click Calendar to select a date.
All traffic records before but not including this date are moved to the online archive tables.
 4. Click Begin Purge/Archive.
 5. Click Yes to confirm the move action.
As the traffic records are being archived into the archive tables and purged from the current tables, the current task is displayed in the **Purge Status**. After the process is completed, a dialog box appears identifying the number of records archived.
 6. Click Cancel to close the Purge dialog box.
-

Moving data from the current traffic tables to a zip file

About this task

To move current traffic information to a zip file

Procedure

1. From the Tools menu, select Purge.
The Purge/Archive Traffic Reports dialog box appears.
 2. Enable the **Move to Current to Off-Line Zip File** option button.
 3. Enter a date in any valid date format in the **Purge/Move All Before This Date** field or click Calendar to select a date.
All traffic records before but not including this date are moved to the online archive tables.
 4. Click Begin Purge/Archive.
 5. Click Yes to confirm.
The Select/Enter Zip File Destination dialog box appears.
 6. Enter a zip file archive file name and click Open.
7. The default name is **PartnerMgrCCYYMMDD.zip**, where CCYYMMDD represents the date you selected in Step 3.
 8. As the traffic records are being archived into the zip files and purged from the current tables, the current task is displayed in the **Purge Status**. After the process is completed, a dialog box appears identifying the number of records archived.
 9. Click Cancel to close the Purge dialog box.
-

Moving data from the online archive tables to a zip file

About this task

To move archived traffic information to a zip file

Procedure

1. From the Tools menu, select Purge.
 2. The Purge/Archive Traffic Reports dialog box appears.
 3. Enable **Move On-Line Archive to Off-Line Zip File**.
 4. Enter a date in any valid date format in the **Purge/Move All Before This Date** field or click Calendar to select a date.
All traffic records before but not including this date are moved to the online archive tables.
 5. Click Begin Purge/Archive.
 6. Click Yes to confirm.
The Select/Enter Zip File Destination dialog box appears.
 7. Enter a zip file archive file name and click Open.
The default name is **PartnerMgrCCYYMMDD.zip**, where CCYYMMDD represents the date you selected in Step 3.
 8. As the traffic records are being archived into the zip files and purged from the current tables, the current task is displayed in the **Purge Status**. After the process is completed, a dialog box appears identifying the number of records archived.
 9. Click Cancel to close the Purge dialog box.
-

Restoring traffic data from a zip file

About this task

To restore traffic data from a zip file

Procedure

1. From the Tools menu, select Purge.
The Purge/Archive Traffic Reports dialog box appears.
 2. Enable **Restore Zip File**.
 3. Enter the name of the zip file containing the traffic information to be restored. Click Find Zip to browse for the full file name and path of the zip file.
The default zip file name when files are archived is **PartnerMgrCCYYMMDD.zip**, where CCYYMMDD represents the date.
 4. Click Begin Restoration of Zip File to begin restoring the data.
The Purge/Archive Traffic Reports dialog box shows the records being restored. After the process is completed, a dialog box shows that the restore was completed successfully and the number of database tables that were processed.
 5. Click Yes to delete the zip file from which the traffic records were restored or click No to keep the file and return to the Purge/Archive Traffic Reports dialog box.
If you selected to delete the zip file, a dialog box appears confirming the deletion of that file.
 6. Click OK to return to the **Purge/Archive** Traffic Reports dialog box.
 7. Click Cancel to close the Purge dialog box.
-

Performance Tuning and Debugging

This documentation presented here is divided into two major parts. The first part provides a high level overview of the performance optimization techniques. The second part provides a checklist that you can use to guide you through the performance optimization process.

- [Overview](#)
 - [Tuning the IBM WebSphere Transformation Extender](#)
 - [Configuring Partner Manager](#)
 - [Debugging Trading Manager](#)
-

Overview

Throughout the discussions contained in this documentation, you should remember that Trading Manager is only one component in the B2B solution. The final performance depends on how well each of the various parts such as partner and application connectivity, individual data translation maps, routers and Trading Manager itself are configured.

- [Performance related considerations](#)
 - [Performance checklist](#)
-

Performance related considerations

Before continuing with the [Performance checklist](#) you should consider the following:

- CPU and memory requirements
 - transmission sizes
 - logical vs physical implementations
 - interface with non-IBM WebSphere Transformation Extender systems
 - resource pending configurations and how to avoid them
-
- [CPU and memory requirements](#)
 - [Analyze transmission sizes](#)
 - [Logical vs. physical implementations](#)
 - [Interfacing with non-IBM WebSphere Transformation Extender systems](#)
 - [Avoiding resource pending configurations](#)
-

CPU and memory requirements

Analyze the CPU and Memory requirements of other applications running on the same server as Message Manager.

Optimum performance is achieved when the competition for shared resources is minimized. Tuning Message Manager requires knowing how many resources are available to it. The IBM WebSphere Transformation Extender is, in an optimal configuration, a CPU intensive application that can degrade the performance of other applications (for example, databases) on the same server. The performance of the IBM WebSphere Transformation Extender can also be degraded by other applications on that server. For example, antivirus software that checks every file created for viruses will drastically hinder the performance of the IBM WebSphere Transformation Extender, and therefore Trading Manager and any application maps, due to the amount of files, such as work files, logs, and intermediate files, that are created during the normal course of events. Therefore, it is best to limit the number of other applications running on the IBM WebSphere Transformation Extender server.

Analyze transmission sizes

The size of the transmissions to be processed by Trading Manager determines the best configuration options for Message Manager. The default system, using workfiles in memory, is best for small to medium size transmissions. The cutoff transmission size depends on the amount of memory available to Message Manager on the server after accounting for other applications and services. See ["Analyzing Small Transmissions"](#) for additional information.

Logical vs. physical implementations

Consider the logical vs. physical implementations of Trading Manager. Trading Manager uses Post Offices to define the sources and destinations of data. They are logical sources and destinations that can match the same physical locations. If you have external processes to get the data from partners (either via a VAN or direct connect) you probably define a single Get Post Office in Trading Manager, the default location being:

`install_dir\tmgr_vn.n\mmgr\mail`.

The performance of the system can be increased if multiple GET Post Offices are defined (instead of one) even if they point to the same ...`\mail` folder, as long as the GET Post Office name can be easily derived from the input data. Message Manager handles trade link information by GET Post Office and defining multiple GET Post Offices spreads the trade link information, thus reducing overhead. For example, you can use a VAN for most of the inbound EDI traffic, but you can also have numerous other partners that connect via FTP. The best configuration is to create a single GET Post Office for the VAN transmissions and group all of the FTPs into another GET Post Office. There is no advantage in breaking up the transmissions retrieved from a VAN into multiple GET Post Offices since the overhead of breaking up the transmission, in most cases, overshadows the benefits.

Interfacing with non-IBM WebSphere Transformation Extender systems

Consider the following when interfacing non-IBM WebSphere Transformation Extender systems with the IBM WebSphere Transformation Extender. The Launcher triggers a map as soon as the trigger file is created. This can cause maps to execute before the trigger file is actually completed writing, especially with large files. To solve this problem, refer to "Solving IBM WebSphere Transformation Extender interface problems".

Avoiding resource pending configurations

Avoid configuring maps that can cause them to become "Resource Pending".

The Launcher prepares maps for execution through a two step process. The following is a simplified discussion of the process. For a detailed discussion, refer to the Launcher documentation.

Step 1: Allocates a thread for a map/trigger file combination. Map/trigger file combinations that fail to fulfill the requirements to successfully allocate a thread get placed on hold with a status of "Init Pending".

Step 2: Maps/trigger file combinations that have a thread allocated will then be executed unless they fail resource requirements. Such combinations will be placed on hold with a status of "Resource Pending".

The Launcher is configured with a finite number of threads to execute maps (either Message Manager maps and/or your own maps). If a map becomes **Resource Pending** (for example, if multiple instances of it append to a single file), it prevents other maps from using that thread. Multiple instances of the same map, except for the one that owns the resource in a Resource Pending state will, in effect, stall the Launcher's processing capabilities down to the one map that owns the resource. If this occurs, it is better to configure that map so that it becomes **Init Pending** before it can become **Resource Pending**, and therefore allow other maps to use the available thread.

Performance checklist

The following checklist is designed to help you configure Trading Manager for optimum performance. Although the steps in this checklist are presented in the order in which they should be executed, you should only implement those steps that are required for your particular system to achieve optimum performance.

- [Message Manager checklist](#)
- [External X12 partner checklist](#)
- [Inbound X12 trade link checklist](#)
- [Outbound X12 trade link checklist](#)
- [Small transmission size checklist](#)
- [Large transmission size checklist](#)

Message Manager checklist

About this task

The options described below are accessed from the **Message Manager Configuration Maintenance** window. Open this window as described in the section titled ["Opening the Message Manager Configuration Window"](#), then check the following options:

| | |
|-------|---|
| ----- | Option 1: Select the Use Stored Procedures option (under the Database tab). Turn ON if stored procedures are available. See "Use Stored Procedures" |
|-------|---|

| | |
|--|---|
| | option". |
| | Option 2: Optional X12/EDIFACT Validation (select Mapping Options tab). Turn OFF. See Optional X12/EDIFACT validation . |
| | Option 3: TSV Sort Option (select Database tab). Turn OFF. This option is needed only when using backend databases that sort rows without regard to upper or lower case. See " TSV Sort Option ". |
| | Option 4: TA1 Control option (select X12 Acknowledgements tab). Turn OFF if your partners are not capable of accepting and processing TA1 segments. See " TA1 Control Option ". |
| | Option 5: Store 997 Content option (select X12 Acknowledgements tab). Turn OFF if you do not need to view inbound and outbound 997 transactions. See " Store 997 Content Option " |
| | Option 6: Ignore Missing Trade Links option (select X12 Acknowledgements tab). Turn ON unless you want to post-process your partners 997s. See " Ignore Missing Trade Links Option ". |
| | Option 7: Inbound Transmissions into Trading Manager (Archive Options tab) Turn ON unless you already have a process in place to archive all of the transmissions received from your partners. See " Inbound Transmissions into Trading Manager ". |
| | Option 8: Outbound Transmissions into Trading Manager (Archive Options tab) . Turn OFF. See " Outbound Transmissions into Trading Manager ". |
| | Option 9: Inbound Interchange (Archive Options tab) . Turn OFF if testing is complete and the Edit/Resend feature is not needed. See " Inbound Interchange ". |
| | Option 10: Outbound Interchange before re-enveloping (Archive Options tab) . Turn OFF if testing is complete and the Edit/Resend feature is not needed. See " Outbound Interchange before re-enveloping ". |
| | Option 11: Outbound Interchange after re-enveloping (Archive Options tab) . Turn OFF if testing is complete and the Edit/Resend feature is not needed. See " Outbound Interchange after re-enveloping ". |
| | Option 12: Outbound Transmissions from Trading Manager (Archive Options tab) . Turn ON unless you have a process in place to archive all of the transmissions sent to your partners. See " Outbound Transmissions from Trading Manager ". |

External X12 partner checklist

About this task

The options described below are accessed from the Address Book for External X12 window. Open this window as described in "[Options for External X12 Partners](#)", then check the following options:

| | |
|--|--|
| | Option 1: Duplicate Control (Trading Partner tab) . Turn OFF if duplicate checking is not necessary. See " Duplicate Control Option ". |
| | Option 2: Authorization and Security Validation (Trading Partner tab). Turn OFF. This option is rarely used. See " Authorization and Security Validation ". |

Inbound X12 trade link checklist

About this task

The options described below are accessed from the X12 Inbound Trade Links window. Open this window as described in "[Options for Inbound X12 Trade Links](#)", then check the following options:

| | |
|--|---|
| | Option 1: Routing (Application Post Offices/Routing tab). Turn ON if the criteria described in " Routing Options (Inbound) " is met. |
| | Option 2: Acknowledgements (Acknowledgements/Validate By tab) . Turn OFF. See " Acknowledgments Option ". |
| | Option 3: HIPAA Validation Types option (HIPAA Validation Types tab). Turn OFF if you are using the Packs for HIPAA EDI elsewhere in your system. See " HIPAA Validation Option ". |

Outbound X12 trade link checklist

About this task

The following options are configured from the "[Opening the X12 Outbound Trade Links Window](#)" with the Application tab view displayed.

| | |
|--|---|
| | Option 1: Routing options. Turn ON for no X12 validation, or OFF for validation. See " Routing Options (Outbound) ". |
| | Option 2: Outbound 997s . Should always be defined as Route Only. See " Outbound 997's ". |

Small transmission size checklist

About this task

Options 1 and 2 in the following checklist are configured using the Integration Flow Designer. See the Integration Flow Designer documentation for more information. Options 3 and 4 are configured using the "[Opening the Message Manager Configuration Window](#)".

| | |
|--|--|
| | Option 1: In StampAndSort change output cards two and three from File to Sink . This affects both inbound and outbound X12 files. See " Analyzing Small Transmissions ". |
| | Option 2: In sendx12 change output cards two and three from File to Sink. This affects outbound X12 files. See " Analyzing Small Transmissions ". |

| | |
|-------|--|
| ===== | Option 3: Set the Work File options (Optimization tab). Set to -WM . See " Work File Option ". |
| ===== | Option 4: Set the X12 Outbound PO Packaging option (Optimization tab). Select the One File per Put PO button if you generate small batches of outbound data. See " X12 Outbound PO Packaging Option ". |

Large transmission size checklist

About this task

Option 1 in the following checklist is configured using the Integration Flow Designer. See the Integration Flow Designer documentation for more information. Options 2 and 5 are configured using the ["Opening the Message Manager Configuration Window"](#).

| | |
|-------|--|
| ===== | Option 1: Change workspace settings for stampandsortfile , sortx12 and sendx12 . This option is set using the msg_mgr.msd opened in the <i>Integration Flow Designer</i> . See " "StampandSort and sendx12" ". |
| ===== | Option 2: Set the Maximum X12 FG Validation Group option (Optimization tab). See " "Maximum FG Validation Size" ". |
| ===== | Option 3: Set Work File options (Optimization tab). Set to -WU . See " "Work File Option" ". |
| ===== | Option 4: Set the Maximum Reject File Size option (Mapping Options tab). See " "Select Maximum Reject File Size" ". |
| ===== | Option 5: Set the X12 Outbound PO Packaging option (Optimization tab). Select the One File per Put PO per Interchange button if you generate large batches of outbound data. See " "X12 Outbound PO Packaging Option" ". |

Tuning the IBM WebSphere Transformation Extender

This information presented here provides information about those IBM WebSphere Transformation Extender tuning procedures that may be required in order to achieve optimum performance from your Trading Manager application.

Specific information related to IBM WebSphere Transformation Extender tuning is contained in the IBM WebSphere Transformation Extender Performance documentation and in the Launcher documentation. You should familiarize yourself with the information before continuing with these procedures.

Note: When changing values in the **dtx.ini** file, some of the parameters mentioned here are shipped as a "remark". Any parameter beginning with a semi-colon is a remark and is ignored. When changing these parameters, make sure you *remove* the semi-colon to activate it.

- [Turning on the Launcher trace file](#)
- [Updating the max threads parameter](#)
- [Solving IBM Transformation Extender interface problems](#)
- [Running multiple Launchers](#)

Turning on the Launcher trace file

Turn on the Launcher trace file in the **dtx.ini** file. This option facilitates debugging in both Trading Manager and the application maps. The option for performing this task is:

LauncherLog=ewsc

See the IBM Transformation Extender Launcher documentation for additional information.

Note: Turning on trace will significantly slow down the Launcher process. Once a problem has been solved, remember to turn off trace by adding a "remark" (semi-colon) in front of the parameter in the **dtx.ini** file, and recycling the Launcher service.

Updating the max threads parameter

The Max Threads parameter is updated in the **dtx.ini** file. By default the IBM WebSphere Transformation Extender is configured to have a maximum number of threads of 20 . This might be too high for platforms with one or two CPUs. Low end Windows platforms with only one CPU will do best with an approximate setting of 5 but it depends on what else is loaded on that server. Perform benchmarks with a range of settings to determine the best setting. The option is:

MaxThreads=20

For specific information on the Max Threads parameter, as well as all other Launcher performance parameters, refer to the IBM WebSphere Transformation Extender Performance documentation. Also see the IBM WebSphere Transformation Extender Launcher documentation.

Solving IBM Transformation Extender interface problems

There are two ways to solve the problems that occur when interfacing non-IBM Transformation Extender systems a with IBM Transformation Extender systems:

- Create files with a **.tmp** extension and rename the file when done writing. The IBM Transformation Extender with Launcher ignores any file with a **.tmp** extension and will process the file after it is renamed. This is the technique used by the IBM Transformation Extender internally, and it is called a "shadow file".
- Create two files: first the actual data file and then a one-byte trigger file. This is a popular option when FTPing into a system running under the Launcher, there has to be a relationship in the file names between the trigger file and the data file, so that the Launcher can derive the name of the data file from the trigger file name. The triggered map has to have the trigger file card before the data file card and the data file card should not be a trigger, only the trigger card should be a trigger.

Running multiple Launchers

If you are running UNIX, you have probably created multiple instances of the Launcher and segregated your application maps across them. Trading Manager is distributed to run in a single instance and even with the new multi-instance capabilities of the Launcher, the benefits of breaking Message Manager into multiple instances will only be obtained in the most extreme cases. Segregating application maps, however, into a separate system from Message Manager is still a good way to control and improve maintainability, performance, and throughput.

Configuring Partner Manager

This section describes how to change the configuration settings Partner Manager in order to achieve optimum performance in the Trading Manager application.

These procedures are, for the most part, detailed descriptions of the checklist items provided in the ["Overview and Performance Checklist"](#).

- [Overview](#)
- [Configuring Message Manager](#)
- [Options for external X12 partners](#)
- [Options for inbound X12 trade links](#)
- [Options for outbound X12 trade links](#)
- [Analyzing small transmissions](#)
- [Analyzing large transmissions](#)

Overview

Trading Manager provides a rich set of functionality that you may, or may not require. It contains functions may be turned on or off as needed. Functions such as:

- extended X12 validation
- TA1 generation and routing
- several layers of archiving, etc.

All of these require processing that translates into wasted overhead if the functionality is not used.

In the sections that follow, turn OFF any functionality that you do not need, and turn on functionality that you can use.

Configuring Message Manager

The Message Manager options described here are all accessed using the Message Manager Configuration Maintenance window. Access the window as described in the following section, then refer to the following sections for option details.

- [Opening the Message Manager Configuration Window](#)

Opening the Message Manager Configuration Window

About this task

To access Message Manager configuration options, first open the Message Manager Configuration Maintenance window:

Procedure

1. Start Partner Manager.
2. Select Utilities from the Partner Manager toolbar.
3. Expand **Configuration** in the **Utilities** navigator and select Message Manager Config.

Results

The Message Manager Configuration window opens:

- [Use stored procedures option](#)
- [TSV sort option](#)
- [Optional X12/EDIFACT validation](#)
- [TA1 control option](#)
- [Store 997 content option](#)
- [Ignore missing trade links option](#)
- [Archive options](#)

Use stored procedures option

About this task

By default, Message Manager uses in-line SQL statements to generate thread ids and control numbers. Trading Manager provides stored procedures for those databases that support it. If these stored procedures are available they should be used.

To set this option:

Procedure

1. Perform the steps provided in "[Opening the Message Manager Configuration Window](#)".
 2. Select the Database tab view.
 3. Turn the option ON by checking the **Use Stored Procedures** checkbox.
-

TSV sort option

About this task

This option impacts export processing performance. It forces a resorting of the TSV (Trading Manager files containing partner and transaction information) during the export so that the Trading Manager maps can take advantage of the SEARCHUP function. If the sort order of your database matches the sort order expected by the SEARCHUP function, you do not need to use the TSV Sort Option. The sort order expected by the SEARCHUP function depends on the operating system running Message Manager. Refer to the SEARCHUP documentation.

To set this option:

Procedure

1. Perform the steps provided in "[Opening the Message Manager Configuration Window](#)".
 2. Select the Database tab view.
 3. Disable the option by unchecking the **Sort TSV Files During Extract** checkbox, if you do not need this functionality.
-

Optional X12/EDIFACT validation

About this task

Although it is important that this option be enabled when testing with a new partner, after testing is concluded it is not necessary for it to remain in that state. For this reason, it is recommended that this option be turned OFF.

To set this option:

Procedure

1. Perform the steps provided in "[Opening the Message Manager Configuration Window](#)".
 2. Select the **Mapping Options** tab view.
 3. Uncheck the checkbox in the **Optional X12/EDIFACT Validation** area of the window.
-

TA1 control option

About this task

This option should be turned OFF if your partners are not capable of accepting and processing TA1 segments.

To set this option:

Procedure

1. Perform the steps provided in "[Opening the Message Manager Configuration Window](#)".
 2. Select the **Acknowledgements** tab view.
 3. Check the **No TA1's** button.
-

Store 997 content option

About this task

This option should be turned OFF unless you have the explicit need to view inbound and outbound 997 transactions online.

To set this option:

Procedure

1. Perform the steps provided in "[Opening the Message Manager Configuration Window](#)".
 2. Select the **Acknowledgements** tab view.
 3. Uncheck the **Store 997 Content in Database**.
-

Ignore missing trade links option

About this task

This option should be turned ON unless you are planning on post-processing your partner's X12 997 and EDIFACT CONTRL messages. To save memory, you should also avoid creating tradelinks for these message types if you can safely ignore them. Note, however, that not creating tradelinks for these messages may result in an inaccurate **Overdue Acknowledgment Reporting**.

To set this option:

Procedure

1. Perform the steps provided in "[Opening the Message Manager Configuration Window](#)".
 2. Select the **Acknowledgements** tab view.
 3. Check the **Ignore Missing Trade Links for Inbound 997/CONTRL**.
-

Archive options

About this task

Archive Options: Message Manager archives data at different times and some of the options might be redundant:

To set these options:

Procedure

1. Perform the steps provided in "[Opening the Message Manager Configuration Window](#)".
2. Select the **Archive** tab view.
3. Make selections to the checkboxes.

Inbound Transmissions into Trading Manager

About this task

This option should always be ON (checkbox checked) unless you already have a process in place to archive all the transmissions received from your partners elsewhere.

Outbound Transmissions into Trading Manager

About this task

Turn this option ON (checkbox checked) if you want to archive all outbound transmissions, i.e. files created by your "Application to EDI" conversion maps using data from your internal systems. These files can usually be recreated from your internal applications and, therefore, it is recommended that this option be turned OFF.

Inbound Interchange

About this task

When enabled (checkbox checked) a text file is created for each inbound interchange and the **Edit/Resend** feature in Partner Manager is enabled. Although this feature may be useful during testing it can be turned OFF if the **Edit/Resend** feature is not used or not allowed.

Outbound Interchange before re-enveloping

About this task

When enabled (checkbox checked) a text file is created for each outbound interchange and the **Edit/Resend** feature in Partner Manager is enabled. Although this feature may be useful during testing it can be turned OFF if the **Edit/Resend** feature is not used or not allowed.

Outbound Interchange after re-enveloping

About this task

When enabled (checkbox checked) a text file is created for each outbound interchange and the **Edit/Resend** feature in Partner Manager is enabled. Although this feature may be useful during testing it can be turned OFF if the **Edit/Resend** feature is not used or not allowed.

Outbound Transmissions from Trading Manager

About this task

This option should always be ON unless you already have a process in place to archive all the transmissions sent to your partners elsewhere.

Options for external X12 partners

Options for each External X12 Partner in the Address Book are accessed as follows. These options are set using the Edit X12 Trading Partner in External Folder X12 window.

- [Opening the Edit X12 Trading Partner Window](#)
-

Opening the Edit X12 Trading Partner Window

About this task

Refer to the following steps to open the window:

Procedure

1. Start Partner Manager.
2. Select Address Book from the Partner Manager toolbar.
The Address Book for External X12 window opens.
3. Select a Trading Partner from the listbox.
The Edit X12 Trading Partner In External X12 window opens with the Trading Partner tab view displayed.

Results

The following sections describe how to set options to maximize Trading Manager performance.

- [Duplicate control option](#)
 - [Authorization and security validation](#)
-

Duplicate control option

About this task

When the Edit X12 Trading Partner in External X12 window opens, the Trading Partner tab is displayed. If this is not the case, click the Trading Partner tab.

To set this option:

Procedure

1. Perform the steps provided in the ["Opening the Edit X12 Trading Partner Window"](#).
 2. Select the **Trading Partner** tab view.
 3. Note the condition of the **Duplicate Control** checkboxes and set accordingly:
Turn the option ON (checkboxes checked) only if it is required for the partner. Most partners have internal controls that make duplicate checking unnecessary.
-

Authorization and security validation

About this task

The **Validate Authorization** and the **Validate Security** options should ordinarily be turned OFF as these options are rarely used.

To set these options:

Procedure

1. Perform the steps provided in the ["Opening the Edit X12 Trading Partner Window"](#).
2. Select the **Authorization/Security** tab view.

3. Ensure that the **Validation** check boxes are turned OFF.

Options for inbound X12 trade links

Options for each Inbound X12 Trade Link are accessed as follows.

- [Opening the X12 Inbound Trade Links Window](#)

Opening the X12 Inbound Trade Links Window

About this task

Open the window as follows:

Procedure

1. Start Partner Manager.
2. Select Trade Links from the Partner Manager toolbar.
3. Make certain that **Inbound** is expanded in the navigator and select Inbound > **X12**.
The **X12 Inbound Trade Links** window opens. Select a trade link to modify using the **Search** buttons at the top of the screen.
4. Click the Applications tab.
 - [Routing options \(inbound\)](#)
 - [Acknowledgments option](#)
 - [HIPAA validation option](#)

Routing options (inbound)

About this task

This option allows you to use Trading Manager to route EDI data without performing data validation.

Acknowledgments, if configured, report transactions with the **Routing** option as valid.

In deciding whether or not to turn this option ON or OFF, remember the following points:

- If you have an external system in place before Trading Manager that performs validation (for example, the IBM Healthcare Packs) there is no need to perform any additional validation in Trading Manager. Therefore, turn this option ON to improve performance.
- This option can also be turned ON for those transactions that are known to be good (inbound 997s), or when you trust the partner and have some error detection/reporting process in place in your **EDI to Application** conversion maps.

To set this option:

Procedure

1. Perform the steps provided in "[Opening the X12 Inbound Trade Links Window](#)" to open the window.
2. Select the **Post Offices Routing** tab view.
3. Enable or disable the **Route Only; No Validation** checkbox and set according to your individual requirements.

Acknowledgments option

About this task

Most partners require acknowledgments to be returned. There may, however, be select cases in which they are not needed. If this is the case, proceed as follows:

Make certain that you check with each partner before turning off acknowledgements.

To change this option:

Procedure

1. Perform the steps provided in "[Opening the X12 Inbound Trade Links Window](#)" to open the window.
2. Select the **Acknowledgements/Validate By** tab view.
3. In the **Ack Level** field, select No Acknowledgement from the drop down menu.

HIPAA validation option

To change the HIPAA validation options:

1. Perform the steps provided in [Opening the X12 Inbound Trade Links Window](#).
2. HIPAA validation settings are only available when a HIPAA version is selected. Therefore, you should check only those settings that are required by your installation.

Note: If you are using the IBM WebSphere Transformation Extender Pack for HIPAA EDI as the pre-processor for Trading Manager, the validation settings here will duplicate the functionality of that pack, so enabling these settings are not required.

Options for outbound X12 trade links

The options discussed in this section are configured for each outbound X12 trade link. The options are accessed from the **X12** Outbound Trade Links window.

- [Opening the X12 Outbound Trade Links Window](#)

Opening the X12 Outbound Trade Links Window

Procedure

1. Start Partner Manager.
2. Select Trade Links from the Partner Manager toolbar.
3. Make certain that **Outbound** is expanded in the navigator and select Outbound > **X12**.
The **X12 Outbound Trade Links** dialog opens.
4. Click the Applications tab.
 - [Routing options \(outbound\)](#)
 - [Outbound 997's](#)

Routing options (outbound)

About this task

This option allows you to use Trading Manager to route EDI data without performing data validation.

By default, Trading Manager validates each outbound transaction and guarantees that the partner always receives valid X12 data. Once a system is in production, however, the likelihood of it creating invalid data is minimal, and a partner's negative acknowledgement would highlight when it occurs.

You can increase the performance of Trading Manager by turning ON the **Route Only; No Validation** option for outbound transactions with minimal risk. To further minimize the risk, since this option applies by outbound Functional Group, it can be used only for those transactions with which you feel comfortable.

To set this option:

Procedure

1. Perform the steps provided in "[Opening the X12 Outbound Trade Links Window](#)" to open the window.
2. Select the **Post Offices Routing** tab view.
3. Turn ON or OFF the **Route Only; No Validation** checkbox and according to your individual requirements. In most cases, this checkbox will be ON (checked).

Outbound 997's

Outbound 997's should always be defined as Route Only (**Route Only; No Validation** checkbox checked).

Analyzing small transmissions

In order to maximize Trading Manager performance, it is important that you analyze the size of your:

- Transmission
- Interchange
- Functional Group
- Transaction.

Servers that have large amounts of memory available but are used to process small transmissions, and are used primarily for Message Manager can further increase Trading Manager performance by:

- In **stampandsortfile** change output cards #2 and #4 from:
 - **File**
to
 - **Sink** This affects inbound and outbound files.
- In **sendx12** change output cards #2 and #3 from:
 - **File**
to
 - **Sink** This affects outbound X12 files.

The procedures required to change these options are described in the following sections.

- [Changing stampandsort options](#)
 - [Changing sendx12 options](#)
 - [Work file option for small transmissions](#)
-

Changing stampandsort options

About this task

To perform this procedure follow these steps:

Procedure

1. Open the **msg_mgr.msd** file located in: *install_dir\tmgr_vn.n\mmgr*
This file opens in the Integration Flow Designer. For more information, see the Integration Flow Designer documentation.
 2. In **stampandsortfile** change output cards #2 and #4 from **File** to **Sink**.
-

Changing sendx12 options

Procedure

1. Open the **msg_mgr.msd** file located in: *install_dir\tmgr_vn.n\mmgr*
This file opens in the Integration Flow Designer. For more information, see the Integration Flow Designer documentation.
 2. In **sendx12** change output cards #2 and #3 from **File** to **Sink**.
-

Work file option for small transmissions

About this task

To set this option for small transmissions:

Procedure

1. Perform the steps provided in the "[Opening the Message Manager Configuration Window](#)".
 2. Select the Optimization tab view.
 3. Select -WM from the **Work File Options** drop-down menu.
-

Analyzing large transmissions

About this task

Servers processing large transmissions and/or not having much memory available will have to configure Message Manager so that the amount of memory used for processing is reduced.

With one exception, the options described here are accessed through the Message Manager Configuration Maintenance window. Access the window as described in "[Opening the Message Manager Configuration Window](#)".

Open the window and then refer to the following sections for option details.

- [stampandsort and sendx12](#)
- [Maximum FG validation size](#)

- [Work file option](#)
 - [Select maximum reject file size](#)
 - [X12 outbound PO packaging option](#)
-

stampandsort and sendx12

About this task

For very large transmissions, Trading Manager performance can be improved by changing the work space settings for **stampandsortfile** (affects both inbound and outbound), **sortx12** (affects inbound X12), and **sendx12** (affects outbound X12).

To perform this procedure follow these steps:

Procedure

1. Open the **msg_mgr.msd** file located in: *install_dir\tmgr_vn.n\mmgr*
This file opens in the Integration Flow Designer. For more information, see the *Integration Flow Designer Reference Guide*.
 2. Change the entry in the **WorkSpace** field for **stampandsortfile**, **sortx12** and **sendx12** from **Memory** to **File**.
-

Maximum FG validation size

About this task

To select this option:

Procedure

1. Perform the steps provided in the "[Opening the Message Manager Configuration Window](#)".
 2. Select the Optimization tab view.
 3. Specify the maximum size of a functional group to validate in a single pass. Enter this value in the **Maximum Validation Size** field.
The value is in Megabytes. Start with a value that reflects the average size of your functional groups or ¼ of the RAM memory available (whichever is less) and decrease as needed.
-

Work file option

About this task

To select this option:

Procedure

1. Perform the steps provided in the "[Opening the Message Manager Configuration Window](#)".
2. Select the Optimization tab view.
3. In the **Work Options** field, select -WU from the **Work File Options** drop down menu.

Results

This will reduce the amount of memory used by the Message Manager system in general.

Select maximum reject file size

About this task

Under mapping options, select the maximum size of the reject file to be created by the system.:

To select this option:

Procedure

1. Perform the steps provided in the "[Opening the Message Manager Configuration Window](#)".
2. Select the **Mapping Options** tab view.
3. Enter the desired value in the **Maximum Reject File Size** field.

4. If you are processing large X12 transactions you should define a **Max Reject Size** limit to prevent the server from becoming overloaded when generating reject files. A size of 1-5 Kb is adequate to provide accurate information about the transaction in error.

X12 outbound PO packaging option

About this task

To select this option:

Procedure

1. Perform the steps provided in the "[Opening the Message Manager Configuration Window](#)".
2. Select the Optimization tab view
3. Select the **One File per Put PO per Interchange** button for each Interchange in the mapping options screen.

Debugging Trading Manager

These topics describe how to perform debugging operations on the Trading Manager system. It describes:

- ["Deployment Errors and Failures"](#)
- ["Startup Errors"](#)
- ["Running Message Manager"](#)
- [Deployment errors and failures](#)
- [Startup errors](#)
- [Running Message Manager](#)

Deployment errors and failures

It is important to remember that both **MessageManager** and **RunMaps** have to be deployed, one after the other. Deploying **MessageManager** will not include the **RunMaps**. Failing to deploy RunMaps will cause export run maps to fail with condition code 3 (map not available).

The **msg_mgr.msd** provides three different deployment definitions for MessageManager and one for RunMaps, all preconfigured for a default Windows platform. The following should be checked:

| Check | Notes |
|---|---|
| Build and transfer maps | This should display the mmgr folder on the server that is going to run Message Manager. If it is not correct, the global search/replace failed and has to be redone. Showing a Windows path when the server is a UNIX variant is an indication the global search/replace failed or was not done at all |
| Generate and transfer Launcher Control File | Must be configured to point to the systems folder on the server and must be done manually for all deployment scripts that have it check marked |
| Transfer Additional Files | Must be configured to point to the mmgr folder on the server and must be done manually for all deployment scripts that have it check marked. You have to press ENTER for the path to take, simply clicking OK after editing a path will result in the edited path being lost. This option is only necessary if transferring files to a server or different Windows drive. For example, if the mmgr.mrn file is being transferred to a UNIX server. If the design and runtime installs are the same, (deploying and running on the same Windows machine, this option can be left unchecked). |

Startup errors

Startup is completed successfully if the export process manages to build all the **tsv** files the system needs in the **mmgr\share** directory. Typical errors are:

| Error condition | Reason |
|---|--|
| Export runs fine but there are no files in the mmgr\share folder. The tsv files are then found in the mmgr folder, some of them preceded with the string %export_dir%. | This means that the user failed to properly configure the Launcher Resource Registry (mrc and/or mrn files) and/or the Launcher Administration tool. The Launcher Administration tool has to point to a user-created mrc file that in turn has to point to the mmgr.mrn file provided with CM |
| Export runs but fails with "Source Not Available". | The back end database is not properly configured, usually an error in the startup.mdq file. Adding "-t" to the database card will generate a database trace that can be used to determine the problem. Both MessageManager and RunMaps have to be re-deployed once the mdq file has been corrected. |
| Export runs, but creates a reject file indicating all run maps failed with return code 3. | Run Maps have not been deployed. |
| Export runs, but creates a reject file indicating some or all run maps failed with return code 12 or similar. | Run Maps have not been deployed after fixing the update.mdq file, or the database creation failed. |

| Error condition | Reason |
|--|--|
| Export runs, but creates a reject file indicating that a dll/so is either not found or the wrong version. | Message Manager requires two dll/so's: cmsvc and cmsvu . They have to be deployed in binary mode and their properties have to be defined as executables. On Windows the dlls have to be in the mmgr folder, on UNIX the so's have to be in the DSTX libs folder. |
| Export runs, but creates a reject file indicating HIPAA components are missing. | If you select HIPAA validation, the export process verifies that the components required by Trading Manager from the HIPAA Pack have been deployed. Two maps from the HIPAA Pack (x4010hipaasegmentdataaudit.mmc and x12type6qualifier.mmc) have to be deployed into the mmgr folder on the server. Additionally, one file (hipaa_x12_type_6_value.dat) has to be deployed into the mmgr\share folder on the server. See Validating HIPAA X12 Data the IBM WebSphere Transformation Extender Pack for HIPAA EDI documentation. for more information. |
| Launcher starts up fine but nothing is running, the Management Console is blank. | The Launcher did not find any msl Launcher Control file. Either Trading Manager was not deployed or the location in the deployment definition is incorrect. Redeploy and review the MSEDploy.TXT file to verify where the maps and *.msl file are being deployed. |
| Launcher startup, customer maps are running, but files are piling up in the mmgr\mail folder. | See previous error condition. |

Running Message Manager

Trading Manager has been enhanced to create logs with more detailed information about failures while running Message Manager. Therefore it is critical that all reject files be examined to get additional information about errors.

| Error condition | Reason |
|---|--|
| System runs slow, high CPU utilization. | Could be caused by Max Threads parameter being set too high in the dtx.ini file. A lower setting allows the server to perform more useful work and less swapping between threads. |
| Random map failures due to not enough memory available | Could be caused by Max Threads parameter being set too high in the dtx.ini file. See the section titled " Analyzing Large Transmissions " for more memory reduction options. |
| Files do not show up in the PUT Post Office | Usually occurs when the path is not correctly defined in Partner Manager. A reject file and the Partner Manager Traffic Log provide specific error information. At times a reject file is not created; in that case the files might be found in the root of the server. The Partner Manager Traffic report will show the location where the files are written |
| Reject files getting created by StampAndSort containing partial files. | Occurs when files are written to the mail folder by an application other than the IBM Transformation Extender. See the section titled " Interfacing with non-IBM Transformation Extender systems " for additional information. |
| System runs slow, low CPU utilization; most maps are in Resource pending state. | This is a configuration error. Avoid configuring maps that can cause them to become Resource Pending. See the section titled " Avoiding Resource Pending Configurations " for additional information. |
| File errors when writing to other servers. | If the paths are correct and/or the drive mappings have not been reset then it is most likely a permission problem. The Launcher user id must have write permissions on all the folders that are configured as PUT Post Offices. |
| Valid files fail X12 Validation. | Each new transaction/version combination is added, the EDI Wizard has to be executed to generate a new x12mail.mtt type tree that has to go into the folder where all the Message Manager sources are located. Next, the customer has to deploy the RunMaps system for the new transactions to go into effect on the server. Deploying MessageManager is not needed. |
| Tradelink errors even though the Trade Links have just been added to Partner Manager. | Message manager exports the Trade Links information at Launcher startup and each night at midnight. For new Trade Links to go in effect immediately, the export has to be triggered from the Message Manager Config. section in Partner Manager. The export has to be triggered using the Run Export Process on Demand feature from the Message Manager Config. section in Partner Manager. |
| Valid files are getting rejected by StampAndSort as invalid file names when they do conform to the Trading Manager standards | Message Manager is failing to access the database. Either RunMaps where not redeployed after correcting the update.mdq file or the database is not available. |

XML Autoload Utility

The XML Autoload Utility contains the following key components:

- Utilities
- Run Options
- Remember Database User and Password
- Data Exceeding Maximum Length
- Report Options
- [Overview](#)
- [Using the XML Autoload Utility](#)
- [Defining tables and fields](#)

Overview

The XML Autoload Utility enables you to add, change, or delete selected data objects in the IBM WebSphere Transformation Extender Partner Manager database without the need for manual data entry in the Partner Manager GUI. A custom program may be used to create the XML input file, or it can be created using maps that use a Transformation Extender Type Tree based on the XML DTD file included with the product. The XML AutoLoad Utility can be run interactively, or it can be run silently through command line parameters. An embedded security function is also included to prevent inadvertent additions, changes, or deletions of data.

This utility is provided with XML input and open database connectivity (ODBC) system data source name (DSN) pointing to the destination database. This ensures that duplicate data are rejected, foreign keys exist, range values are achieved, and surrogate keys are generated.

Note: If you have used the XML AutoLoad Utility previously, be aware that in this version of the product, a test post office has been introduced to X12 outbound links. The existing post office field, PutMailPO, is now the production post office. If you make no changes to your XML script generation, the post office that you have used previously for X12 Outbound Links will be this production post office. See the fields TestIndicator, ProdIndicator and TestPutPO fields in the tag X12OBAppLink for more information. If you do not want to take advantage of these new test vs. production post offices, set the two post offices to the same value.

- [Add, change, or delete](#)
 - [Embedded security](#)
-

Add, change, or delete

The XML Autoload Utility enables you to add (from any source), change, or delete the following data objects in the Partner Manager database:

- Post Offices (both Get and Put)
 - Trading Partners/Application Partners (X12, EDIFACT, and TRADACOMS, Internal and External).
 - Contact information associated with Trading/Application Partners.
 - Trade Links(X12, EDIFACT, and TRADACOMS, Internal and External).
-

Embedded security

The embedded security feature, originally a function of Partner Manager, prevents the potential security risk of users changing or deleting data in your database.

If you have Partner Manager Users defined, the XML Autoload will open a window that requests a User Name and Password. The user must have the XML Autoload check box enabled in their Partner Manager's Security Group Maintenance window to proceed.

The **XML Autoload** check box is disabled as a result of a conversion from a prior revision.

To prevent inadvertent additions, changes, or deletions, the check boxes **Allow Add**, **Allow Change**, and **Allow Delete** in the Partner Manager XML Data Load Utility dialog box must be enabled to allow the operations to proceed.

Using the XML Autoload Utility

The IBM WebSphere Transformation Extender XML Autoload Utility contains the following key components:

- Utilities
- Run Options
- Update Options
- Remember Database User and Password
- Data Exceeding Maximum Length
- Report Options

Foreign Keys Requirement - Because data is inserted by an SQL transaction, before you use the XML AutoLoad Utility, foreign keys (dependencies) must already be in the database. For example, you cannot add **Trading Partners** and then a **Trade Link** for those new **Trading Partners** within the same XML input file; separate these two into two XML files.

Another example is **Post Offices**. You cannot add a new **Post Office** and then reference that new **Post Office** in a **Trade Link** within the same XML file. You must separate these into separate utility runs.

There are two exceptions to this rule. **Application Partners** and **Contacts** do not require their parent records added in a previous run of this utility because these records are direct siblings in the XML input format (their definition exists only within their parent record).

When entering data in Partner Manager be aware that case sensitivity can vary by Database Vendor and even by database instance. For example, Microsoft Access considers "Acme" to equal "ACME"; most other databases consider these fields unequal. Consult your Database Administrator to determine if your database is case sensitive and plan your database entries accordingly, especially if using Partner Manager to copy and merge functionality between database vendors.

- [Starting the XML Autoload Utility](#)
 - [Utilities](#)
 - [Run Options](#)
 - [Update options](#)
 - [Remember database user and password](#)
 - [Data exceeding maximum length](#)
 - [Report options](#)
 - [Unattended operation](#)
-

Starting the XML Autoload Utility

About this task

The XML AutoLoad Utility may be started from the Start menu or a desktop icon.

To start the XML AutoLoad Utility from the Start menu:

Procedure

Click **Programs -> Trading Manager -> XML AutoLoad Utility**. The WebSphere DataStage TX Partner Manager XML AutoLoad Utility window opens.

Utilities

The **Utilities** command box in the **XML Autoload** window contains the **Edit XML File** and **Generate Config** buttons. The Edit XML File function is explained below. The Generate Config button is described in the section titled "[Generating a Configuration File](#)".

- [Edit XML file](#)
-

Edit XML file

You click the **Edit XML File** command button to activate the **Autoload XML Editor** window. The Autoload XML Editor is a word-processor designed for XML syntax. It can validate the XML against a document type definition (DTD) and automatically indent your XML for easy debugging.

DTD for XML Data Validation - XML data is validated through the **tmnn.dtd** that is installed in the *install_dir\tmgr_vn.n\pmgr* directory (where *nn* and *n.n* represent the current Trading Manager version number).

Run Options

The **Run Options** box on the **XML Autoload** window contains two buttons: **Run Load** and **Validate Only; Do not change Database**.

- [Run Load](#)
 - [Validate only](#)
-

Run Load

When you enable the **Run Load** button the utility will change your database.

Validate only

When you enable the **Validation Only; Do not change Database** button, the database is not updated. Data is checked and reports all errors/warnings. Errors that could only be found if the data was written to the database are not reported. For example, if you add a **Trading Partner** and then add the same **Trading Partner** later in the validation run, that error is not caught in a **Validation Only** run, as the database was not updated.

Note: It is recommended to run your XML using the **Validation Only** option first to find syntax and other foreign key validation errors.

Update options

The **Update Options** group box on the **XML Autoload** window contains three check boxes. To prevent inadvertent additions, changes, or deletions, these check boxes, **Allow Add**, **Allow Change**, and **Allow Delete** must be enabled to allow the operations to proceed. The check boxes permit the control of the processing and prevent accidental changes and options to control the processing of errors during additions, changes, and deletions.

- [Allow add](#)
 - [Allow change](#)
 - [Allow delete](#)
 - [Change or delete error processing options](#)
-

Allow add

Enable this check box to allow the XML Autoload Utility to add data, the default processing option.

Allow change

Enable this check box to allow the XML Autoload Utility to change existing data.

Allow delete

Enable this check box to allow the XML Autoload Utility to delete existing data.

Change or delete error processing options

The following options are available during a change or delete operation. This setting is useful, for example, if you want to ignore deletions or changes that can not be processed because the record does not exist, and you want to continue processing the other records in the XML input file. Most errors will be reported as "warnings" when the **Continue Processing** option is enabled. Due to the complexity of **Trade Link** records, these settings only apply to **Post Offices** and **Trading Partners**. An error reported during a change to a **Trade Link** will always abort the entire run. These settings only apply to change or delete processing. Errors reported while adding new data will always abort. These settings do not apply when running a **Validate Only** run, or if the **Allow Update/Delete** options are not enabled.

- **Abort Run** - Choose this option to stop processing at the first error detected
- **Continue Processing** - Choose this option to skip the record in error and continue processing. Certain fatal errors (for example, database errors and XML structure errors) will abort regardless of this setting.

It is recommended that you choose the **Abort Run** as a default option.

Remember database user and password

As an additional security function, the **WebSphere DataSage TX Partner Manager XML Autoload** XML Data Load Utility window includes a **Remember Database User and Password** check box. If this check box is disabled, the **User** and **Password** fields must be populated manually each time the utility is run.

This is the user and password assigned by your DBA for access to your database. It is *not* the user and password created in Partner Manager.

For more information on the security features see the **Security** section of the Partner Manager documentation.

Data exceeding maximum length

The **Data Exceeding Maximum Length** option box on the **XML Autoload XML Data Load Utility** window contains two buttons: **Reject** and **Truncate**:

- The **Reject** - choose this option to truncate data that is too long for the database definitions.
- The **Truncate** - choose this option to truncate data that is too long for the database definitions. Truncated data is reported as a warning.

Report options

The **Report Options** group box in the **XML Autoload** window contains two buttons: **Detailed** and **Summary**.

- [Detailed](#)
- [Summary](#)
- [Print report](#)

Detailed

When you enable the **Detailed** option button it displays every field and record as it is processed.

Summary

When you enable the **Summary** option button it displays only the errors, warnings, and final results of the processing. A report listing warnings/errors is displayed.

Warnings are preceded by and followed by three dashes (---). Errors are preceded by and followed by three asterisks (***) . The contents of this report can also be copied to your clipboard.

To copy the report to your clipboard:

1. Click on the report area.
2. Press **Ctrl+A** to select all.
3. Press **Ctrl+C** to copy it to your clipboard.

Print report

Once an XML file has been processed, the **Print Report** button on the **XML Autoload** window is enabled. Pressing this button prints the contents of the report listing warnings and errors.

Unattended operation

The XML Autoload Utility can run unattended by setting options that are available from a configuration file and using command line flags. The configuration file can be built manually, or one can be generated from the parameters on the GUI. Instead of the report generated on the screen of the XML Autoload, all output is written to a file that you specify.

The default configuration file, **PMAutoloadConfig.xml**, is found in the same location as the XML Autoload Utility executable. It is supplied with all options empty. Parameters are specified in XML format. For example, if the XML input file is **c:\xml\input.xml**, the configuration file would contain:

```
<xmlsource>c:\xml\input.xml</xmlsource>
```

You can have as many configuration files as needed. A command line flag (**-cf**) specifies the configuration file to use. For example, you could have one configuration file that is "Validate Only", and another that actually updates the database.

All configuration files are validated by the DTD: **XMLConfignn.dtd**. This DTD is supplied in the same directory as the programs; typically this is:

IBM\WebSphere Transformation Extender\tnmgr_vn.n\pmgr

where *n.n* is the product version number.

You cannot create a configuration file without referencing this DTD.

When the program completes, the **End Of Report** message will contain:

Process Return Code: X

where x is 0 = OK, -1 = Error; no database changes made, 2 = Warnings issued.

This allows you to parse the output file to determine success or failure.

Warning: The XML Autoload may need a password to connect to your database, and also may need a Partner Manager user ID and password if those are in use. These passwords will be protected in the configuration file if you build the configuration file from the XML Autoload Utility. If you build the configuration file yourself, you will have to enter passwords in "clear text", which will be readable from anyone with access to your system. It is recommended you always start with a generated configuration file and do not attempt to modify the passwords manually.

- [Generating a configuration file](#)
 - [Command line flags](#)
 - [Running unattended](#)
 - [Configuration and command line parameters](#)
 - [Examples](#)
-

Generating a configuration file

About this task

To automatically generate a configuration file, refer to the following procedures:

Procedure

1. Run the XML Autoload as you normally would, for example, from the **Start** > Programs menu item.
2. Set all options to be saved to the configuration file.
3. In the **Utilities** group box, click the **Generate Config...** button. The system will check to make certain that the database connection information is correct.
4. The system will determine if a Partner Manager user id and password is required in that database. If they are required, you will be asked to provide them.
5. The system will prompt you for the name of the configuration file to save, defaulting to **PMAutoloadConfig.xml**. If the file exists, a message will be displayed that asks if it is OK to over-write the file. The saved configuration file is then ready for use, or for manual modification. You will be given the option of viewing and editing the generated file when the process completes.

Results

Note: Only one configuration can be placed in a file. For example, you cannot duplicate the input file with the expectation that the XML Autoload will run twice. A XML DTD is also supplied to enforce this rule.

Command line flags

Command line flags are used to run the XML Autoload, and will override the values in a configuration file. All command line flags must begin with "-" (a dash), and are two characters in length, with the parameter value immediately following (no space separating flag and value). For example, to specify the XML source (input) file **c:\xml\my.xml**, the command line flag is **-xs**, and the entire command line would be:

```
PMAutoload.exe -xsc:c:\xml\my.xml
```

If a command line parameter value contains a space or a "-", it must be enclosed in double quotes (""). Use of a single quote in parameter values is not supported. Also, do not use a forward slash ("\\") in file names. You must always use a back-slash ("\\").

Command line parameters that are not specified will be looked up in the file: **PMAutoLoadConfig.xml**, or the file specified in "-cf".

Running unattended

To run unattended, run the XML Autoload and supply the command line flag "-cf". For example, the following DOS Command, if issued from the directory the XML Autoload program is located in, will run the XML Autoload using the default configuration file, **PMAutoLoadConfig.xml**:

PMAutoLoad.exe -cf

By default, the XML Autoload Utility is installed in the following directory:*install_dir \WebSphere Transformation Extender\tnmgr_vn.n\pmgr* directory. If the XML Autoload Utility is not located in this directory, you must fully specify the location of **PMAutoLoad.exe** in the command line.

Unattended operation is a background process that is not readily apparent to you. Assuming no error messages appear on the screen, you can check the status of the program's execution by looking at the **Output File**. See the <outputfile> tag, below, for more information

Debugging a Background Task

- [**Debugging a background task**](#)

A special command-line flag, **-dm** (debug mode), exists to help you with the potentially difficult task of debugging a background task.

Debugging a background task

A special command-line flag, **-dm** (debug mode), exists to help you with the potentially difficult task of debugging a background task.

If you are having trouble with your command line flags or configuration file, add **-dm** and the following will occur:

- The screen will be visible, and you can see your settings on the GUI.
- In debug mode, the program will not exit by itself. The following message will be displayed to inform you that the program is terminating: **Program ending in debug mode; Press OK**
- All messages normally written to the output file will also appear as popup messages. Because many messages may appear, two options will be presented when displaying these popup messages: **End Program** or **Stop Displaying Messages**.

Debug mode can only be activated if the **Run** option is **Validate Only**. This is to insure that you cannot modify the parameters through the GUI, or cancel the program when it is updating the database.

Configuration and command line parameters

All parameters not specified "optional" in the following table must be specified, either in the configuration file or in concert with the command line flags.

Even if a parameter is optional, it may be required by your database configuration.

If the command line is malformed to the extent that further processing cannot proceed, the output file cannot be determined, or a fatal file operation occurs, a message will display on the screen that describes the error. In this case, nothing will be written to an output file. A record of the fatal error will display in your Windows **Event Viewer**, under the **Application Log** group, with a source name of **XML Autoload**. Examples include failing to close a quote in a parameter, or being unable to create an output file due to permission errors. All other errors will be reported in the output file.

Warning: Use care when specifying output file options. If you specify "overwrite" the contents of the last output file will be lost; you will not be prompted of this condition. If a file is overwritten a warning will appear in the output file.

In addition, remember the following as you work with these command line options:

- All parameter values are case sensitive except file names.
- All XML tags and command line flags must be lowercase.
- XML tags must be entered in the order displayed (the DTD validation will fail otherwise)
- The command line flags can be listed in any order, but cannot be repeated.

| Configuration Parameter | Command Line | Notes |
|-------------------------|--------------|--|
| Not applicable | -cf | Configuration file name: specify the configuration file to run unattended.
Required to run in unattended mode. Parameter value, if not specified, will default to PMAutoLoadConfig.xml. |
| <xmlsource> | -xs | XML Input file. When used in the command line enclose in quotes if contains a space or a dash. |
| <pwprotection> | | Password protection: 0=clear text (no protection) 1=protected. Tells system format of passwords listed below. |
| <dsn> | | ODBC Data Source Name. |
| <dbuser> | | Database user name. Optional; may be required for your database. |
| <dbpass> | | Database password. Optional; may be required for your database. |
| <database> | | Database name. Optional; may be required for your db. |
| <dbvendor> | | Database vendor: Access, DB2, SQL Server, Oracle, Sybase |
| <pmluser> | | Partner Manager User ID. Optional; required if users are defined in Partner manager |
| <pmpass> | | Password for Partner Manager User ID. Optional; required if <pmluser> above has a password. |
| <rptformat> | | Report Format Option: d=Detailed, s=Summary. |
| <exceedmax> | | Data exceeding Max Len: r=Reject, t=Truncate. |

| Configuration Parameter | Command Line | Notes |
|-------------------------|--------------|---|
| <runoption> | | Run Option: r=Run, v=Validation Only. |
| <updateoption> | | Update Options: a=Allow Add, c=Allow Change, d=Allow Delete. Can combine; for example, to allow all: <update option>acd</update option> |
| <errors> | | Change/Delete Error Processing Control: a=Abort, c=Continue |
| <outputfile> | -of | File to write of all messages/errors. Optional; default: PMAutoloadOut.txt, in same directory as exe. |
| <outputcontrol> | -oc | If output file exists, a=Append, o=Overwrite. No warning issued when overwriting, but a message will appear in output file. Default is "a" (Append) |

Examples

The following examples show how to use the XML Autoload Utility.

Example 1

The XML Autoload is to run unattended, using all parameters set in the default configuration file: **PMAutoload.exe -cf**

Example 2

The XML Autoload is run in its normal mode, with the GUI displayed: **PMAutoload.exe**

Example 3

The XML Autoload is to run unattended, using the configuration file **c:\My Files\MyXML.xml**, and with the output control to "append". The file name contains spaces, so it is enclosed in double quotes:

PMAutoload.exe -cf"c:\My Files\MyXML.xml" -oca

Defining tables and fields

The XML Autoload Utility field and table definitions, provided here, are used for adding new objects or changing or deleting objects in the Partner Manager database.

The following are field descriptions for the XML tables

- **Field Label** is the label on the screen as it appears in Partner Manager.
- **FieldName/Tag** is the database field name (if one exists), and is used as the XML tag. There are two distinct sets of XML tags included in the XML Autoload Utility: one set is used to add data and another set is used to change existing data. These separated tags ensure backward compatibility for users of prior versions of this utility. (Your script generation routines for adding data remain unchanged.) The separation of tags also prevents you from accidentally changing data when you intended to add data. The new change and delete tags always end with **Change** or **Delete**, for example, **AddChangeDelete**. The tags related to Documents on a Trade Link always end with **AddChangeDelete**, for example, **X12InboundGroupsAddChangeDelete**.
- **Req'd** means this field must be in the input to AutoLoad, it is not the database definition of required. The abbreviation **fk** = Foreign Key (that value must exist in the specified table).
- **Max. Len.** is the maximum length of the field (minimums, if any, are in the notes).
- **Validation** defines the valid values for the fields.
- **Notes** contain detailed information about the tag. System generated fields such as Surrogate keys, Optimistic Locking Fields, and Audit Information (for example, **CreatedBy/DateCreated**) and are not listed here. These fields are populated automatically by the Autoload Utility. Leading and trailing spaces, if any, are trimmed before saving to the database. **CreatedBy** is set to **AutoLoad** so that any problems associated with this utility can be tracked.
- [Table and Field Types](#)
- [Post offices](#)
- [Contacts](#)
- [X12 Trading Partners](#)
- [Automatic creation of X12 acknowledgement trade links](#)
The documentation provided here describes the tags needed to create trade links in the XML AutoLoad Utility. An additional feature for X12 trade links is the ability to automatically create the acknowledgement trade link at the same time.
- [X12 inbound trade links](#)
- [X12 outbound trade links](#)
- [EDIFACT partners](#)
- [EDIFACT inbound trade links](#)
- [EDIFACT outbound trade links](#)
- [TRADACOMS Trading/Application Partners](#)
- [TRADACOMS inbound trade links](#)
- [TRADACOMS Outbound Trade Links](#)

Table and Field Types

The following sections describe the specific requirements for defining the XML input files' table and field requirements for the XML AutoLoad Utility. The table and field types described in these sections are:

- "[Post Offices](#)"
- "[Contacts](#)"
- "[X12 Trading Partners](#)"
- "[X12 Inbound Trade Links](#)"
- "[X12 Outbound Trade Links](#)"
- "[EDIFACT Partners](#)"
- "[EDIFACT Inbound Trade Links](#)"
- "[EDIFACT Outbound Trade Links](#)"
- "[TRADACOMS Trading/Application Partners](#)"
- "[TRADACOMS Inbound Trade Links](#)"
- "[TRADACOMS Outbound Trade Links](#)"

Post offices

The following table, field, and tag descriptions are for Post Offices fields associated with the XML Autoload Utility:

- "[Post Offices \(Tag: PostOffices\)](#)"
- "[Post Offices Delete \(Tag: PostOfficesDelete\)](#)"
- "[Post Offices Change \(Tag: PostOfficesChange\)](#)"

WARNING: If the "Use One GET Post Office per Standard" is set, you will be prevented from adding a Get Post Office, changing a Put Post Office, or deleting a Get Post Office. Any Get Post Office specified in an XML input file will be ignored.

- [Post Offices \(Tag: PostOffices\)](#)
- [Post Offices Delete \(Tag: PostOfficesDelete\)](#)
- [Post Offices Change \(Tag: PostOfficesChange\)](#)

Post Offices (Tag: PostOffices)

Use this tag to add new Post Office records. A warning is reported (or an error if **Abort Run** is selected) if the Post Office already exists in the Partner Manager database.

The tag and field descriptions are:

| Field Label | FieldName/Tag | Req'd | Max. Len. | Validation | Notes |
|-----------------------------|-------------------|-------|-----------|-------------------|-------|
| Post Office Name | POName | Yes | 50 | Unique | |
| Get/Put | Get_Put_Indicator | Yes | 1 | G or P | |
| PO Type | POType | Yes | 50 | See note 1 | 1 |
| Nickname | PONickname | No | 3 | Unique if present | 2 |
| Directory or Adapter String | AdapterString1 | No | 255 | | 3 |

Notes:

1. The PO Type must be an active Post Office Type, as maintained in Partner Manager. Examples include: VAN, FILE, and EMAIL.
2. If the PO Nickname is not supplied, it is generated by the system when saved. Must be 3 characters long if supplied, and not already in use.
3. If the type is File then a trailing right or left slash (\, /) is appended if not present and this field was supplied (UNIX or DOS directory separator).

Post Offices Delete (Tag: PostOfficesDelete)

Use this tag to delete existing Post Office records. A warning is reported (or error if **Abort Run** is selected) and will not delete if a Post Office is assigned to a trade link. A warning is reported (or an error if **Abort Run** is selected) if a Post Office does not exist.

The tag and field descriptions are:

| Field Label | FieldName/Tag | Req'd | Max. Len. | Validation | Notes |
|------------------|---------------|-------------|-----------|------------|-------|
| Post Office Name | POName | See Note 1. | 50 | | 1 |
| Nickname | PONickname | See Note 1. | 3 | | 1 |

Notes:

1. Supply **POName**, **PONickname**, or both to delete. At least one of these fields must be specified.

Post Offices Change (Tag: PostOfficesChange)

Use this tag to change existing Post Office records. A warning is reported (or an error if **Abort Run** is selected) if the Post Office does not exist.

The tag and field descriptions are:

| Field Label | FieldName/Tag | Req'd | Max. Len. | Validation | Note |
|-------------|---------------|-------|-----------|------------|------|
| | | | | | |

| Field Label | FieldName/Tag | Req'd | Max. Len. | Validation | Note |
|-----------------------------|-------------------|-------------|-----------|-------------|------|
| Post Office Name | POName | See Note 1. | 50 | | 1 |
| Get/Put | Get_Put_Indicator | Yes | 1 | G or P | |
| PO Type | POType | No | 50 | See Note 2. | 2 |
| Nickname | PONickname | See Note 1. | 3 | | 1 |
| Directory or Adapter String | AdapterString1 | No | 255 | | 3 |
| N/A | PONameChange | N | 50 | | 4 |
| N/A | PONicknameChange | N | 3 | | 4 |

Notes:

- Specify the **POName**, **PONickname**, or both, of the existing Post Office you wish to change. Do not use these fields to change a Post Office name or nickname; see the fields **PONameChange** and **PONicknameChange**.
- The PO Type, if specified, must be an active Post Office Type, as maintained in Partner Manager. Examples include: VAN, FILE, and EMAIL.
- If the type is File then a trailing right or left slash (\, /) is appended if not present and this field was supplied. (UNIX or DOS directory separator).
- If you want to change the name, or nickname, of an existing PostOffice, specify the change in these two fields. These fields, if specified, must be unique: they cannot already be in use on another Post Office. Changing the name does not automatically change the nickname.

Contacts

The following table, field, and tag descriptions are for **Contacts** fields associated with the XML Autoload Utility.

- [Contacts \(Tag: Contacts\)](#)
-

Contacts (Tag: Contacts)

Contacts may be associated with the Trading or Application Partner being added. For example, the **<Contacts>...</Contacts>** group(s) may be between the **<X12TradingPartners>...</X12TradingPartners** and **<X12AppPartners>...</X12AppPartners>** (or similar EDIFACT and TRADACOMS tag, and/or associated Change) tags. Adding contacts to an existing Trading or Application Partner in the Partner Manager database is supported if nested within a Change tag. When adding contacts to an existing object you cannot add a contact if one previously exists with the same last and first names through this utility. There can be zero or many contacts associated with each Trading or Application Partner. If the associated Trading or Application Partner is rejected, all contacts for the rejected Partner are rejected. The tag and field descriptions are described in the following table:

| Field Label | FieldName/Tag | Req'd | Max. Len. | Validation | Notes |
|-----------------------------------|------------------|-------|-----------|---------------|-------|
| Last Name | LastName | Yes | 30 | | |
| First Name | FirstName | Yes | 30 | | |
| Title | Title | No | 25 | | |
| Primary Address | Address_1 | No | 25 | | |
| Secondary Address | Address_2 | No | 25 | | |
| City | City | No | 30 | | |
| State | State | No | 30 | | |
| Zip Code | ZipCode | No | 30 | | |
| Country | Country | No | 30 | | |
| Phone | Phone | No | 20 | | |
| E-Mail | Email | No | 255 | | |
| Beeper | Beeper | No | 20 | | |
| Notes | Notes | No | 255 | | |
| Alert Type | AlertType | No | 50 | Internal Only | 1, 2 |
| (Alert Options)
E-Mail | AlertEmail | No | 255 | Internal Only | 3 |
| (Alert Options) Monitor Directory | MonitorDirectory | No | 255 | Internal Only | 3 |
| Attach Data | AlertEmailAttach | No | 1 | T/F | 4 |

Notes:

- All Alert fields apply to Internal Trading/Application Partners only. The valid values for AlertType are: UFO, Syntax, and ANY. The values are case sensitive. This is for future use.
- Default: UFO
- If any of these exist, the corresponding Ind (check box on the form) field is set to True (-1) in the Contacts table, otherwise set the Ind fields to False (0). Unlike the data entry form, the AlertEmail field does not default to the Email field; you must specify both if requiring both.
- Set to T (true) to attach the data to the email alert. Requires an email address in the field AlertMail.

X12 Trading Partners

The following table, field, and tag descriptions are for X12 Trading Partners fields associated with the XML AutoLoad Utility:

- "[X12 Trading Partners \(Tag: X12TradingPartners\)](#)"
- "[X12 Application Partners \(Tag: X12AppPartners\)](#)"
- "[X12 Application Partners Delete \(Tag: X12AppPartnersDelete\)](#)"
- "[X12 Application Partners Change: \(Tag X12AppPartnersAddChange\)](#)"
- "[X12 Trading Partners Change \(Tag: X12TradingPartnersChange\)"](#)
- "[X12 Trading Partners Delete \(Tag: X12TradingPartnersDelete\)"](#)

- [**X12 Trading Partners \(Tag: X12TradingPartners\)**](#)
- [**X12 Application Partners \(Tag: X12AppPartners\)**](#)
- [**X12 Application Partners Delete \(Tag: X12AppPartnersDelete\)**](#)
- [**X12 Application Partners Change: \(Tag X12AppPartnersAddChange\)**](#)
- [**X12 Trading Partners Change \(Tag: X12TradingPartnersChange\)**](#)
- [**X12 Trading Partners Delete \(Tag: X12TradingPartnersDelete\)**](#)

X12 Trading Partners (Tag: X12TradingPartners)

Use this tag to add new X12 Trading Partners.

Trading Partners can have **Contacts** added at the same time; nest the Contact record within the Trading Partner tags. See [Contacts](#) for more information.

The tag and field descriptions are described in the following table:

| Field Label | FieldName/Tag | Req'd | Max. Len. | Validation | Notes |
|--|---------------------|-------|-----------|------------------------------------|-------|
| Internal/
External | IntExt | Yes | 1 | "I" or "E" | 1, 2 |
| Partner Name | PartnerName | Yes | 35 | Unique | 3 |
| ISA Address | IchgQual | Yes | 2 | fk to IchgIDQualifiers | 4 |
| Interchange ID | IchgID | Yes | 15 | Unique with IchgQual | |
| Authorization Qualifier | INBAAAuthQual | No | 2 | fk to AuthQualifiers | 14 |
| Authorization ID | INBAAuth | No | 10 | | 14 |
| Security Qualifier | INBSecurityQual | No | 2 | fk to SecurityQualifiers | 14 |
| Security Password | INBSecurityPWD | No | 30 | | 14 |
| Interchange Standards ID or Repetition Character | IchgCtrlStdsID | No | 8 | fk to IchgStandardsIC or HexValues | 6 |
| Interchange Ctl Version ID | IchgCtrlVerNo | No | 5 | fk to IchgStandardsVer | 7 |
| Element Separator | ElementSeparator | No | 5 | fk to HexValues | 8, 9 |
| Sub Element Separator | SubElementSeparator | No | 5 | fk to HexValues | 9, 10 |
| Segment Terminator | SegmentTerminator | No | 5 | fk to HexValues | 9, 11 |
| Duplicate Control | Duplicate_Control | No | 3 | | 13 |
| Last Interchange Control # | ISAControlNoValue | No | 9 | | 12 |
| TA1GetPO | TA1 Get Post Office | No | 3 | Nickname of a Get PO | |
| TA1PutPO | TA1 Put Post Office | No | 3 | Nickname of a Put PO | |
| Authorization Qualifier | OUTBAAuthQual | No | 2 | fk to AuthQualifiers | 14 |
| Authorization ID | OUTBAAuth | No | 10 | | 14 |
| Security Qualifier | OUTBSecurityQual | No | 2 | fk to SecurityQualifiers | 14 |
| Security Password | OUTBSecurityPWD | No | 30 | | 14 |
| Validate Authorization | ValidateAuth | No | 1 | Y/N | 15 |
| Validate Security | ValidateSec | No | 1 | Y/N | 15 |

Notes:

1. Determines folder.
2. Standard Internal and External X12 Folders are used (no custom folders). These custom folders are: -2 = Internal FolderNo; and -21 = External FolderNo.
3. Do not use the name "UNKNOWN, or "ALL", or any variation of the case of these two words. For example, do not use "Unknown", or "All". Unique across all X12 Trading Partners.
4. Cannot contain single or double quotation marks. Combination of **IchgQual + IchgID** must be unique across all X12 Trading Partners.
5. Default: 00
6. The field **IchgCtrlStdsID** takes on two meanings, depending on the value of the next field, **IchgCtrlVerNo**. If the **IchgCtrlVerNo** is greater than, or equal to, 00402, this field is the Repetition Character, otherwise it is the Interchange Standards ID. Both field types are optional: the default value is "A" (caret, Hex 5E) if the field denotes the Repetition Character; the default value is "U" for the Interchange Standards ID. When specifying the Repetition Character, use the hexadecimal values as stored in the table HexValues. When specifying an Interchange Standards ID, the value must be in the table **IchgStandardsIC**. The reason for this dual meaning is that the X12 committee stopped using the Interchange Standards ID and replaced it with the Repetition Character, starting with Interchange Control version 00402.
7. Default: 00305
8. Default: 2A
9. If more than one hex value, separate by a space (for example, 0D 0A). Defaults are: 2A (*); 3A (colon); and 0A: Linefeed.
10. Default: 3A
11. Default: 0A
12. Field can only be set for External Trading Partners that have a Duplicate Control Interchange setting of Y. If set, it must be a positive integer no longer than nine characters.
13. This field can only be specified for external partners. This is a three character string containing the letters **N** or **Y**: set each position to a **Y** if the **Duplicate Control** checkbox is checked as follows: position 1=Interchange, position 2=FG within Interchange, position 3=Transaction within FG.
14. Use for External Trading Partners only.
15. Use for External Trading Partners only. Default is **N**.

X12 Application Partners (Tag: X12AppPartners)

Use this tag to add X12 Application Partners while adding X12 Trading Partners. There can be zero or more Application Partners associated with a Trading Partner, however, an Application Partner is required when a Trade Link is created for a Trading Partner. If any Trading Partners are rejected, all associated Application Partners are rejected as well. X12 Application Partners can have contacts added. See "[Contacts](#)" for more information.

The tag and field descriptions are:

| Field Label | FieldName/Tag | Req'd | Max. Len. | Validation | Notes |
|--------------------------|------------------|-------|-----------|--------------------------------------|-------|
| N/A (ISA + IID) | ISA_IID | Yes | 17 | IchgQual + IchgID of Trading Partner | 1 |
| Application Partner Name | ApplicationName | Yes | 35 | Unique within the Trading Partner | 2 |
| GS Address | GS_ApplicationID | Yes | 15 | Unique within the Trading Partner | 3 |
| Alias | AliasName | No | 20 | Unique if present | 4 |
| User Defined 1 | UserDefined1 | No | 20 | | |
| User Defined 2 | UserDefined2 | No | 20 | | |
| Institution | Institution | No | 255 | | 5 |

Notes:

- Combination of previous Trading Partner's ISA Address and Interchange ID. Used to confirm that the XML is properly formed (Application Partners nested within the parent Trading Partner).
- Do not use the name "UNKNOWN", or "ALL", or any variation of the case of these two words. For example, do not use "Unknown", or "All".
- Cannot contain single or double quotes.
- Alias Names are unique across all X12 Application Partners, if present.
- Specify optional HIPAA Institution, used for HIPAA Type 7 validations. If specified, must exist in the table HIPAA_Institutions. Valid only for External Application Partners.

X12 Application Partners Delete (Tag: X12AppPartnersDelete)

Use this tag to delete an X12 Application Partner. It will also delete any contacts associated with the Application Partner, but will not delete if Application Partner is assigned to a trade link (see [X12TradingPartnersDelete](#) if this functionality is needed). It reports a warning (or error if **Abort Run** is selected) if Application Partner does not exist.

To identify the Application Partner to delete, you must specify the **ISA_IID**, ad then identify the Application Partner via **ApplicationName** and/or **GS_ApplicationID**. You can specify just **ApplicationName**, or **GS_ApplicationID**, or both. If specifying both, the system will ensure that name belongs to the ID. The tag and field descriptions are:

| Field Label | FieldName/Tag | Req'd | Max. Len. | Validation | Notes |
|--------------------------|------------------|-------|-----------|--------------------------------------|-------|
| Application Partner Name | ApplicationName | No | 35 | | |
| GS Address | GS_ApplicationID | No | 15 | | |
| N/A (ISA + IID) | ISA_IID | Yes | 17 | IchgQual + IchgID of Trading Partner | |

X12 Application Partners Change: (Tag X12AppPartnersAddChange)

Use this tag to add an X12 Application Partner to an existing Trading Partner, or change an existing Application Partner. To add, set the tag "AddFlag" to **Y**; leaving "AddFlag" blank, not specified, or set to **N** will change the existing X12 Application Partner if it exists.

Contacts can be added via this tag as well; include a [Contacts](#) tag set within the **X12AppPartnersAddChange** tags to add new Contacts to an existing Application Partner. You cannot add a Contact if one exists with that First and Last Name.

Note: Deleting or changing existing Contacts is not supported by the XML Autoload Utility, and must be done using Partner Manager.

ApplicationName and **GS_ApplicationID** together are not required when changing, but at least 1 must be specified to identify the Application Partner. If AddFlag is **Y**, then these two fields are required.

If changing the Application Name or GS Address, use **ApplicationNameChange** and **GS_ApplicationIDChange**. These "Change" fields cannot be specified when the **AddFlag** is **Y**. The tag and field descriptions are:

| Field Label | FieldName/Tag | Req'd | Max. Len. | Validation | Notes |
|--------------------------|------------------------|-------|-----------|--------------------------------------|-------|
| N/A (ISA + IID) | ISA_IID | Yes | 17 | IchgQual + IchgID of Trading Partner | 1 |
| Application Partner Name | ApplicationName | No | 35 | Unique within the Trading Partner | 2 |
| GS Address | GS_ApplicationID | No | 15 | Unique within the Trading Partner | 3 |
| Alias | AliasName | No | 20 | Unique if present | 4 |
| User Defined 1 | UserDefined1 | No | 20 | | |
| User Defined 2 | UserDefined2 | No | 20 | | |
| Institution | Institution | No | 255 | | 5 |
| N/A | AddFlag | No | Y/N | | |
| N/A | ApplicationNameChange | No | 35 | | |
| N/A | GS_ApplicationIDChange | No | 15 | | |

Notes:

1. Combination Trading Partner's ISA Address and Interchange ID associated with this Application Partner.
2. Do not use the name "UNKNOWN", or "ALL", or any variation of the case of these two words. For example, do not use "Unknown", or "All".
3. Cannot contain single/double quotes.
4. Alias Names are unique across all X12 Application Partners, if present.
5. Specify optional HIPAA Institution, used for HIPAA Type 7 validations. If specified, must exist in the table HIPAA_Institutions. Valid only for External Application Partners.

X12 Trading Partners Change(Tag: X12TradingPartnersChange)

Use this tag to change existing X12 Trading Partners. The fields in this tag are similar to X12 Trading Partners, but the **IntExt** (Internal/External) field is not specified because you cannot change an Internal Trading Partner to an External using this tag. If this is required, delete and re-add the partner. Also, unlike the tag **X12TradingPartners**, **X12AppPartners** are not nested within this tag. (Use the [X12AppPartnersAddChange](#) tag to add Application Partners to an existing Trading Partner, or to change existing Application Partners).

To specify Trading Partner to change, specify **PartnerName**, **IchgQual** and **IchgID**, or all 3 (you must specify both the Qualifier and ID if specifying either). To change the fields that identify a Partner (the Partner Name, Interchange Qualifier, or Interchange ID), use the fields **PartnerNameChange**, **IchgQualChange**, or **IchgIDChange** tags.

Trading Partners can have [Contacts](#) added at the same time; nest the Contact record within the Trading Partner tags. Use this tag, along with the Contacts tag, to add Contacts to an existing Trading Partner.

The tag and field descriptions are described in the following table:

| Field Label | FieldName/Tag | Req'd | Max. Len. | Validation | Notes |
|----------------------------|----------------------|------------|-----------|--------------------------|-------|
| Partner Name | PartnerName | See Note 1 | 35 | Unique | 1 |
| ISA Address | IchgQual | See Note 1 | 2 | fk to IchgIDQualifiers | 1 |
| Interchange ID | IchgID | See Note 1 | 15 | Unique with IchgQual | 1 |
| Authorization Qualifier | INBAAuthQual | No | 2 | fk to AuthQualifiers | |
| Authorization ID | INBAAuth | No | 10 | | |
| Security Qualifier | INBSecurityQual | No | 2 | fk to SecurityQualifiers | |
| Security Password | INBSecurityPassword | No | 30 | | |
| Interchange Standards ID | IchgCtrlStdsID | No | 1 | fk to IchgStandardsIC | |
| Interchange Ctl Version ID | IchgCtrlVerNo | No | 5 | fk to IchgStandardsVer | |
| Element Separator | ElementSeparator | No | 5 | fk to HexValues | 2 |
| Sub Element Separator | SubElementSeparator | No | 5 | fk to HexValues | 2 |
| Segment Terminator | SegmentTerminator | No | 5 | fk to HexValues | 2 |
| Duplicate Control | Duplicate_Control | No | 3 | | 3 |
| Last Interchange Control # | ISAControlNoValue | No | 9 | | 6 |
| N/A | PartnerNameChange | No | 35 | | 4 |
| N/A | IchgQualChange | No | 2 | | 4 |
| N/A | IchgIDChange | No | 15 | | 4 |
| Reset ISA Control | ResetISA_Ind | No | 1 | T/F | 5 |
| TA1GetPO | TA1 Get Post Office | No | 3 | Nickname of a Get PO | |
| TA1PutPO | TA1 Put Post Office | No | 3 | Nickname of a Put PO | |
| Authorization Qualifier | OUTBAAuthQual | No | 2 | fk to AuthQualifiers | |
| Authorization ID | OUTBAAuth | No | 10 | | |
| Security Qualifier | OUTBSecurityQual | No | 2 | fk to SecurityQualifiers | |
| Security Password | OUTBSecurityPassword | No | 30 | | |

Notes:

1. Identify the Partner using these three fields. You can specify just the name, or both ID and Qualifier, or all three.
2. If more than one Hex Value, separate by a space (for example, 0D 0A).
3. This field can only be specified for external partners. This is a three character string containing the letters N or Y: set each position to a Y if the **Duplicate Control** checkbox is checked as follows: position 1=Interchange, position 2=FG within Interchange, position 3=Transaction within FG.
4. Use these fields to change a Partner Name, or the unique identifier for a Trading Partner (Interchange ID and ISA Address/Qualifier).
5. Valid value is **T** or **F** (**T**=True/enabled; **F**=False/disabled). If setting to **T**, you must specify **ISAControlNoValue** (on screen as "Last Interchange Control Number"); if you specified an **ISAControlNoValue** this field will be set to enabled for you. If setting to **F**, this will clear the existing **ISAControlNoValue**. You can only specify **ResetISA_Ind** on External Trading Partners.
6. Field can only be set for External Trading Partners that have a duplicate Control Interchange setting of Y. If set, it must be a positive integer no longer than nine characters.

X12 Trading Partners Delete (Tag: X12TradingPartnersDelete)

Use this tag to delete existing X12 Trading Partners. To delete, specify either the Partner Name, or the combination of ISA Address (Qualifier) and Interchange ID. You can also specify all three values; the system will insure that name exists with that Qualifier/Interchange ID. A warning is reported (or error if **Abort Run** is selected), if a Trading Partner does not exist.

Use with caution, as this tag deletes all records associated with the Trading Partner - all Application Partners, Contacts, and Inbound/Outbound Trade Links.

The tag and field descriptions are described in the following table:

| Field Label | FieldName/Tag | Req'd | Max Len. | Validation | Notes |
|----------------|---------------|----------|----------|------------------------|-------|
| Partner Name | PartnerName | See Note | 35 | Unique | |
| ISA Address | IchgQual | See Note | 2 | fk to IchgIDQualifiers | |
| Interchange ID | IchgID | See Note | 15 | Unique with IchgQual | |

Automatic creation of X12 acknowledgement trade links

The documentation provided here describes the tags needed to create trade links in the XML AutoLoad Utility. An additional feature for X12 trade links is the ability to automatically create the acknowledgement trade link at the same time.

Here you will find documentation that explains how to use this automatic acknowledgement feature.

Creating or modifying a trade link so that an acknowledgement is required, or expected, will check to see if the corresponding trade link needs to be created. If that link does not exist, it will be created for you if you specify the **Acknowledgement Post Offices** option. The corresponding trade link has the same internal and external partners, but in the opposite (inbound and outbound) direction. An acknowledgement is *required* on an Inbound trade link when the XML tag Ack Level is set to a non-zero value in the XML file. An acknowledgement is *expected* on an Outbound trade link when the XML tag **ACK_Within** (number of hours an acknowledgement is expected) is set to a non-zero value. This feature duplicates the automatic acknowledgement creation feature found in Partner Manager.

Note: If you would prefer to create the acknowledgement trade link yourself, do not specify the **Acknowledgement Post Offices** and skip the remainder of this procedure. The automatic creation feature supports three different acknowledgement functional groups:

- 824
- 997
- 999

How the system determines which functional group to use is described in the following procedures.

Note: The tags mentioned are explained in detail in the trade link tables found in this documentation.

- **Inbound/outbound trade link acknowledgments**

For an inbound trade link, the acknowledgement can be a 824, 997, and/or a 999. For outbound trade links, the acknowledgement will only be 997, set to the version specified on the outbound trade link

- **Activating the feature**

In order to activate this feature, you must specify the **Acknowledgement Post Offices** option. These are the post offices that are used on the corresponding trade link. In all four X12 trade link application tags, three optional tags are used to specify these post offices:

Inbound/outbound trade link acknowledgments

For an inbound trade link, the acknowledgement can be a 824, 997, and/or a 999. For outbound trade links, the acknowledgement will only be 997, set to the version specified on the outbound trade link

For inbound trade links, it should be noted that the transaction(s) created will be based on the **HIPAA Validation Type** XML tag **ValType**, or will just be a 997 if it is a non-HIPAA transaction. The version of the 824 created depends on the setting of the 824 **Ack Version** tag specified (and will default to 4050 if not specified). The version of the 997 created depends on the setting of the **HIPAAAckFlag** tag specified (and will default to the version of the Inbound application if not specified). The version of the 999 is always version 5010.

Activating the feature

In order to activate this feature, you must specify the **Acknowledgement Post Offices** option. These are the post offices that are used on the corresponding trade link. In all four X12 trade link application tags, three optional tags are used to specify these post offices:

- **AckGetPO**
- **AckProdPutPO**
- **AckTestPutPO**

All three tags are optional, but if any are specified, at least one Put post office (test or production) must be specified.

The Get post office will become required if a Put is specified, and the corresponding trade link does not exist already. If this Get post office is specified, and the trade link exists already, the Get post office set here will be ignored (and the one used on the existing trade link remains). Failure to specify an **AckGetPO** when one is required will result in warnings in the XMLautoload in process.

These Acknowledgment Post Offices are used in the following tags and are explained in detail below:

- [X12 Outbound Applications Trade Links \(Tag: X12OBAppLink\)](#)
- [X12 Inbound Trade Links Applications Add, Change, Delete \(Tag: X12InboundGroupsAddChangeDelete\)](#)
- [X12 Inbound Trade Link Applications \(Tag: X12InboundGroups\)](#)
- [X12 Inbound Trade Links Applications Add, Change, Delete \(Tag: X12InboundGroupsAddChangeDelete\)](#)

When this feature is activated and the XML AutoLoad Utility **Reporting Options** is set to **Detailed**, information messages such as *X12 Outbound 824 Acknowledgement added to an existing trade link*, and *X12 Outbound 824 Acknowledgement added to an existing trade link*, and *X12 Inbound 997 Acknowledgement added to a newly created trade link* will be included in the report. When the XML AutoLoad Utility is in **Validation** mode, this message may repeat more than once, as the data is not actually created. In **Validation** mode, these messages will read: *would have been added* instead of *added*. No messages regarding this feature will be displayed if the reporting option is set to **Summary**.

X12 inbound trade links

Trade links can be added, changed, or deleted. In addition, existing Trade Links can have their individual Applications added, changed, or deleted ("Applications" are identified by the combination of Version and Functional ID, and are also known as "Documents").

To add a trade link, use one of the following tags and nest **X12InboundGroups** tags within the tag:

- **X12IBTradeLink**
- **X12Unknown_Link**

Note: There are two types of X12 Inbound Trade Links: Known and Unknown. When creating a Known Trade Link, start with the **X12IBTradeLink** tag. When creating an Unknown Trade Link, start with the **X12UnknownLink** tag.

To change an existing X12 Inbound Trade Link, use the tags **X12IBTradeLinkChange** and **X12InboundGroupsAddChange**. You define the link to change using the **X12IBTradeLinkChange** tag, and add, change, or delete applications using the **X12InboundGroupsAddChangeDelete** tag.

When adding a Trade Link, one or more Applications (for example, 3010 PO) must be included in the XML data. The Application data structure does not differ by Known/Unknown status. For example, the XML structure for one Known Link having two Applications, and one Unknown Link having just one Application would be:

```
<x12IBTradeLink>
  ...header data
  <x12InboundGroups>
    ...application data #1
  </x12InboundGroups>
  <x12InboundGroups>
    ...application data #2
  </x12InboundGroups>
</x12IBTradeLink>
<x12Unknown_Link>
  ...header data
  <x12InboundGroups>
    ...application data
  </x12InboundGroups>
</x12Unknown_Link>
```

Note: ALL Transaction Sets for a given Application is automatically written to the Trade Link (into table **X12IBTrans**).

Trade Links are associated with the From and To Application Partners, and Application Partners are identified by GS Application ID. However, GS Application ID's are not unique across the system. There could be the same GS Application ID but for different Trading Partners. For this reason, both the ISA and GS Identifiers are required when specifying Trade Links (same applies for Outbound X12 Trade Links as well), and the Trading Partners must previously exist in the database for the Trade Link to be added.

The following table, field, and tag descriptions are for X12 inbound trade links:

- ["X12 Known Inbound Trade Link: \(Tag: X12IBTradeLink\)"](#)
- ["X12 Unknown Inbound Trade Link \(Tag: X12Unknown_Link\)"](#)
- ["X12 Unknown Inbound Trade Link Change \(Tag: X12Unknown_LinkChange\)"](#)
- ["X12 Inbound Trade Link Applications \(Tag: X12InboundGroups\)"](#)
- ["X12 Known Inbound Trade Link Change \(Tag: X12IBTradeLinkChange\)"](#)
- ["X12 Inbound Trade Links Applications Add, Change, Delete \(Tag: X12InboundGroupsAddChangeDelete\)"](#)
- ["X12 Known Inbound Trade Link Delete \(Tag: X12IBTradeLinkDelete\)"](#)
- ["X12 Unknown Inbound Trade Link Delete \(Tag: X12Unknown_LinkDelete\)"](#)
- ["X12 Known Inbound Trade Link: \(Tag: X12IBTradeLink\)"](#)
- ["X12 Unknown Inbound Trade Link \(Tag: X12Unknown_Link\)"](#)
- ["X12 Unknown Inbound Trade Link Change \(Tag: X12Unknown_LinkChange\)"](#)
- ["X12 Inbound Trade Link Applications \(Tag: X12InboundGroups\)"](#)
- ["X12 Known Inbound Trade Link Change \(Tag: X12IBTradeLinkChange\)"](#)
- ["X12 Inbound Trade Links Applications Add, Change, Delete \(Tag: X12InboundGroupsAddChangeDelete\)"](#)
- ["X12 Known Inbound Trade Link Delete \(Tag: X12IBTradeLinkDelete\)"](#)
- ["X12 Unknown Inbound Trade Link Delete \(Tag: X12Unknown_LinkDelete\)"](#)

X12 Known Inbound Trade Link: (Tag: X12IBTradeLink)

Use this tag to add a new X12 Known Inbound Trade Link. There is one **X12IBTradeLink** record per **Known Trade Link**. All fields are required.

The tag and field descriptions are described in the following table:

Field Label	FieldName/Tag	Req'd	Max. Len.	Validation	Notes
N/A (From ISA + IID)	From_ISA_IID	Yes	17	IchgQual + IchgID of From Trading Partner	1, 2
N/A (To ISA + IID)	To_ISA_IID	Yes	17	IchgQual + IchgID of To Trading Partner	3, 4
N/A (From GS Address)	From_GS_ID	Yes	15	GS_ApplicationID of From Ap Partner	5, 2
N/A (To GS Address)	To_GS_ID	Yes	15	GS_ApplicationID of To Ap Partner	6, 4
Get Post Office	GetMailPO	Yes	3	Nickname of Get PO	7

Notes:

1. Must match an existing External Trading Partner.
2. Combination of these two fields must match an External Trading Partner/Application Partner association (for example, the Application Partner must belong to the specified Trading Partner, not just be a valid Application Partner). This relationship can be in any External X12 Folder.
3. Must match an existing Internal Trading Partner.

4. Combination of these two fields must match an Internal Trading Partner/Application Partner association (for example, the Application Partner must belong to the specified Trading Partner, not just be a valid Application Partner). This relationship can be in any Internal X12 Folder.
5. Must match an existing External Application Partner.
6. Must match an existing Internal Application Partner.
7. Must be exactly 3 characters long. Nickname must exist as a Get Post Office.

X12 Unknown Inbound Trade Link (Tag: X12Unknown_Link)

Use this tag to add a new X12 Unknown Inbound Trade Link.

There is one **X12Unknown_Link** record per Unknown Trade Link. There is no From Partner in an Unknown Link.

The tag and field descriptions are described in the following table:

Field Label	FieldName/Tag	Req'd	Max. Len.	Validation	Notes
N/A (To ISA + IID)	To_ISA_IID	Yes	17	IchgQual + IchgID of To Trading Partner	1
N/A (To GS Address)	To_GS_ID	Yes	15	GS_ApplicationID of To Ap Partner	2
Get Post Office	GetMailPO	Yes	3	Nickname	3

Notes:

1. Must match an existing Internal Trading Partner.
2. Must match an existing Internal Application Partner.
3. Must be exactly 3 characters long. Nickname must exist as a Get Post Office.

X12 Unknown Inbound Trade Link Change (Tag: X12Unknown_LinkChange)

Use this tag to change an existing X12 Unknown Inbound Trade Link. To add, change, or delete applications on an existing X12 Unknown Trade Link, nest **X12InboundGroupsAddChangeDelete** records within this tag.

The tag and field descriptions are described in the following table:

Field Label	FieldName/Tag	Req'd	Max. Len.	Validation	Notes
N/A (To ISA + IID)	To_ISA_IID	Yes	17	IchgQual + IchgID of To Trading Partner	1
N/A (To GS Address)	To_GS_ID	Yes	15	GS_ApplicationID of To Ap Partner	2
Get Post Office	GetMailPO	No	3	Nickname	3

Notes:

1. Must match an existing Internal Trading Partner.
2. Must match an existing Internal Application Partner.
3. must be exactly 3 characters long, if specified. Nickname must exist as a Get Post Office.

X12 Inbound Trade Link Applications (Tag: X12InboundGroups)

Use this tag to add applications to a new X12 Inbound (Known or Unknown) Trade Link.

Note: Zero or more Applications per Link. This is optional, but without it the Trade Link is not complete (such links are valid in Partner Manager in the event you wish to complete it at a later time).

The tag and field descriptions are:

Field Label	FieldName/Tag	Req'd	Max. Len.	Validation	Notes
Application Version	GS_VersionRelease	Yes	12	fk to X12_Version	1
Functional ID	GS_FunctionalID	Yes	2	fk to X12_FuncID with Version	1
Ack Level	AckLevel	No	1	0,1,2,3,4	2
Ack To Application Partner	AckTo_ISA_IID and AckTo_GS_ID	No	17,15	See note	3
Validate By	ValidateByInd	No	1	F=Functional Group, T=Transaction Set I =Interchange Default: T	
Test Indicator	TestIndicator	No	1	T=True, F=False Default: F	4
Production Indicator	ProdIndicator	No	1	T=True, F=False Default: T	4
Test PO Nickname	TestPutPO	Yes if Test Indicator is T	3	Nickname of Valid Put PO	5

Field Label	FieldName/Tag	Req'd	Max. Len.	Validation	Notes
Prod PO Nickname	ProdPutPO	Yes if ProdIndicator is T	3	Nickname of Valid Put PO	6
HIPAA Validation Type	ValType	If HIPAA	7	See " HIPAA Validation Type Description "	
Map Name	Type3MapName	No	255	See " HIPAA Validation Type Description "	
997 Version	HIPAAAckFlag	No	12		7
Routing: Route Only	RouteOnly	No	1	Y/N Default: N	
4050X151 PI BIN Validation (824/999)	BinSegmentValidation	No	1		8
824 Ack Version	Version824Ack	No	12		9
AckGetPO		No	3		10
AckProdPutPO		No	3		11
AckTestPutPO		No	3		12

Notes:

1. Version + Functional ID must be unique within the same Trade Link.
2. **N/A** for Unknown Links; **0**=No Ack; **1**=Functional Group Level; **2**=Transaction Set Level; **3**=Segment Level; **4**=Data Element Level; Default is **0** (zero).
3. N/A for **Unknown Links**. Note the two **FieldNames/Tags**, both are required if **AckTo** is specified. Must point to a valid (existing) External Trading/Application Partner association. Defaults to the external (from) Ap Partner if not specified and the **AckLevel** (above) is not zero.
4. At least one of (**TestIndicator** or **ProdIndicator**) must be **T** (True).
5. Must be exactly 3 characters if present. Ignored if **TestIndicator** is **F** (False), required otherwise. Nickname must point to a valid Put Post Office.
6. Must be exactly 3 characters if present. Ignored if **ProdIndicator** is False, required otherwise. Nickname must point to a valid Put Post Office.
7. Set the **HIPAAAckFlag** to the 997 version of the Acknowledgement desired (if any). Defaults to the version specified in **GS_VersionRelease**. This field is useful for HIPAA data in particular, when the 997 must be a different version from the incoming data and may be used for non-HIPAA data as well.
8. Available for 4050X151 PI (275) documents when Type 4 Validation is selected for 824 or 999 reporting. Set to **N** for "no validation", **X** for "validate XML format only", or **S** for "validate XML format and against schema". The schema must be in the same directory as the Message Manager maps. The default value for this field is **N**.
9. Specifies the version of the Outbound 824 Acknowledgement that can be used for this inbound document. Applicable only if HIPAA type 3, type 4, and/or type 6 is enabled and reported as an 824. If not specified, the value will default to 4050. Valid values are 4010 or 4050, but others will be accepted if entries are manually added to the table **HIPAA824Versions**.
10. To automatically create an acknowledgement link, specify the Get post office nickname. The nickname will be used to create that acknowledgement. See [Automatic creation of X12 acknowledgement trade links](#).
11. To automatically create an acknowledgement link, specify at least one Put post office. You can specify the Production post office nickname here, and/or use **AckTestPutPO**. See [Automatic creation of X12 acknowledgement trade links](#).
12. To automatically create an acknowledgement link, specify at least one Put post office. You can specify the Test post office here, and/or use **AckProdPutPO**. See [Automatic creation of X12 acknowledgement trade links](#).
- [HIPAA validation type description](#)

HIPAA validation type description

This field is required if this Inbound Group is for a HIPAA Version; do not specify if this is not a HIPAA Inbound Group. This field is a 7 character position string, each position denoting the HIPAA Validation Type (for example, position 1 is for Type 1; position 3 is for Type 3). The values for these positions are as follows:

- position 1: **0** = Disabled, **2** = Enabled, Reported in 997
- position 2: 0 = Disabled, 1 = Enabled, 2=Enabled, reported in 999, 3=Enabled, Reported in 997.
- position 3: 0 = Disabled, 1 = Enabled, 2=Enabled, reported in 999, 3=Enabled, Reported in 824, 4=Enabled, Execute Map. If 4 is specified, the field **Type3MapName** must contain a value.
- position 4: **0** = Disabled, **1** = Enabled, 2=Enabled, reported in 999, 3=Enabled, Reported in 824.
- position 5: **0** = Disabled, **1** = Enabled, 2=Enabled, reported in 999, 3=Enabled, Reported in 997.
- position 6: **0** = Disabled, **1** = Enabled, 2=Enabled, reported in 999, 3=Enabled, Reported in 824.
- position 7: **0** = Disabled, **1** = Enabled, 2=Enabled, reported in 999, 3=Enabled, Reported in 824. You can only specify Type 7 (options **1** or **3**) if the External Application Partner for this link has a "HIPAA Institution" set.

The XML Autoload will perform the following validations:

- HIPAA Qualifier records exist when Types 3, 4, 6 and 7 are specified. These are loaded in Standards Maintenance.
- If a 997/999 is specified, an Acknowledgement Level (field **AckLevel**) has been selected.
- If an Acknowledgement Level is set, at least one of above selections denote a 997/999.
- If Types 3, 4, 5, 6, and 7 are set to anything other than **Disabled**, Type 2 must also not be disabled.

X12 Known Inbound Trade Link Change (Tag: X12IBTradeLinkChange)

Use this tag to change an existing X12 Inbound Trade Link. This tag is required to identify the Trade Link, and must exist even if you are not changing any data at this level. The fields in this tag are the same as X12IBTradeLink, but here the Get Post Office is optional. The Trade Link specified must exist.

The tag and field descriptions are:

Field Label	FieldName/Tag	Req'd	Max. Len.	Validation	Notes
-------------	---------------	-------	-----------	------------	-------

Field Label	FieldName/Tag	Req'd	Max. Len.	Validation	Notes
N/A (From ISA + IID)	From_ISA_IID	Yes	17	IchgQual + IchgID of From Trading Partner	1, 2
N/A (To ISA + IID)	To_ISA_IID	Yes	17	IchgQual + IchgID of To Trading Partner	3, 4
N/A (From GS Address)	From_GS_ID	Yes	15	GS_ApplicationID of From Ap Partner	5, 2
N/A (To GS Address)	To_GS_ID	Yes	15	GS_ApplicationID of To Ap Partner	6, 4
Get Post Office	GetMailPO	No	3	Nickname of Get PO	7

Notes:

1. Must match an existing External Trading Partner.
2. Must match an existing Internal Trading Partner.
3. Combination of these two fields must match an External Trading Partner/Application Partner association (for example, the Application Partner must belong to the specified Trading Partner, not just be a valid Application Partner). This relationship can be in any External X12 Folder.
4. Must match an existing External Application Partner.
5. Must match an existing Internal Application Partner.
6. Combination of these two fields must match an Internal Trading Partner/Application Partner association (for example, the Application Partner must belong to the specified Trading Partner, not just be a valid Application Partner). This relationship can be in any Internal X12 Folder.
7. If specified, must be exactly 3 characters long a

X12 Inbound Trade Links Applications Add, Change, Delete (Tag: X12InboundGroupsAddChangeDelete)

Use this tag to add, change or delete Applications on an existing X12 Inbound Trade Link. To add an application, set the **AddFlag** to Y. If you want to delete an existing Application, set the **DeleteFlag** to Y. Set both flags to N, or do not specify them at all, to change an existing Application.

This tag must be nested within the **X12IBTradeLinkChange** tag.

The fields in this tag are similar when adding using **X12InboundGroups**, with one major exception: The Test and Production Indicators can be set to R to remove one of those Post Offices. See Note 4 below for more information on this "Remove" feature. The tag and field descriptions are described in the following table:

Field Label	FieldName/Tag	Req'd	Max. Len.	Validation	Notes
Application Version	GS_VersionRelease	Yes	12	fk to X12_Version	1
Functional ID	GS_FunctionalID	Yes	2	fk to X12_FuncID with Version	1
Ack Level	AckLevel	No	1	0, 1, 2, 3, 4	2
Ack To Application Partner	AckTo_ISA_IID and AckTo_GS_ID	No	17,15	See note 3	3
Validate By	ValidateByInd	No	1	F=Functional Group, T=Transaction Set, I = Interchange	
Test Indicator	TestIndicator	No	1	T=True, F=False, R=Remove	4
Production Indicator	ProdIndicator	No	1	T=True, F=False, R=Remove	4
Test PO Nickname	TestPutPO	Yes if Test Indicator is T	3	Nickname of Valid Put PO	5
Prod PO Nickname	ProdPutPO	Yes if Prod Indicator is T	3	Nickname of Valid Put PO	6
HIPAA Validation Type	ValType	If HIPAA	7	See " HIPAA Validation Type Description "	
Map Name	Type3MapName		255		7
N/A	AddFlag	No	1	Y=Add as new Application	
N/A	DeleteFlag	No	1	Y=Delete the existing Application	8
997 Version	HIPAAAckFlag	No	12		9
Routing: Route Only	RouteOnly	No	1	Y/N Default: N	
4050X151 PI BIN Validation 9824/999	BinSegmentValidation	No	1		10
824 Ack Version	Version824Ack	No	12		11
AckGetPO		No	3		12
AckProdPutPO		No	3		13
AckTestPutPO		No	3		14

Notes:

1. **Note 1:** Version + Functional ID must be unique within the same Trade Link.
2. **Note 2: N/A** for Unknown Links; **0**=No Ack; **1**=Functional Group Level; **2**=Transaction Set Level; **3**=Segment Level; **4**=Data Element Level; Default is **0** (zero)
3. N/A for Unknown Links. Note the two FieldNames/Tags, both are required if **AckTo** is specified. Must point to a valid (existing) External Trading/Application Partner association. Defaults to the external (from) Ap Partner if not specified and the **AckLevel** (above) is not zero.
4. At least one of (**TestIndicator** or **ProdIndicator**) on this Application must end up being True when the change completes. If you want to remove an existing Test or Production Post Office, set this field to "R". For example, this is available in the event you want to change an existing group from Test only to Production Only: set the **TestIndicator** to R, which will remove the Test Post Office, and set the **Production Indicator** to T and specify the new Production Post Office in **ProdPutPO**. When setting this field to "R", you must specify the other (test/production) Post Office if one does not already exist on the group.
5. Must be exactly 3 characters if present. Ignored if **TestIndicator** is False. Nickname must point to a valid Put Post Office.
6. Must be exactly 3 characters if present. Ignored if **ProdIndicator** is False. Nickname must point to a valid Put Post Office.

7. The **Type3MapName** is required if position 3 of **ValType** is 4. This means that you must perform HIPAA Type 3 validation using this map name. This map must be in the Trading Manager **mmgr** directory, and is a custom map supplied by you. The map name is not validated by the XML Autoload Utility, as it may be on another system (for example UNIX).
8. Set the **DeleteFlag** to **Y** if you wish to remove an Inbound Group from an existing Trade Link. Specify only the **GS_VersionRelease**, **GS_FunctionalID**, and this flag when deleting a group; all other fields will be ignored.
9. Set the **HIPAAAckFlag** to the 997 version of the Acknowledgement desired (if any). When adding a new group, this field defaults to the version of the **GS_VersionRelease** field specified. This field is useful for HIPAA data in particular, when the 997 must be a different version from the incoming data and may be used for non-HIPAA data as well.
10. Available for 4050X151 PI (275) documents when Type 4 Validation is selected for 824 or 999 reporting. Set to **N** for "no validation", **X** for "validate XML format only", or **S** for "validate XML format and against schema". The schema must be in the same directory as the Message Manager maps. The default value for this field is **N**.
11. Specifies the version of the Outbound 824 Acknowledgement that can be used for this inbound document. Applicable only if HIPAA type 3, type 4, and/or type 6 is enabled and reported as an 824. If not specified, the value will default to 4050. Valid values are 4010 or 4050, but others will be accepted if entries are manually added to the table HIPAA824Versions.
12. To automatically create an acknowledgement link, specifically the Get Post Office nickname. The nickname will be used to create that acknowledgement. See [Automatic creation of X12 acknowledgement trade links](#)
13. To automatically create an acknowledgment link, specify at least one Put post office. You can specify the Production post office nickname here, and/or use **AckTestPutPO**. See [Automatic creation of X12 acknowledgement trade links](#).
14. To automatically create an acknowledgement link, specify at least one Put post office. You can specify the test post office nickname here, and/or use **AckProdPutPO**. See [Automatic creation of X12 acknowledgement trade links](#).

X12 Known Inbound Trade Link Delete (Tag: X12IBTradeLinkDelete)

Use this tag to delete an existing X12 Inbound Trade Link. Supply the Identifiers of the "From" and "To" Trading and Application Partners (see: **From_ISA_IID**, **To_ISA_IID**, **From_GS_ID**, and **To_GS_ID**). This tag deletes the entire trade link which includes all applications and transaction sets associated with the link. A warning report (or error if **Abort Run** is selected) is issued if Trade Link does not exist. The tag and field descriptions are:

Field Label	FieldName/Tag	Req'd	Max Len.	Validation	Notes
N/A (From ISA + IID)	From_ISA_IID	Yes	17	IchgQual + IchgID of From Trading Partner	1, 3
N/A (To ISA + IID)	To_ISA_IID	Yes	17	IchgQual + IchgID of To Trading Partner	2, 6
N/A (From GS Address)	From_GS_ID	Yes	15	GS_ApplicationID of From Ap Partner	4, 3
N/A (To GS Address)	To_GS_ID	Yes	15	GS_ApplicationID of To Ap Partner	5, 6

Notes:

1. Must match an existing External Trading Partner.
2. Must match an existing Internal Trading Partner.
3. Combination of these two fields must match an External Trading Partner/Application Partner association (for example, the Application Partner must belong to the specified Trading Partner, not just be a valid Application Partner).
4. Must match an existing External Application Partner.
5. Must match an existing Internal Application Partner.
6. Combination of these two fields must match an Internal Trading Partner/Application Partner association (for example, the Application Partner must belong to the specified Trading Partner, not just be a valid Application Partner).

X12 Unknown Inbound Trade Link Delete (Tag: X12Unknown_LinkDelete)

Use this tag to delete an existing X12 Unknown Inbound Trade Link. Supply the Identifiers of the "To" Trading and Application Partners (see **X12Unknown_Link**, **To_ISA_IID** and **To_GS_ID**). This tag deletes the entire trade link which includes all applications and transaction sets associated with the link. A warning report (or error if **Abort Run** is selected) is issued if Trade Link does not exist. The tag and field descriptions are described in the following table:

Field Label	FieldName/Tag	Req'd	Max. Len.	Validation	Notes
N/A (To ISA + IID)	To_ISA_IID	Yes	17	IchgQual + IchgID of To Trading Partner	1
N/A (To GS Address)	To_GS_ID	Yes	15	GS_ApplicationID of To Ap Partner	2

Notes:

1. Must match an existing Internal Trading Partner.
2. Must match an existing Internal Application Partner.

X12 outbound trade links

Trade links can be added, changed, or deleted. In addition, existing Trade Links can have their individual Applications (also known as "Documents") added, changed, or deleted.

To add a new X12 Outbound trade link, use the following tags:

- [X12OBTPLink](#)
- [X12OBApPartnerLnk](#)
- [X12OBAppLink](#)

To change an existing X12 Outbound Trade Link, use the tags:

- [X12OBTPLinkChange](#)
- [X12OBApPartnerLnkChange](#)
- [X12OBAppLinkAddChangeDelete](#)

To delete an entire X12 Outbound Trade Link, see [X12OBTradeLinkDelete](#).

There are 4 levels of data in an Outbound Trade Link:

- The Trading Partner (ISA)
- Application Partner (GS)
- Applications (Versions/Functional ID's)
- Transaction Levels (ST)

Unlike Inbound Trade Links, there is no such thing as an Unknown Outbound Trade Link, and like Inbound Trade Links, the lowest level (transactions) is not supported in the AutoLoad input; all transactions for a given Version/Functional ID is written to the Transaction tables (with a blank ST Reset value). In the unusual case the ST Reset values are required on an Outbound Trade Link use Partner Manager after the AutoLoad is completed.

Complicating this structure is the fact that the Trading Partner level (ISA) may already exist, but for different Application Partners (GS) than the one being added by the AutoLoad Utility. If this is detected, then any entry in the Trading Partner level is ignored, a warning is issued, and the Link is added. Note the Trading Partner level is required in the input file even if the user knows this level exists.

Nesting is similar to Inbound Trade Links. If multiple Application Partners are specified, they have to be nested under the Trading Partner, and if multiple Applications (Versions/Functional ID's) are specified then they have to be nested under the Application Partner. For example, one trade link, two application partners, each having one application during an "add" operation:

```
<X12OBTPLink>
  <X12OBApPartnerLnk>
    <X12OBAppLink>
      </X12OBAppLink>
    </X12OBApPartnerLnk>
    <X12OBApPartnerLnk>
      <X12OBAppLink>
        </X12OBAppLink>
    </X12OBApPartnerLnk>
  </X12OBTPLink>
```

Adding an Application (also known as a Document) to an existing Trade Link is supported in the following tags:

- [X12OBTPLinkChange](#)
- [X12OBApPartnerLnkChange](#)
- [X12OBAppLinkAddChangeDelete](#)

During a change operation, you can change the Trading Partner Level, change the Application Partner Level, or add, change, or delete the Application level.

Example nesting of these "Change" tags is:

```
<X12OBTPLinkChange> (required)
<X12OBApPartnerLnkChange>
  (optional, but required if changing a document)
<X12OBAppLinkAddChangeDelete>
  (optional Document level)
</X12OBAppLinkAddChangeDelete>
</X12OBApPartnerLnkChange>
</X12OBTPLinkChange>
```

The following table, field, and tag descriptions are for X12 outbound trade links:

- ["X12 Outbound Trading Partner Trade Links \(Tag: X12OBTPLink\)"](#)
- ["X12 Outbound Application Partner Trade Links \(Tag: X12OBApPartnerLnk\)"](#)
- ["X12 Outbound Applications Trade Links \(Tag: X12OBAppLink\)"](#)
- ["X12 Outbound Trading Partner Trade Link Delete \(Tag: X12OBTradeLinkDelete\)"](#)
- ["X12 Outbound Trading Partner Trade Links \(Tag: X12OBTPLink\)"](#)
- ["X12 Outbound Application Partner Trade Links \(Tag: X12OBApPartnerLnk\)"](#)
- ["X12 Outbound Applications Trade Links \(Tag: X12OBAppLink\)"](#)
- ["X12 Outbound Trading Partner Trade Links Change \(Tag: X12OBTPLinkChange\)"](#)
- ["X12 Outbound Application Partner Trade Links Change \(Tag: X12OBApPartnerLnkChange\)"](#)
- ["X12 Outbound Applications Trade Links Add/Change/Delete \(Tag: X12OBAppLinkAddChangeDelete\)"](#)
- ["X12 Outbound Trading Partner Trade Link Delete \(Tag: X12OBTradeLinkDelete\)"](#)

X12 Outbound Trading Partner Trade Links (Tag: X12OBTPLink)

Use this tag to add a new X12 Outbound Trade Link.

Note: One **X12OBTPLink** record per Outbound Trade Link is required.

The tag and field descriptions are described in the following table:

Field Label	FieldName/Tag	Req'd	Max. Len.	Validation	Note
N/A (From ISA + IID)	From_ISA_IID	Yes	17	IchgQual + IchgID of From Trading Partner	1
N/A (To ISA + IID)	To_ISA_IID	Yes	17	IchgQual + IchgID of To Trading Partner	2
Interchange Control # Scheme	ISAControlNoScheme	No	1	U = User Defined, I = Increment Default: I	
Interchange Control # Map	ISAControlNoMap	Yes, if scheme is "U"	255	Not validated	3
Group Control # Scheme	GSControlNoScheme	No	1	U,T,A,F, R See note 4 Default: R	4
Group Control # Map	GSControlNoMap	Yes, if scheme is "U"	255	Not validated	5
Reset: Interchange Control # (ISA)	ISAControlNoValue	No	9	Numeric, positive integer	
Reset: Group Control # (GS)	GSControlNoValue	No	10	Numeric, positive integer specified here ONLY is GS Scheme is T or U.	6

Notes:

1. Must match an existing Internal Trading Partner.
2. Must match an existing External Trading Partner.
3. Required only if **ISAControlNoScheme** is **U**. Typically, the **.mmc** file is used here. For the purposes of **AutoLoad**, this filename is not validated (Partner Manager does insist it can find this file), except to check for single/double quotes (not allowed).
4. **U**=User Defined; **T**=Increment by Trading Partner; **A**=Increment by Application Partner; **F**=Increment by Functional ID; **R**=Relative (the default if not specified)
5. Required only if **GSControlNoScheme** is **U**. Typically, the **.mmc** file is used here. For the purposes of AutoLoad, this filename is not validated (Partner Manager does insist it can find this file), except to check for single/double quotes (not allowed).
6. Only include if GS Scheme is **T** (Increment by Trading Partner), or **U** (User Defined via map). **ResetGS_Ind** is set in the table to True if a value is present here. These two fields are only stored in X12OBTPLink if the GS Scheme is **T** or **U**, otherwise, they are stored in the GS Level (**X12OBAppPartnerLnk**), if Scheme is **A**; otherwise, it is at the Application Level (Scheme is **F**). If included here and the GS Scheme is not in (**T**, **U**) then the entire link is rejected. A GS Scheme of **R** (Relative) means that no GS Reset Value is ever entered; any present is ignored.

X12 Outbound Application Partner Trade Links (Tag: X12OBAppPartnerLnk)

Use this tag to specify the Application Partners on a new X12 Outbound Trade Link. See the related tag [X12OBAppPartnerLnkChange](#), if changing an existing Trade Link. One or more X12OBAppPartnerLnk record(s) per Outbound Trade Link is required. If more than one Application Partner combination is defined, then the next level (Application) must be nested within this tag.

The tag and field descriptions are described in the following table:

Field Label	FieldName/Tag	Req'd	Max. Len.	Validation	Notes
N/A (From GS App ID)	From_GS_ID	Yes	15	See note 1	1
N/A (To GS App ID)	To_GS_ID	Yes	15	See note 2	2
Reset: Group Control # (GS)	GSControlNoValue	No	9	Numeric, positive integer. Only if scheme is A .	3
Get Post Office	GetMailPO	Yes	3	Get PO	4

Notes:

1. Must point to a valid Internal Application Partner that belongs to the Internal Trading Partner specified at the level above.
2. Must point to a valid External Application Partner that belongs to the External Trading Partner specified at the level above.
3. Optionally, specify only if the GS Scheme, specified in the level above (X12OBTPLink), is **A**.
4. Must be exactly 3 characters in length and exist as a Get Post Office Nickname.

X12 Outbound Applications Trade Links (Tag: X12OBAppLink)

Note: If you have used the XML AutoLoad Utility previously, be aware that in this version of the product, a test post office has been introduced to X12 outbound links. The existing post office field, PutMailPO, is now the production post office. If you make no changes to your XML script generation, the post office that you have used previously for X12 Outbound Links will be this production post office. See the fields TestIndicator, ProdIndicator and TestPutPO fields in the tag X12OBAppLink for more information. If you do not want to take advantage of these new test vs. production post offices, set the two post offices to the same value.

Use this tag to add Applications (documents) to a new X12 Outbound Trade Link. To add, change, or delete Applications on an existing Trade Link, see the related tag: [X12OBAppLinkAddChangeDelete](#). One or more Applications per Trade Link are required. At least one record is required per link. The tag and field descriptions are described in the following table:

Field Label	FieldName/Tag	Req'd	Max. Len.	Validation	Notes
Application Version	GS_VersionRelease	Yes	12	fk to X12_Version	1
Functional ID	GS_FunctionalID	Yes	2	fk to X12_FuncID with Version	1
Reset GS Control # Value	GSControlNoValue	No	10	Only if GS Scheme is "F". Numeric, positive integer	2

Field Label	FieldName/Tag	Req'd	Max. Len.	Validation	Notes
ST Control Scheme	STControlNoScheme	No	1	U, R, or T	3 Default: R
ST Control Map	STControlNoMap	No	255	Only if ST Scheme is U	
Put Post Office	PutMailPO	Yes	3	Put PO	4, 11
Validate By	ValidateByInd	No	1	F, T, or I	5 Default: I
Acknowledgement Hours	ACK_Within	No	5	> 0 and < 99999	6 Default: 0
Application Format	Leave_Intact	No	1	Y=Fully N=Partial M=Control Numbers Only (aka "Mostly")	Default: N
Routing: RouteOnly	RouteOnly	No	1	Y/N	Default: N
GS Time Stamp Option	TimeStampOption	No	1	0, 1, 2, 3	7
Test Indicator	TestIndicator	No	1	T=True, F=False Default: F	4
Production Indicator	ProdIndicator	No	1	T=True, F=False Default: F	4
Test PO Nickname	TestPutPO	Yes if Test Ind. is T	3	Nickname of Valid Put PO	5
AckGetPO		No	3		8
AckProdPutPO		No	3		9
AckTestPutPo		No	3		10

Notes:

1. Version/Functional ID is verified to be unique within this Application Partner Link.
2. GS Scheme, specified at the Trading Partner level, must be F (functional group) to specify GS Reset values here.
3. **U**=User Defined; **R**=relative; **T**=Increment by Transaction. If **U** then **STMap** is required.
4. Must be exactly 3 characters in length and exist as a Put Post Office Nickname.
5. Valid values, if specified are **F**=Functional Group, **T**= Transaction Set, **I**=interchange. Defaults to **I**.
6. Number of hours an acknowledgement is expected.
7. **0**=hhmm, **1**=hhmmss, **2**=hhmmssd, **3**=hhmmssdd. If not specified, the default will be zero (hhmm) if the version is less than 4010, otherwise the default is one (hhmmss)
8. To automatically create an acknowledgement link, specify the Get Post Office nickname. That nickname will be used to create the acknowledgement. See [Automatic creation of X12 acknowledgement trade links](#).
9. To automatically create an acknowledgement link, specify at least one Put post office. You can specify the Production post office nicnname here, and/or use AckTestPutPO. See [Automatic creation of X12 acknowledgement trade links](#).
10. To automatically create an acknowledgement link, specify at least one Put post office. You can specify the Test Post Office nickname here and/or use AckProdPutPO. See [Automatic creation of X12 acknowledgement trade links](#).
11. This is the production post office.

X12 Outbound Trading Partner Trade Links Change (Tag: X12OBTPLinkChange)

Use this tag to change an existing X12 Outbound Trade Link. At a minimum, specify the unique Identifier of the two Trading Partners (**From_ISA_IID** and **To_ISA_IID**). The remaining fields need not be specified if your changes are at a lower level (Application Partner or Document). Note changes here will impact all the Trade Links defined for these two Trading Partners, even if multiple Application Partner-level Trade Links are defined. Changes here relate to the "ISA Control" popup, and if the GS Control Scheme is "Trading Partner", the "GS Control" popup found in Partner Manager. If the desired changes are only at this level, the remaining tags need not be specified. The tag and field descriptions are:

Field Label	FieldName/Tag	Req'd	Max. Len.	Validation	Notes
N/A (From ISA + IID)	From_ISA_IID	Yes	17	IchgQual + IchgID of From Trading Partner	1
N/A (To ISA + IID)	To_ISA_IID	Yes	17	IchgQual + IchgID of To Trading Partner	2
Interchange Control # Scheme	ISAControlNoScheme	No	1	U = User Defined, I = Increment	
Interchange Control # Map	ISAControlNoMap	Yes, if scheme is "U"	255	Not validated	3
Group Control # Scheme	GSControlNoScheme	No	1	U,T,A,F,R See note	4
Group Control # Map	GSControlNoMap	Yes, if scheme is "U"	255	Not validated	5
Reset: Interchange Control # (ISA)	ISAControlNoValue	No	9	Numeric, positive integer	
Reset: Group Control # (GS)	GSControlNoValue	No	10	Numeric, positive integer specified here ONLY is GS Scheme is T or U.	6

Notes:

1. Must match an existing Internal Trading Partner.
2. Must match an existing External Trading Partner.
3. Required only if **ISAControlNoScheme** is **U**. Typically, the **.mmc** file is used here. For the purposes of **AutoLoad**, this filename is not validated (Partner Manager does insist it can find this file), except to check for single/double quotes (not allowed).

4. **U**=User Defined; **T**=Increment by Trading Partner; **A**=Increment by Application Partner; **F**=Increment by Functional ID; **R**=Relative (the default if not specified)
5. Required only if **GSControlNoScheme** is **U**. Typically, the **.mme** file is used here. For the purposes of **AutoLoad**, this filename is not validated (Partner Manager does insist it can find this file), except to check for single/double quotes (not allowed).
6. Only include if GS Scheme is **T** (Increment by Trading Partner), or **U** (User Defined via map). These two fields are only stored in **X12OBTPLink** if the GS Scheme is **T** or **U**, otherwise, they are stored in the GS Level (**X12OBAppPartnerLnk**), if Scheme is **A**; otherwise, it is at the Application Level (Scheme is **F**). If included here and the GS Scheme is not in (**T**, **U**) then the entire link is rejected. A GS Scheme of **R** (Relative) means that no GS Reset Value is ever entered; any present is ignored.

X12 Outbound Application Partner Trade Links Change (Tag: **X12OBAppPartnerLnkChange**)

Use this tag to change the Application Partner level information on an X12 Outbound Trade Link, or if changing the Document Level information on an existing X12 Outbound Trade Link. The Application Partner Level must already exist; to add one, the tags [X12OBTPLink](#), [X12OBAppPartnerLnk](#), and [X12OBAppLink](#) must be used instead.

The fields are identical to the [X12OBAppPartnerLnk](#) tag, except **GetMailPO** is optional. The tag and field descriptions are:

Field Label	FieldName/Tag	Req'd	Max. Len.	Validation	Notes
N/A (From GS App ID)	From_GS_ID	Yes	15	See note 1	1
N/A (To GS App ID)	To_GS_ID	Yes	15	See note 2	2
Reset: Group Control # (GS)	GSControlNoValue	No	9	Numeric, positive integer. Only if scheme is A.	3
Get Post Office	GetMailPO	No	3	Get PO	4

Notes:

1. Must point to a valid Internal Application Partner that belongs to the Internal Trading Partner specified at the level above.
2. Must point to a valid External Application Partner that belongs to the External Trading Partner specified at the level above.
3. Optionally, specify only if the GS Scheme, specified in the level above (X12OBTPLink) is **A**.
4. If specified, must be exactly 3 characters in length and exist as a Get Post Office Nickname.

X12 Outbound Applications Trade Links Add/Change/Delete (Tag: **X12OBAppLinkAddChangeDelete**)

Use this tag to add, change, or delete Applications on an existing X12 Outbound Trade Link. This tag is always nested within an [X12OBAppPartnerLnkChange](#) tag.

Specify the unique identifier for the Application: **GS_VersionRelease** and **GS_FunctionalID**. If you are adding, set the **AddFlag** to **Y**. If you want to delete just this Document from the Trade Link, set the **DeleteFlag** to **Y**. Deleting a record requires tags **GS_VersionRelease** and **GS_FunctionalID**; any values specified in the other tags will be ignored. The tag and field descriptions are described in the following table:

Field Label	FieldName/Tag	Req'd	Max. Len.	Validation	Notes
Application Version	GS_VersionRelease	Yes	12	fk to X12_Version	1
Functional ID	GS_FunctionalID	Yes	2	fk to X12_FuncID with Version	1
Reset GS Control # Value	GSControlNoValue	No	10	Only if GS Scheme is "F". Numeric	2
ST Control Scheme	STControlNoScheme	No	1	U, R, or T	3
ST Control Map	STControlNoMap	No	255	Only if ST Scheme is U	
Put Post Office	PutMailPO	No	3	Put PO	4, 11
Validate By	ValidateByInd	No	1	F, T, or I	5
Acknowledgement Hours	ACK_Within	No	5	> 0 and < 99999	6
N/A	AddFlag	No	1	Y or N	
N/A	DeleteFlag	No	1	Y or N	
Application Format	Leave_Intact	No	1	Y=Fully N=Partial M=Control Numbers Only (aka "Mostly")	Default: N
Routing: RouteOnly	RouteOnly	No	1	Y/N Default: "N"	
GS Time Stamp Option	TimeStampOption	No	1	0, 1, 2, 3	7
Test Indicator	TestIndicator	No	1	T=True, F=False Default: F	4
ProdIndicator	ProdIndicator	No	1	T=True, F=False Default: T	4
Test PO Nickname	TestPutPO	Yes if Test Ind.is T	3	Nickname of Valid Put PO	6
AckGetPO		No	3		8
AckProdPutPO		No	3		9
AckTestPutPO		No	3		10

Notes:

1. Version/Functional ID is verified to be unique within this Application Partner Link.
2. GS Scheme, specified at the Trading Partner level, must be **F** (functional group) to specify GS Reset values here.
3. User Defined; **R**=relative; **T**=Increment by Transaction. If **U** then **STMap** is required.
4. If specified must be exactly 3 characters in length and exist as a Put Post Office Nickname. Required if adding.
5. Valid values (if specified) are: **F**=Functional Group; **T**=Transaction Set; **I**=interchange. Defaults to **I**.
6. Number of hours an acknowledgement is expected.
7. **0**=hhmm, **1**=hhmmss, **2**=hhmmssd, **3**=hhmmssdd. If not specified, the default will be zero (hhmm) if the version is less than 4010, otherwise the default is one (hhmmss)
8. To automatically create an acknowledgement link, specify the Get post office nickname. This nickname will be used to create that acknowledgement. See [Automatic creation of X12 acknowledgement trade links](#)
9. To automatically create an acknowledgement link, specify at least one Put post office. You can specify the production post office nickname here, and/or use **AckTestPutPO**. See [Automatic creation of X12 acknowledgement trade links](#)
10. To automatically create an acknowledgement link, specify at least one Put post office. You can specify the test post office nickname here, and/or use **AckProdPutPO**. See [Automatic creation of X12 acknowledgement trade links](#)
11. This is the production Put post office.

X12 Outbound Trading Partner Trade Link Delete (Tag: X12OBTradeLinkDelete)

Use this tag to delete an existing X12 Outbound Trade Link. Supply the Identifiers of the "From" and "To" Trading and Application Partners (see [X12OBTPLink](#)): **From_ISA_IID**, **To_ISA_IID**, **From_GS_ID**, and **To_GS_ID**. This tag deletes the entire trade link which includes all applications and transaction sets associated with the link. A warning report (or error if Abort Run is selected) is issued if Trade Link does not exist. The tag and field descriptions are described in the following table:

Field Label	FieldName/Tag	Req'd	Max. Len.	Validation	Notes
N/A (From ISA + IID)	From_ISA_IID	Yes	17	IchgQual + IchgID of From Trading Partner	1, 2
N/A (To ISA + IID)	To_ISA_IID	Yes	17	IchgQual + IchgID of To Trading Partner	3, 4
N/A (From GS Address)	From_GS_ID	Yes	15	GS_ApplicationID of From Ap Partner	5, 2
N/A (To GS Address)	To_GS_ID	Yes	15	GS_ApplicationID of To Ap Partner	6, 4

Notes:

1. Must match an existing Internal Trading Partner.
2. Combination of these two fields must match an Internal Trading Partner/Application Partner association (for example, the Application Partner must belong to the specified Trading Partner, not just be a valid Application Partner).
3. Must match an existing External Trading Partner.
4. Combination of these two fields must match an External Trading Partner/Application Partner association (for example, the Application Partner must belong to the specified Trading Partner, not just be a valid Application Partner).
5. Must match an existing Internal Application Partner.
6. Must match an existing External Application Partner.

EDIFACT partners

The following table, field, and tag descriptions are for EDIFACT Trading Partners:

- ["EDIFACT Trading Partners \(Tag: EDFTradingPartners\)"](#)
- ["EDIFACT Trading Partners Change \(Tag: EDFTradingPartnersChange\)"](#)
- ["EDIFACT Trading Partners Delete \(Tag: EDFTradingPartnersDelete\)"](#)
- ["EDIFACT Application Partners \(Tag: EDFAppPartners\)"](#)
- ["EDIFACT Application Partners Change \(Tag: EDFAppPartnersAddChange\)"](#)
- ["EDIFACT Application Partners Delete \(Tag: EDFAppPartnersDelete\)"](#)
- ["EDIFACT Trading Partners \(Tag: EDFTradingPartners\)"](#)
- ["EDIFACT Trading Partners Change \(Tag: EDFTradingPartnersChange\)"](#)
- ["EDIFACT Trading Partners Delete \(Tag: EDFTradingPartnersDelete\)"](#)
- ["EDIFACT Application Partners \(Tag: EDFAppPartners\)"](#)
- ["EDIFACT Application Partners Change \(Tag: EDFAppPartnersAddChange\)"](#)
- ["EDIFACT Application Partners Delete \(Tag: EDFAppPartnersDelete\)"](#)

EDIFACT Trading Partners (Tag: EDFTradingPartners)

Use this tag to add new EDIFACT Trading Partners. EDIFACT Trading Partners can have Contacts added as well.

The tag and field descriptions are described in the following table:

Field Label	FieldName/Tag	Req'd	Max. Len.	Validation	Notes
Internal/External	IntExt	Yes	1	"I" or "E"	1 Determines Folder
Partner Name	PartnerName	Yes	35	Unique	2
Syntax ID	SyntaxID	No	4	fk to EDFSyntaxID (UNOA-H)	3
Syntax Version	SyntaxVer	No	10	>0 (Numeric; decimals allowed) if specified	4 Default: 1

Field Label	FieldName/Tag	Req'd	Max. Len.	Validation	Notes
Partner ID Code Qualifier	IchgIdCdQual	No	4	fk to EDFIdCdQual	5 Defaults to 4 spaces (None)
Interchange ID	IchgID	Yes	35	unique; see note; Cannot be "UNKNOWN" (lower or upper case)	5
Routing Address	RoutingAddress	No	14	See note	5
Password	Password	No	14	External only	6
Password Qualifier	PasswordQual	No	2	External only; "AA" or "BB"	6
Enveloping Option	Leave_Intact	No	1	F= Fully, P= Partial. Internal partners only. Default: "P"	
Use UNA Segment	AddUNASeg	No	1	"T" = True; "F" = False Default: "F"	
Component Data Element Separator	ComponentDESep	No	2	fk to EDFHexValues	7
Data Element Separator	DESeparator	No	2	fk to EDFHexValues	7
Decimal Notation Separator	DecimalNotation	No	2	fk to EDFHexValues	7
Release Indicator Separator	ReleaseIndicator	No	2	fk to EDFHexValues	7
Segment Terminator Separator	SegmentTerminator	No	8	fk to EDFHexValues	7
Reserved Character Separator	ReservedCharacter	No	2	fk to EDFHexValues	7
Comm Agreement ID	CommAgreementID	No	35		
Duplicate Control	Duplicate_Control	No	1	N = none I = Interchange External Only Default: "N"	
Last UNB Control #	UNBControlNoValue	No	14		8

Notes:

- The standard Internal and External EDIFACT folders is used (no custom folders). These custom folders are: -3=Internal FolderNo and -23=External FolderNo.
- Do not use the name "UNKNOWN, or "ALL", or any variation of the case of these two words. For example, do not use "Unknown", or "All". Unique across all EDIFACT Trading Partners.
- Defaults to UNOA. If specified, must be in table EDFSyntaxID, which currently contains UNOA, UNOB, UNOC, UNOD, UNOE, UNOF, UNOG, and UNOH.
- Default is 1. If specifying, must be an integer from 1 to 9.
- Cannot contain single or double quotation marks. Combination of IchgIdCdQual+ IchgID + RoutingAddress must be unique across all EDIFACT Trading Partners.
- Password and Password Qualifier must only be specified for External Trading Partners.
- The default values are: ComponentDESep = 3A, DESeparator = 2B, SegmentTerminator = 27, ReleaseIndicator = 3F, DecimalNotation = 2C, and ReservedCharacter = 20.
- This field can only be set for External Trading Partners that have the Duplicate Control flag set to I (Interchange). If set, it must be a positive integer no longer than nine characters in length.

EDIFACT Trading Partners Change (Tag: EDFTradingPartnersChange)

Use this tag to change an existing EDIFACT Trading Partner. The fields in this tag are similar to EDFTradingPartners, but the **IntExt** (Internal/External) field is not specified because you cannot change an Internal Trading Partner to an External using this tag. If this is required, delete and re-add the partner. Also, unlike the tag EDFTradingPartners, EDFAppPartners are not nested within this tag.

To specify the Trading Partner to change, supply **PartnerName**, or the codes that create the unique identifier of an EDIFACT Trading Partner: **IchgIdCdQual** plus **IchgID**, and if required, the **RoutingAddress**. You can also supply both the Partner Name and the unique identifier. The **RoutingAddress** is required if more than one Partner has the same **IchgIdCdQual** and **IchgID**. To change the fields that identify a Partner (**PartnerName**, **IchgIdCdQual**, **IchgID**, or **RoutingAddress**) use the fields **PartnerNameChange**, **IchgIdCdQualChange**, **IchgIDChange**, or **RoutingAddressChange**.

Trading Partners can have [Contacts](#) added at the same time; nest the Contact record within the Trading Partner tags (see "[Contacts](#)"). Use this tag, along with the Contacts tag, to add Contacts to an existing Trading Partner.

The tag and field descriptions are described in the following table:

Field Label	FieldName/Tag	Req'd	Max. Len.	Validation	Notes
Partner Name	PartnerName	No	35	Unique	1
Syntax ID	SyntaxID	No	4	fk to EDFSyntaxID (UNOA-H)	2
Syntax Version	SyntaxVer	No	10	>0 (Numeric; decimals allowed) if specified	3
Partner ID Code Qualifier	IchgIdCdQual	No	4	fk to EDFIdCdQual	4
Interchange ID	IchgID	Yes	35	unique; see note; Cannot be "UNKNOWN" (lower or upper case)	4
Routing Address	RoutingAddress	No	14	See note	4
Password	Password	No	14	External only	5
Password Qualifier	PasswordQual	No	2	External only; "AA" or "BB"	5

Field Label	FieldName/Tag	Req'd	Max. Len.	Validation	Notes
Enveloping Option	Leave_Intact	No	1	F= Fully, P= Partial Internal partners only.	
Use UNA Segment	AddUNASeg	No	1	T=True, F=False	
Component Data Element Separator	ComponentDESep	No	2	fk to EDFHexValues	
Data Element Separator	DESeparator	No	2	fk to EDFHexValues	
Decimal Notation Separator	DecimalNotation	No	2	fk to EDFHexValues	
Release Indicator Separator	ReleaseIndicator	No	2	fk to EDFHexValues	
Segment Terminator Separator	SegmentTerminator	No	8	fk to EDFHexValues	
Reserved Character Separator	ReservedCharacter	No	2	fk to EDFHexValues	
Comm Agreement ID	CommAgreementID	No	35		
Duplicate Control	Duplicate_Control	No	1	N = none, I = Interchange External Only	
Last UNB Control #	UNBControlNoValue	No	14		6
N/A	PartnerNameChange	No	35		7
N/A	IchgIdCdQualChange	No	4		7
N/A	IchgIDChange	No	35		7
N/A	RoutingAddressChange	No	14		7
Reset UNB Control	ResetUNB_Ind	No	T/F		8

Notes:

1. Do not use the name "UNKNOWN, or "ALL", or any variation of the case of these two words. For example, do not use "Unknown", or "All". Unique across all EDIFACT Trading Partners.
2. If specified, must be in table EDFSyntaxID, which currently contains UNOA, UNOB, UNOC, UNOD, UNOE, UNOF, UNOG, and UNOH.
3. If specifying, must be an integer from 1 to 9.
4. Cannot contain single or double quotation marks. Combination of IchgIdCdQual+ IchgID + RoutingAddress must be unique across all EDIFACT Trading Partners.
5. Password and Password Qualifier must only be specified for External Trading Partners.
6. This field can only be set for External Trading Partners that have the Duplicate Control flag set to I (Interchange). If set, it must be a positive integer no longer than nine characters in length.
7. Use these fields to change the unique identifiers of an EDIFACT Trading Partner. Any, all, or none of these "change" fields can be specified. The resulting name/identifier must not already exist in the database.
8. Valid values are T or F (T =True/enabled, F=False/disabled). If setting to T, you must specify UNBControlNoValue (on screen as "Last UNB Control Number"); if you specified an UNBControlNoValue this field will be set to T by default. If setting to F, this will clear the existing UNBControlNoValue. You can only specify this field on external Trading Partners.

EDIFACT Trading Partners Delete (Tag: EDFTradingPartnersDelete)

Use this tag to delete an EDIFACT Trading Partner.

Use with caution, deleting a Trading Partner will delete all Application Partners, Contacts, and Trade Links associated with that Trading Partner.

Supply **PartnerName**, or the codes that create the unique identifier of an EDIFACT Trading Partner: **IchgIdCdQual** plus **IchgID**, and if required, the **RoutingAddress**. You can also supply both the Partner Name and the unique identifier. The **RoutingAddress** field is required if more than one Partner has the same **IchgIdCdQual** and **IchgID**.

This tag reports a warning (or error if **Abort Run** is selected) if a Trading Partner does not exist.

Though all required fields are set to **No** (see **Req'd** field in the following table) you must specify the Partner Name and/or the unique identifier of the Trading Partner to delete.

The tag and field descriptions are described in the following table:

Field Label	FieldName/Tag	Req'd	Max. Len.	Validation
Partner Name	PartnerName	No	35	Unique
Partner ID Code Qualifier	IchgIdCdQual	No	4	fk to EDFIdCdQual
Interchange ID	IchgID	No	35	
Routing Address	RoutingAddress	No	14	

EDIFACT Application Partners (Tag: EDFAppPartners)

As with X12 Trading Partners, there can be zero or more Application Partners associated with an EDIFACT Trading Partner. Unlike X12, Application Partners are not required to create a Trade Link. Use this tag, when nested within a Trading Partner, to add Application Partners at the same time. If any Trading Partners are rejected, all associated Application Partners are rejected as well. EDIFACT Application Partners can have [Contacts](#) added . The tag and field descriptions are described in the following table:

Field Label	FieldName/Tag	Req'd	Max. Len.	Validation	Notes
N/A	IchgIdCdQual	Yes	4	IchgIdCdQual of Trading Partner. May be 4 spaces	1
N/A	IchgID	Yes	35	IchgID of Trading Partner	1

Field Label	FieldName/Tag	Req'd	Max. Len.	Validation	Notes
N/A	RoutingAddress	No	14	Routing Address of Trading Partner	1
Application Partner Name	ApplicationName	Yes	35	Unique within the Trading Partner	2
UNG Application ID	UNG_AppID	Yes	35	Unique within the Trading Partner	
Application Partner ID Code Qualifier	IDCdQual	No	4 Default: 4 spaces (None)	fk to EDFIdCdQual	
Password	Password	No	14	External only	
Alias	AliasName	No	20	Unique if present	3
User Defined 1	UserDefined1	No	20		
User Defined 2	UserDefined2	No	20		

Notes:

1. The IDs of the owner of this Application Partner; equal to the previously specified Trading Partner. Used to confirm that the XML is properly formed (Application Partners nested within the parent Trading Partner).
2. Do not use the name "UNKNOWN", or "ALL", or any variation of the case of these two words. For example, do not use "Unknown", or "All".
3. Alias Names are unique across all EDIFACT Application Partners, if present.

EDIFACT Application Partners Change (Tag: EDFAppPartnersAddChange)

Use this tag to add an EDIFACT Application Partner to an existing Trading Partner, or change an existing EDIFACT Application Partner. To add, set the tag **AddFlag** to **Y**; leaving **AddFlag** blank, not specified, or set to **N** will change the existing EDIFACT Application Partner if it exists. If **AddFlag** is **Y**, the **ApplicationName** and **UNG_AppID** fields are required.

To change the unique identifiers of an EDIFACT Application Partner, use the fields **ApplicationNameChange** and **UNG_AppIDChange**. Do not specify these fields when the AddFlag is **Y**. The resulting change from these fields cannot already exist in the database.

Application Partners can have Contacts added at the same time; nest the Contact record within this tag. Use this tag, along with the [Contacts](#) tag, to add Contacts to an existing Application Partner.

The tag and field descriptions are described in the following table:

Field Label	FieldName/Tag	Req'd	Max. Len.	Validation	Notes
N/A	IchgIdCdQual	No	4	IchgIdCdQual of Trading Partner. May be 4 spaces	1
N/A	IchgID	Yes	35	IchgID of Trading Partner	1
N/A	RoutingAddress	No	14	Routing Address of Trading Partner	1
Application Partner Name	ApplicationName	No	35	Unique within the Trading Partner	2
UNG Application ID	UNG_AppID	No	35	Unique within the Trading Partner	
Application Partner ID Code Qualifier	IDCdQual	No	4	fk to EDFIdCdQual	
Password	Password	No	14	External only	
Alias	AliasName	No	20	Unique if present	3
User Defined 1	UserDefined1	No	20		
User Defined 2	UserDefined2	No	20		
N/A	AddFlag	No	1	Y/N	
N/A	ApplicationNameChange	No	35		
N/A	UNG_AppIDChange	No	15		

Notes:

1. ID's of the owner of this Application Partner (Trading Partner's unique identifier).
2. Do not use the name "UNKNOWN", or "ALL", or any variation of the case of these two words. For example, do not use "Unknown", or "All".
3. Alias Names are unique across all EDIFACT Application Partners, if present.

EDIFACT Application Partners Delete (Tag: EDFAppPartnersDelete)

Use this tag to delete an existing EDIFACT Application Partner. Supply the Application Partner's Name in **ApplicationName**, and/or the Application Partners **UNG_AppID**. You must also supply the associated Trading Partner's name in **PartnerName** and/or the Trading Partner's unique identifier, which is the combination of **IchgIdCdQual** plus **IchgID**, and if required, the **RoutingAddress**. You can also supply both the **PartnerName** and the unique identifier. The **RoutingAddress** is required if more than one Trading Partner has the same **IchgIdCdQual** and **IchgID**. This tag will also delete any Contacts associated with the Application Partner. This tag will not delete if Application Partner is assigned to a trade link (see [EDFTradingPartnersDelete](#) if this is functionality needed). A warning report (or error if "Abort Run" is selected) is issued if Trade Link does not exist. The tag and field descriptions are described in the following table:

Field Label	FieldName/Tag	Req'd	Max. Len.	Validation
Application Partner Name	ApplicationName	No	35	
UNG Application ID	UNG_AppID	No	35	

Field Label	FieldName/Tag	Req'd	Max. Len.	Validation
N/A	IchgIdCdQual	Yes	4	IchgIdCdQual of Trading Partner. May be 4 spaces
N/A	IchgID	Yes	35	IchgID of Trading Partner
N/A	RoutingAddress	No	14	Routing Address of Trading Partner

EDIFACT inbound trade links

Trade links can be added, changed, or deleted. In addition, existing Trade Links can have their individual Documents added, changed, or deleted.

To add a new EDIFACT Inbound trade link, use the tags **EDFIBTradeLink**, **EDFIBAPLink**, and **EDFIInboundGroups**. To change an existing EDIFACT Inbound Trade Link, use the tags **EDFIBTradeLinkChange**, **EDFIBAPLinkChange**, and **EDFIInboundGroupsAddChangeDelete**. To delete an entire EDIFACT Inbound Trade Link, use **EDFIBTLDelete**.

There are two types of EDIFACT Inbound Trade Links: **Known** and **Unknown**. Unlike X12, there is no special Unknown Record type for EDIFACT Inbound Trade Links. Define unknown Trading Partners by setting the **FromIchgID** to **UNKNOWN** and leaving **FromIchgIdCdQual** and **FromRoutingAddress** blank (or not specified) in the XML data.

- Applications are specified in the **EDFIInboundGroups** record when adding a new Trade Link; otherwise use the **EDFIInboundGroupsAddChangeDelete** tag. An Application is the unique combination of **Message Version**, **Message Release**, **Controlling Agency**, the optional **Association Assign Code**, and a **Functional ID/Message Type**. Adding Applications to an existing Trade Link is supported in the "change" tags. See:
 - [EDFIBTradeLinkChange](#)
 - [EDFIBAPLinkChange](#)
 - [EDIFACT Inbound Applications Trade Links Add, Change, Delete \(Tag: EDFIInboundGroupsAddChangeDelete\)](#)

Trading/Application Partners must previously exist in the database for the Trade Link to be added. Unlike X12, Application Partners may or may not be specified on a Trade Link, depending on the setting of **MandatoryUNG**. Application Partners, if required, are specified within the **EDFIBAPLink** and **EDFIBAPLinkChange** tags, and must belong to the Trading Partner specified in the **EDFIBTradeLink/EDFIBTradeLinkChange** tags. Application Partners can only be specified if the **MandatoryUNG** field in **EDFIBTradeLink** is set to **O** (optional) or **M** (mandatory).

Applications (Documents) are nested within the Trade Link record when there is no **UNG** level; otherwise Applications are nested within the Application Partner record (**EDFIBAPLink/EDFIBAPLinkChange**). The same Application cannot appear more than once within a Trade Link.

- [Examples](#)
- [EDIFACT Inbound Trade Links \(Tag: EDFIBTradeLink\)](#)
- [EDIFACT Inbound Application Partner Trade Links \(Tag: EDFIBAPLink\)](#)
- [EDIFACT Inbound Applications Trade Links \(Tag: EDFIInboundGroups\)](#)
- [EDIFACT Inbound Trade Link Change \(Tag: EDFIBTradeLinkChange\)](#)
- [EDIFACT Inbound Application Partner Trade Links Change \(Tag: EDFIBAPLinkChange\)](#)
- [EDIFACT Inbound Applications Trade Links Add, Change, Delete \(Tag: EDFIInboundGroupsAddChangeDelete\)](#)
- [EDIFACT Inbound Trade Links Delete \(Tag: EDFIBTLDelete\)](#)

Examples

The following examples show how the XML data would look during an add operation:

Example 1 - Mandatory

Example 1: If **MandatoryUNG** is **M** (Mandatory), the XML would look like:

```
<EDFIBTradeLink>
  ...header data
  <EDFIBAPLink>
    ...specify application partners here
    <EDFIInboundGroups>
      ...application data
    </EDFIInboundGroups>
  </EDFIBAPLink>
</EDFIBTradeLink>
```

Example 2 - None

Example 2: If **MandatoryUNG** is **N** (None), the XML would look like:

```
<EDFIBTradeLink>
  ...header data
  <EDFIInboundGroups>
    ...application data
  </EDFIInboundGroups>
</EDFIBTradeLink>
```

Example 3 - Optional

Example 3: **MandatoryUNG** is **O** (Optional), and only one of the Applications is associated with an Application Partner (UNG). The XML would look like:

```

<EDFIBTradeLink>
...header data
<EDFIBAPLink>
...specify application partners here
<EDFIboundGroups>
...application data associated with the Application Partner
</EDFIboundGroups>
</EDFIBAPLink>
<EDFIboundGroups>
...application data not associated with any Application Partner
</EDFIboundGroups>
</EDFIBTradeLink>

```

The following table, field, and tag descriptions are for EDIFACT inbound trade links:

- "[EDIFACT Inbound Trade Links \(Tag: EDFIBTradeLink\)](#)"
- "[EDIFACT Inbound Application Partner Trade Links \(Tag: EDFIBAPLink\)](#)"
- "[EDIFACT Inbound Applications Trade Links \(Tag: EDFIboundGroups\)](#)"
- "[EDIFACT Inbound Trade Link Change \(Tag: EDFIBTradeLinkChange\)](#)"
- "[EDIFACT Inbound Application Partner Trade Links Change \(Tag: EDFIBAPLinkChange\)](#)"
- "[EDIFACT Inbound Applications Trade Links Add, Change, Delete \(Tag: EDFIboundGroupsAddChangeDelete\)](#)"
- "[EDIFACT Inbound Trade Links Delete \(Tag: EDFIBTLDelete\)](#)"

EDIFACT Inbound Trade Links (Tag: EDFIBTradeLink)

Use this tag to add a new EDIFACT Inbound Trade Link. It should be noted that there is one **EDFIBTradeLink** record per Trade Link.

The tag and field descriptions are described in the following table:

Field Label	FieldName/Tag	Req'd	Max. Len.	Validation	Notes
N/A	FromIchgIdCdQual	No	4	fk to EDFIdCdQual	1, 5
N/A	FromIchgID	Yes	35		1
N/A	FromRoutingAddress	No	14		1
N/A	ToIchgIdCdQual	No	4	fk to EDFIdCdQual	2, 5
N/A	ToIchgID	Yes	35		2
N/A	ToRoutingAddress	No	14		2
Get PostOffice	GetMailPO	Yes	3	Nickname of Get PO	3
UNG Option	MandatoryUNG	Yes	1	N = No UNG; O = Optional UNG; M = Mandatory UNG	4

Notes:

1. These three fields are confirmed to match an existing External Trading Partner. This partner can be in any EDIFACT folder or sub-folder. Set **FromIchgID** to **UNKNOWN** for the unknown partner, leaving **FromIchgIdCdQual** and **FromRoutingAddress** blank or un-specified.
2. These three fields must match an existing Internal Trading Partner. This partner can be in any EDIFACT folder or sub-folder
3. Must be exactly 3 characters long. Nickname must exist as a Get Post Office.
4. If **MandatoryUNG** is **M** (Mandatory) then Application Partners must be specified in **EDFIBAPLink** for every Application added (for example, **EDFIboundGroups** are nested within the **EDFIBAPLink** record). If **O** (Optional) then some or all Applications can be nested within an **EDFIBAPLink** record. If **N** (None) then no **EDFIBAPLink** records can be specified.
5. Defaults to 4 blanks.

EDIFACT Inbound Application Partner Trade Links (Tag: EDFIBAPLink)

Use this tag to associate Application Partners with a new EDIFACT Inbound Trade Link. This tag is required only if **MandatoryUNG** is **M** (Mandatory) in **EDFIBTradeLink**. May be included if **MandatoryUNG** is **O** (Optional). It cannot be used if **MandatoryUNG** is **N** (None). When used, one or more **EDFIboundGroups** records must be nested within this record. The tag and field descriptions are described in the following table:

Field Label	FieldName/Tag	Req'd	Max. Len.	Validation	Notes
N/A	FromUNG_AppID	Yes	35		1
N/A	ToUNG_AppID	Yes	35		2

Notes:

1. Equal to an existing EDIFACT Application Partner's **UNG_AppID** that is owned by the External Trading Partner as defined in the three **From** fields in **EDFIBTradeLink**.
2. Equal to an existing EDIFACT Application Partner's **UNG_AppID** that is owned by the Internal Trading Partner as defined in the three **To** fields in **EDFIBTradeLink**.

EDIFACT Inbound Applications Trade Links (Tag: EDFInboundGroups)

Use this tag to add Documents to a new EDIFACT Inbound Trade Link. This record can appear within a Trade Link or within an Application Partner record. This record type is optional when not within an Application Partner record, however, if not included you will have an incomplete Trade Link (such links are valid in Partner Manager in the event you wish to complete it at a later time). The tag and field descriptions are described in the following table:

Field Label	FieldName/Tag	Req'd	Max. Len.	Validation	Notes
Message Version Number	MessageVersionNo	Yes	3	fk to EDFMsgVerNo	1
Message Release Number	MessageRelNo	Yes	3	fk to EDFMsgRelNo	1
Controlling Agency	ControllingAgency	Yes	2	fk to EDFControlAgency	1
Association Assign Code	AssocAssignCd	No	6	fk to EDFAssocAssignCd	1
Functional ID/Message Type	Func_MsgType	Yes	6	fk to EDF_FuncID along with the above fields	1
Test Indicator	TestIndicator	No	1	T=True, F=False Default: F	2
Production Indicator	ProdIndicator	No	1	T=True, F=False Default: T	2
Validate By	ValidateByInd	No	1	"M"= Message; "F"=Functional Group/Interchange Default M.	3
Test PO Nickname	TestPutPO	Yes if TestIndicator is T	3	Nickname of Valid Put PO	4
Prod PO Nickname	ProdPutPO	Yes if ProdIndicator is T	3	Nickname of Valid Put PO	5
AckLevel	AckLevel	No	1	See Note	6
Routing: Route Only	RouteOnly	No	1	Y/N Default: N	

Notes:

1. The combinations of these five fields define the Application and must be unique within one Trade Link. For example, do not duplicate the Application in a **EDFIBAPLink** as well as within an **EDFIBTradeLink**, even if the **MandatoryUNG** is **O** (Optional).
2. At least one of (**TestIndicator** or **ProdIndicator**) must be **True**.
3. Note the field labels within Partner Manager change for this field depending on **MandatoryUNG** in **EDFIBTradeLink**. When **MandatoryUNG** is set to **N** (None), **Validate By** can be **Message** or **Interchange**. When **MandatoryUNG** is set to **M** (Mandatory) or **O** (Optional), **Validate By** can be **Message** or **Functional Group**. For the purposes of this XML input, set to **M** or **F** where **F** can mean either **Functional Group** or **Interchange** and **M** denotes **Message**.
4. Must be exactly 3 characters if present. Ignored if **TestIndicator** is **False**, required otherwise. Nickname must point to a valid Put Post Office.
5. Must be exactly 3 characters if present. Ignored if **ProdIndicator** is **False**, required otherwise. Nickname must point to a valid Put Post Office.
6. Specify the optional Acknowledgement Level. If not specified, this value will be zero, No Acknowledgement. Valid values are **0**, **1**, **2**, **3**, **4** or **5**. These values are: **0** = No Acknowledgement, **1** = Interchange Level, **2** = Functional Group Level, **3** = Message Level, **4** = Segment Level, **5** = Data Element Level. Value **2** (Functional Group Level) can only be specified if Mandatory **UNG** is in use (Application Partners in tag **EDFIBAPLink** are defined for this link).

EDIFACT Inbound Trade Link Change (Tag: EDFIBTradeLinkChange)

Use this tag to change an existing EDIFACT Inbound Trade Link. This tag is used to determine the existing Trade Link at the Trading Partner level, so you must specify the From and To Trading Partner ID in the first 6 fields. Notice that the Routing Address fields are only required if more than one Trading Partner exists for the Interchange ID/Qualifiers. The remaining field, **GetMailPO**, is optional when changing a Trade Link. The child tags (**EDFIBAPLinkChange** and **EDFInboundGroupsAddChangeDelete**) need not be included if only changing data at this level (you only want to change the Get Post Office).

Changing the **UNG** option (Mandatory/None/Optional) is not supported. If you need to change the **UNG** option, use Partner Manager, or remove the existing trade link with **EDFIBTLDelete**, then add it again using **EDFIBTradeLink**.

The tag and field descriptions are described in the following table:

Field Label	FieldName/Tag	Req'd	Max. Len.	Validation	Notes
N/A	FromIchgIdCdQual	No	4	fk to EDFIdCdQual	1, 4
N/A	FromIchgID	Yes	35		1
N/A	FromRoutingAddress	No	14		1
N/A	ToIchgIdCdQual	No	4	fk to EDFIdCdQual	1, 4
N/A	ToIchgID	Yes	35		2
N/A	ToRoutingAddress	No	14		2
Get PostOffice	GetMailPO	No	3	Nickname of Get PO	3

Notes:

1. These three fields are confirmed to match an existing External Trading Partner. This partner can be in any EDIFACT folder or sub-folder. Set **FromIchgID** to **UNKNOWN** for the unknown partner, leaving **FromIchgIdCdQual** and **FromRoutingAddress** blank or un-specified.
2. These three fields must match an existing Internal Trading Partner. This partner can be in any EDIFACT folder or sub-folder.
3. Must be 3 characters long (if specified). Nickname must exist as a Get Post Office.
4. Defaults to 4 blanks.

EDIFACT Inbound Application Partner Trade Links Change (Tag: EDFIBAPLinkChange)

Use this tag to identify the Application Partners associated with an EDIFACT Inbound Trade Link you wish to change. This tag is required only if **MandatoryUNG** is **M** (Mandatory) in **EDFIBTradeLink**. May be included if **MandatoryUNG** is **O** (Optional). It cannot be used if **MandatoryUNG** is **N** (None). This tag's sole purpose is to identify the Application Partners associated with groups at the next level (**EDFIInboundGroupsAddChangeDelete**). No changes to the database can be made within this tag itself. The tag and field descriptions are described in the following table:

Field Label	FieldName/Tag	Req'd	Max. Len.	Validation	Notes
N/A	FromUNG_AppID	Yes	35		1
N/A	ToUNG_AppID	Yes	35		2

Notes:

1. Equal to an existing EDIFACT Application Partner's **UNG_AppID** that is owned by the External Trading Partner as defined in the three **From** fields in **EDFIBTradeLinkChange**.
2. Equal to an existing EDIFACT Application Partner's **UNG_AppID** that is owned by the Internal Trading Partner as defined in the three **To** fields in **EDFIBTradeLinkChange**.

EDIFACT Inbound Applications Trade Links Add, Change, Delete (Tag: EDFIInboundGroupsAddChangeDelete)

Use this tag to add Documents to an existing Inbound EDIFACT Trade Link, or to change existing Documents. This tag must be nested within **EDFIBTradeLinkChange** tag, or if required, the **EDFIBAPLinkChange** tag. To add a Document, set the **AddFlag** to **Y**. To delete, set the **DeleteFlag** to **Y**. Note when adding or changing a Document so that it requires an acknowledgement, the outbound acknowledgement trade link (CONTRL) record must already exist. The tag and field descriptions are described in the following table:

Field Label	FieldName/Tag	Req'd	Max. Len.	Validation	Notes
Message Version Number	MessageVersionNo	Yes	3	fk to EDFMsgVerNo	1
Message Release Number	MessageRelNo	Yes	3	fk to EDFMsgRelNo	1
Controlling Agency	ControllingAgency	Yes	2	fk to EDFControlAgency	1
Association Assign Code	AssocAssignCd	No	6	fk to EDFAssocAssignCd	1
Functional ID/Message Type	Func_MsgType	Yes	6	fk to EDF_FuncID along with the above fields	1
Test Indicator	TestIndicator	No	1	T=True, F=False, R=Remove	2
Production Indicator	ProdIndicator	No	1	T=True, F=False, R=Remove	2
Validate By	ValidateByInd	No	1	"M"= Message; "F"=Functional Group/Interchange	3
Test PO Nickname	TestPutPO	Yes if TestIndicator is T	3	Nickname of Valid Put PO	4
nProd PO Nickname	ProdPutPO	Yes if ProdIndicator is T	3	Nickname of Valid Put PO	5
AckLevel	AckLevel	No	1	See Note	6
N/A	AddFlag	No	1		
N/A	DeleteFlag	No	1		7
Routing: Route Only	RouteOnly	No	1	Y/N Default: N	

Notes:

1. The combinations of these five fields define the Application and must be unique within one Trade Link. For example, do not duplicate the Application in a EDFIBAPLink as well as within an EDFIBTradeLink, even if the MandatoryUNG is O (Optional).
2. At least one of (**TestIndicator** or **ProdIndicator**) must be **True** after completion of any database change. If you want to remove an existing Test or Production Post Office, set this field to **R**. For example, this is available in the event you want to change an existing group from Test only to Production Only: set the **TestIndicator** to **R**, which will remove the Test Post Office, and set the ProductionIndicator to **T** and specify the new Production Post Office in **ProdPutPO**. When setting this field to **R**, you must specify the other (test/production) Post Office if one does not already exist on the group.
3. Note the field labels within Partner Manager change for this field depending on **MandatoryUNG** in **EDFIBTradeLink**. When **MandatoryUNG** is set to **N** (None), **Validate By** can be **Message** or **Interchange**. When **MandatoryUNG** is set to **M** (Mandatory) or **O** (Optional), **Validate By** can be **Message** or **Functional Group**. For the purposes of this XML input, set to **M** or **F** where **F** can mean either **Functional Group** or **Interchange** and **M** denotes **Message**.
4. Must be exactly 3 characters if present. Ignored if **TestIndicator** is **False**, required otherwise. Nickname must points to a valid Put Post Office.
5. Must be exactly 3 characters if present. Ignored if **ProdIndicator** is **False**, required otherwise. Nickname must points to a valid Put Post Office.
6. Specify the optional Acknowledgement Level. If not specified, this value will be zero, No Acknowledgement. Valid values are **0**, **1**, **2**, **3**, **4**, or **5**. These values are: **0** = No Acknowledgement, **1** = Interchange Level, **2** = Functional Group Level, **3** = Message Level, **4** = Segment Level, **5** = Data Element Level. Value **2** (Functional Group Level) can only be specified if Mandatory **UNG** is in use (Application Partners in tag **EDFIBAPLink** are defined for this link).
7. Set to **Y** if you wish to remove an Inbound Group from a Trade Link. Specify only the first four fields and this flag when deleting a group; all other fields will be ignored. If you want to completely delete a Group with Optional **UNG**, you must delete both records that exist in this case: one with the Application Partners (use the tag **EDFIBAPLinkChange**), and the one without Application Partners (do not use the tag **EDFIBAPLinkChange**).

EDIFACT Inbound Trade Links Delete (Tag: EDFIBTLDelete)

Use this tag to delete an existing EDIFACT Inbound Trade Link. It requires the unique identifier of the EDIFACT 'From' (External) Trading Partner: **FromIchgIdCdQual** plus **FromIchgID**, and if required, the **FromRoutingAddress**. This tag also requires the unique identifier of the EDIFACT 'To' (Internal) Trading Partner: **ToIchgIdCdQual** plus **ToIchgID**, and if required, the **ToRoutingAddress**. The Routing Addresses are required if more than one Partner has the same **IchgIdCdQual** and **IchgID**. A warning report (or error if "Abort Run" is selected) is issued if Trade Link does not exist.

Warning: Use with caution: this will delete *all* Trade Links defined for this Inbound Trading Partner relationship, including all Application Partner-level links that may exist.

The tag and field descriptions are described in the following table:

Field Label	FieldName/Tag	Req'd	Max. Len.	Validation	Notes
N/A	FromIchgIdCdQual	No	4	fk to EDFIdCdQual	1, 3
N/A	FromIchgID	Yes	35		1
N/A	FromRoutingAddress	No	14		1
N/A	ToIchgID	Yes	35		2
N/A	ToRoutingAddress	No	14		2

Notes:

- These three fields are confirmed to match an existing External Trading Partner. This partner can be in any EDIFACT folder or sub-folder. Set **FromIchgID** to **UNKNOWN** for the unknown partner, leaving **FromIchgIdCdQual** and **FromRoutingAddress** blank or un-specified.
- These three fields must match an existing Internal Trading Partner.
- Defaults to 4 blanks.

EDIFACT outbound trade links

Trade links can be added, changed, or deleted. In addition, existing Trade Links can have their individual Messages (also known as Documents or Groups) added, changed, or deleted.

To add a new EDIFACT Outbound trade link, use the tags **EDFOBTPLink**, **EDFOBAPLink**, and **EDFOBMsgLink**. To change an existing EDIFACT Outbound Trade Link, use the tags **EDFOBTPLinkChange**, **EDFOBAPLinkChange**, and **EDFOBMsgLinkAddChangeDelete**. To delete an entire EDIFACT Outbound Trade Link, use **EDFOBTLDelete**.

EDIFACT Outbound Trade Links are very similar to Inbound Trade Links. The main difference is no Unknown Outbound Trade Links. All Post Offices are defined at the Message level. Applications are specified in the tag **EDFOBMsgLink** when adding, or **EDFOBMsgLinkAddChangeDelete** tag when changing.

Trading/Application Partners must previously exist in the database for the Trade Link to be added. As with Inbound Trade Links, Application Partners may or may not be specified, depending on the setting of **MandatoryUNG**. Application Partners, if required, are specified within the **EDFOBAPLink** or **EDFOBAPLinkChange** tags, and must "belong" to the Trading Partner specified in the **EDFOBTPLink/EDFOBTPLinkChange** tag. Application Partners can only be specified if the **MandatoryUNG** field is set to **O** (optional) or **M** (mandatory).

Applications are nested within the Trade Link record when there is no UNG level; otherwise Applications are nested within the Application Partner record. The same Application cannot appear more than once within a Trade Link. The following table, field, and tag descriptions are for EDIFACT outbound trade links:

- ["EDIFACT Outbound Trade Links \(Tag: EDFOBTPLink\)"](#)
- ["EDIFACT Outbound Application Partner Trade Links \(Tag: EDFOBAPLink\)"](#)
- ["EDIFACT Outbound Applications Trade Links \(Tag: EDFOBMsgLink\)"](#)
- ["EDIFACT Outbound Trade Links Change \(Tag: EDFOBTPLinkChange\)"](#)
- ["EDIFACT Outbound Applications Trade Links Add, Change, Delete \(Tag: EDFOBMsgLinkAddChangeDelete\)"](#)
- ["EDIFACT Outbound Trade Link Delete \(Tag: EDFOBTLDelete\)"](#)
- ["EDIFACT Outbound Trade Links \(Tag: EDFOBTLDelete\)"](#)
- ["EDIFACT Outbound Application Partner Trade Links \(Tag: EDFOBAPLink\)"](#)
- ["EDIFACT Outbound Applications Trade Links \(Tag: EDFOBMsgLink\)"](#)
- ["EDIFACT Outbound Trade Links Change \(Tag: EDFOBTPLinkChange\)"](#)
- ["EDIFACT Outbound Application Partner Trade Links Change \(Tag: EDFOBAPLinkChange\)"](#)
- ["EDIFACT Outbound Applications Trade Links Add, Change, Delete \(Tag: EDFOBMsgLinkAddChangeDelete\)"](#)
- ["EDIFACT Outbound Trade Link Delete \(Tag: EDFOBTLDelete\)"](#)

EDIFACT Outbound Trade Links (Tag: EDFOBTPLink)

Use this tag to add a new EDIFACT Outbound Trade Link.

There is One **EDFOBTPLink** record per Trade Link.

The tag and field descriptions are described in the following table:

Field Label	FieldName/Tag	Req'd	Max. Len.	Validation	Notes
N/A	FromIchgIdCdQual	No	4	fk to EDFIdCdQual	1, 7
N/A	FromIchgID	Yes	35		1
N/A	FromRoutingAddress	No	14		1
N/A	ToIchgIdCdQual	No	4	fk to EDFIdCdQual	2, 7
N/A	ToIchgID	Yes	35		2
N/A	ToRoutingAddress	No	14		2
UNG Option	MandatoryUNG	Yes	1	N = No UNG; O = Optional UNG; M = Mandatory UNG	3
Control Scheme	UNBControlNoScheme	No	1		4

Field Label	FieldName/Tag	Req'd	Max. Len.	Validation	Notes
Control Map	UNBControlNoMap	Yes, if scheme is 2	255		5
UNB Control Number	UNBControlNoValue	No	14		6

Notes:

1. These three fields must match an existing Internal Trading Partner. This partner can be in any EDIFACT folder or sub-folder.
2. These three fields must match an existing External Trading Partner. This partner can be in any EDIFACT folder or sub-folder.
3. If **MandatoryUNG** is **M** (Mandatory) then Application Partners must be specified in **EDFOBAPLink** for every Application added (for example, **EDFOBMsgLink** records are nested within the **EDFOBAPLink** record). If **O** (Optional), than some or all Applications can be nested within an **EDFOBAPLink** record. If **N** (None), then no **EDFOBAPLink** records can be specified.
4. Valid values for **UNBControlNoScheme** are: **0**=Trade Link; **1**=Message; **2**=User Defined; **3**=None. The default value is **0** (Trade Link). If set to **2** (User Defined), a **UNBControlNoMap** is required. If set to 1 (Message), control numbers may be set at the Message level (see [EDFOBMsgLink](#)).
5. A Control Number Map is required and can only be specified if **UNBControlNoScheme** is set to 2.
6. A Control Number can only be specified if **UNBControlNoScheme** is either set to or defaults to **0** (Trade Link), or is set to **2** (User Defined). If specified, this value must be a positive integer no longer than 14 characters.
7. Defaults to 4 blanks.

EDIFACT Outbound Application Partner Trade Links (Tag: **EDFOBAPLink**)

Use this tag to add the associated Application Partners to a new EDIFACT Outbound Trade Link. If changing an existing EDIFACT Outbound Link with Application Partners, see [EDFOBAPLinkChange](#).

This tag is required only if **MandatoryUNG** is **M** (Mandatory) in **EDFOBTPLink**, above. It may be included if **MandatoryUNG** is **O** (Optional). This tag cannot be used if **MandatoryUNG** is **N** (None). When used, one or more **EDFOBMsgLink** records must be nested within this record.

The tag and field descriptions are described in the following table:

Field Label	FieldName/Tag	Req'd	Max. Len.	Notes
N/A	FromUNG_AppID	Yes	35	1
N/A	ToUNG_AppID	Yes	35	1

Notes:

1. Equal to an existing EDIFACT Application Partner's UNG_AppID that is owned by the Internal Trading Partner as defined in the three From fields in **EDFOBTPLink**.
2. Equal to an existing EDIFACT Application Partner's **UNG_AppID** that is owned by the External Trading Partner as defined in the three **To** fields in **EDFOBTPLink**.

EDIFACT Outbound Applications Trade Links (Tag: **EDFOBMsgLink**)

Use this tag to add Messages (also known as Documents) to a new EDIFACT Outbound Trade Link. This record can appear within a Trade Link, or within an Application Partner record. This record type is optional when not within an Application Partner record, but in this case, you do not have a complete Trade Link (such links are valid in Partner Manager in the event you wish to complete it at a later time). The tag and field descriptions are described in the following table:

Field Label	FieldName/Tag	Req'd	Max. Len.	Validation	Notes
Message Version Number	MessageVersionNo	Yes	3	fk to EDFMsgVerNo	1
Message Release Number	MessageRelNo	Yes	3	fk to EDFMsgRelNo	1
Controlling Agency	ControllingAgency	Yes	2	fk to EDFControlAgency	1
Association Assign Code	AssocAssignCd	No	6	fk to EDFAssocAssignCd	1
Functional ID/Message Type	MessageType	Yes	6	fk to EDF_FuncID along with the above fields	1
Get Post Office	GetMailPO	Yes	3	Nickname of valid Get PO	
Test Indicator	TestIndicator	No	1	T=True, F=False Default: F	2
Production Indicator	ProdIndicator	No	1	T=True, F=False Default: T	2
Test PO Nickname	TestPutPO	Yes if TestIndicator is T	3	Nickname of valid Put PO	3
Prod PO Nickname	ProdPutPO	Yes if ProdIndicator is T	3	Nickname of Valid Put PO	4
Control No	LastMsgRefNo	No	14		5
Routing: Route Only	RouteOnly	No	1	Y/N Default: N	

Notes:

1. The combination of these five fields define the Application and must be unique within one Trade Link. For example, do not duplicate the Application in a **EDFOBAPLink** as well as within a **EDFOBTPLink**, even if the **MandatoryUNG** is **O** (Optional).
2. At least one of (**TestIndicator** or **ProdIndicator**) must be **True**.
3. Must be exactly 3 characters if present. Ignored if **TestIndicator** is **False**, required otherwise. Nickname must point to a valid Put Post Office.
4. Must be exactly 3 characters if present. Ignored if **ProdIndicator** is **False**, required otherwise. Nickname must point to a valid Put Post Office.
5. If the Control Numbering Scheme of this Trade Link is Message (see the [EDFOBTPLink](#) field **UNBControlNoScheme**) then you may over-ride the control number by placing a value here. If specified, this value must be a positive integer no longer than 14 characters in length.

EDIFACT Outbound Trade Links Change (Tag: EDFOBTPLinkChange)

Use this tag to change an existing EDIFACT Outbound Trade Link. This tag is used to determine the existing Trade Link at the Trading Partner level, so you must specify the From and To Trading Partner ID in the first 6 fields. Note the Routing Address fields are only required if more than one Trading Partner exists for the Interchange ID/Qualifiers. The remaining fields are optional when changing a Trade Link.

The child tags (**EDFOBAPLinkChange** and **EDFOBMsgLinkAddChangeDelete**) need not be included if only changing data at this level.

Changing the **UNG** option (Mandatory/None/Optional) is not supported. If you need to change the **UNG** option, use Partner Manager, or remove the existing trade link with **EDFOBTLDelete**, then add it again using **EDFOBTradeLink** tag.

The tag and field descriptions are described in the following table:

Field Label	FieldName/Tag	Req'd	Max. Len.	Validation	Notes
N/A	FromIchgIdCdQual	No	4	fk to EDFIdCdQual	1, 6
N/A	FromIchgID	Yes	35		1
N/A	FromRoutingAddress	No	14		1
N/A	ToIchgIdCdQual	No	4	fk to EDFIdCdQual	2, 6
N/A	ToIchgID	Yes	35		2
N/A	ToRoutingAddress	No	14		2
Control Scheme	UNBControlNoScheme	No	1		3
Control Map	UNBControlNoMap	Yes, if scheme is 2	255		4
UNB Control Number	UNBControlNoValue	No	14		5

Notes:

1. These three fields must match an existing Internal Trading Partner. This partner can be in any EDIFACT folder or sub-folder.
2. These three fields must match an existing External Trading Partner. This partner can be in any EDIFACT folder or sub-folder.
3. Valid values for **UNBControlNoScheme** are: **0**=Trade Link; **1**=Message; **2**=User Defined; **3**=None. The default value is **0** (Trade Link). If set to **2** (User Defined), a **UNBControlNoMap** is required. If set to **1** (Message), control numbers may be set at the Message level (see [EDFOBMsgLink](#)).
4. A Control Number Map is required and can only be specified if **UNBControlNoScheme** is set to **2**.
5. Control Number can only be specified if **UNBControlNoScheme** is either set to or exists as **0** (Trade Link), or **2** (User Defined).
6. Defaults to 4 blanks.

EDIFACT Outbound Application Partner Trade Links Change (Tag: EDFOBAPLinkChange)

This tag is required if the trade link uses Application Partners; i.e., the Trade Link contains a **MandatoryUNG** setting of "Optional" or "Mandatory". If the **MandatoryUNG** flag is None, do not use this tag. This tag's sole purpose is to identify the Application Partners associated with messages at the next level; no changes to the database can be made within this tag itself. The tag and field descriptions are described in the following table:

Field Label	FieldName/Tag	Req'd	Max. Len.	Validation	Notes
N/A	FromUNG_AppID	Yes	35		1
N/A	ToUNG_AppID	Yes	35		2

Notes:

1. Equal to an existing EDIFACT Application Partner's **UNG_AppID** that is owned by the Internal Trading Partner as defined in the three From fields in **EDFOBTPLinkChange**.
2. Equal to an existing EDIFACT Application Partner's **UNG_AppID** that is owned by the External Trading Partner as defined in the three **To** fields in **EDFOBTPLinkChange**.

EDIFACT Outbound Applications Trade Links Add, Change, Delete (Tag: EDFOBMsgLinkAddChangeDelete)

This tag is used to add, change, or delete an Application (Message or Group) in an existing EDIFACT Outbound Trade Link. Must be nested within **EDFOBTradeLinkChange** tag, or if required, the **EDFOBAPLinkChange** tag. To add an Application, set the **AddFlag** to **Y**. To delete, set the **DeleteFlag** to **Y**.

This tag is similar to the **EDFOBMsgLink** tag used when adding a new Trade Link, with the following significant differences:

- You can remove a Test or Production Post Office by setting the Test/Production Indicators to **R**. See Note 2 for more information on this feature.
- To add a new Message to an existing link, set **AddFlag** to **Y**.
- To remove an existing Message, set the **DeleteFlag** to **Y**.

The tag and field descriptions are described in the following table:

Field Label	FieldName/Tag	Req'd	Max. Len.	Validation	Notes
Message Version Number	MessageVersionNo	Yes	3	fk to EDFMsgVerNo	1

Field Label	FieldName/Tag	Req'd	Max. Len.	Validation	Notes
Message Release Number	MessageRelNo	Yes	3	fk to EDFMsgRelNo	1
Controlling Agency	ControllingAgency	Yes	2	fk to EDFControlAgency	1
Association Assign Code	AssocAssignCd	No	6	fk to EDFAssocAssignCd	1
Functional ID/Message Type	MessageType	Yes	6	fk to EDF_FuncID along with the above fields	1
Get Post Office	GetMailPO	No	3	Nickname of valid Get PO	
Test Indicator	TestIndicator	No	1	T=True, F=False, R=Remove	2
Production Indicator	ProdIndicator	No	1	T=True, F=False, R=Remove	2
Test PO Nickname	TestPutPO	Yes if TestIndicator is T	3	Nickname of valid Put PO	3
Prod PO Nickname	ProdPutPO	Yes if ProdIndicator is T	3	Nickname of Valid Put PO	4
Control No	LastMsgRefNo	No	14		5
N/A	AddFlag	No	1	Y/N	
N/A	DeleteFlag	No	1	Y/N	6
Routing: Route Only	RouteOnly	No	1	Y/N Default: N	

Notes:

- The combination of these five fields define the Application and must be unique within one Trade Link. For example, do not duplicate the Application within a EDOBAPLinkChange as well as within a EDOBTPLinkChange, even if the MandatoryUNG is O (Optional).
- After all changes are applied, at least one of (**TestIndicator** or **ProdIndicator**) must be **True**. If you want to remove an existing Test or Production Post Office, set this field to "R". For example, this is available in the event you want to change an existing message from Test only to Production Only: set the TestIndicator to "R", which will remove the Test Post Office, and set the ProductionIndicator to **T** and specify the new Production Post Office in ProdPutPO. When setting this field to **R**, you must specify the other (test/production) Post Office if one does not already exist on the message
- Must be exactly 3 characters if present. Ignored if **TestIndicator** is **False**, required otherwise. Nickname must point to a valid Put Post Office.
- Must be exactly 3 characters if present. Ignored if **ProdIndicator** is **False**, required otherwise. Nickname must point to a valid Put Post Office.
- If the Control Numbering Scheme of this Trade Link is Message (see [EDOBTPLinkChange](#) field **UNBControlNoScheme**) then you may over-ride the control number by placing a value here. If specified, this value must be a positive integer no longer than 14 characters in length.
- Set the DeleteFlag to **Y** if you wish to remove an existing message from a Trade Link. Specify only the first five fields and this flag when deleting a group; all other fields will be ignored. If you want to complete delete a Group with Optional UNG, you must delete both records that exist in this case: one with the Application Partners (use the tag [EDOBAPLinkChange](#)), and the one without Application Partners (do not use the tag [EDOBAPLinkChange](#)).

EDIFACT Outbound Trade Link Delete (Tag: EDOBTDelete)

Use this tag to delete an existing EDIFACT Outbound Trade Link. Requires the unique identifier of the EDIFACT 'From' (Internal) Trading Partner: **FromIchgIdCdQual** plus **FromIchgID**, and if required, the **FromRoutingAddress**. Also requires the unique identifier of the EDIFACT 'To' (External) Trading Partner: **ToIchgIdCdQual** plus **ToIchgID**, and if required, the **ToRoutingAddress**. The Routing Addresses are required if more than one Partner has the same **IchgIdCdQual** and **IchgID**. A warning report (or error if "Abort Run" is selected) is issued if Trade Link does not exist.

Warning Use with caution: this will delete all Trade Links defined for this Outbound Trading Partner relationship, including all Application Partner-level links that may exist.

The tag and field descriptions are described in the following table:

Field Label	FieldName/Tag	Req'd	Max. Len.	Validation	Notes
N/A	FromIchgIdCdQual	No	4	fk to EDFIdCdQual	1, 3
N/A	FromIchgID	Yes	35		1
N/A	FromRoutingAddress	No	14		1
N/A	ToIchgIdCdQual	No	4	fk to EDFIdCdQual	2, 3
N/A	ToIchgID	Yes	35		2
N/A	ToRoutingAddress	No	14		2

Notes:

- These three fields must match an existing Internal Trading Partner. This partner can be in any EDIFACT folder or sub-folder.
- These three fields must match an existing External Trading Partner. This partner can be in any EDIFACT folder or sub-folder.
- Defaults to 4 blanks.

TRADACOMS Trading/Application Partners

The following table, field, and tag descriptions are for TRADACOMS Partners:

- ["TRADACOMS Trading Partners: \(Tag TRCTradingPartners\)"](#)
- ["TRADACOMS Trading Partners Change \(Tag: TRCTradingPartnersChange\)"](#)
- ["TRADACOMS Trading Partners Delete \(Tag: TRCTradingPartnersDelete\)"](#)
- ["TRADACOMS Application Partners \(Tag: TRCAppPartners\)"](#)
- ["TRADACOMS Application Partners Change \(Tag: TRCAppPartnersChange\)"](#)
- ["TRADACOMS Application Partners Delete \(Tag: TRCAppPartnersDelete\)"](#)

For information about inbound TRADACOMS trade links, see ["TRADACOMS Inbound Trade Links"](#).

- ["TRADACOMS Trading Partners: \(Tag TRCTradingPartners\)"](#)

- [TRADACOMS Trading Partners Change \(Tag: TRCTradingPartnersChange\)](#)
- [TRADACOMS Trading Partners Delete \(Tag: TRCTradingPartnersDelete\)](#)
- [TRADACOMS Application Partners \(Tag: TRCAppPartners\)](#)
- [TRADACOMS Application Partners Change \(Tag: TRCAppPartnersChange\)](#)
- [TRADACOMS Application Partners Delete \(Tag: TRCAppPartnersDelete\)](#)

TRADACOMS Trading Partners: (Tag TRCTradingPartners)

Use this tag to add new TRADACOMS Trading Partners. The tag and field descriptions are:

Field Label	FieldName/Tag	Req'd	Max. Len.	Validation	Notes
N/A	IntExt	Yes	1	I or E I=Internal, E=External	2
Trading Partner Name	PartnerName	Yes	35		
Partner ID	PartnerID	Yes	35		2
Duplicate Control	Duplicate_Control	No	1		3

Notes

1. Cannot be "all" or "unknown".
2. Typically, an EAN number is used in TRADACOMS to identify partners. However, if an EAN is not available and the partners agree, this field can be a name.
3. **N**=none; **I**=Interchange. Can only be set on External Partners.

TRADACOMS Trading Partners Change (Tag: TRCTradingPartnersChange)

Use this tag to change existing TRADACOMS Trading Partners. Fields are similar to TRCTradingPartners, except you there is no "IntExt" field: you cannot change the Internal/External designation of a Trading Partner. Identify the Trading Partner to change using **PartnerName**, **PartnerID**, or both. If you wish to change the name or ID, use the fields **PartnerNameChange** and **PartnerIDChange**. Contacts can be added via this tag; include the <Contact> tag nested within this tag to add. This tag reports a warning (or error if "Abort Run" is selected) if Trading Partner does not exist. The tag and field descriptions are:

Field Label	FieldName/Tag	Req'd	Max. Len.	Validation	Notes
Trading Partner Name	PartnerName	No	35		
Partner ID	PartnerID	No	35		1
Duplicate Control	Duplicate_Control	No	1		2
N/A	PartnerNameChange	No	35		
N/A	PartnerIDChange	No	35		

Notes:

1. Typically, an EAN number is used in TRADACOMS to identify partners. However, if an EAN is not available and the partners agree, this field can be a name.
2. **N**=none; **I**=Interchange. Can only be set on External Partners.

TRADACOMS Trading Partners Delete (Tag: TRCTradingPartnersDelete)

Use this tag to delete a TRADACOMS Trading Partner and all associated data with that Partner. Supply **PartnerName**, **PartnerID**, or both. Use with caution: Deleting a Trading Partner will delete all Application Partners, Contacts, and Trade Links associated with that trading partner.

This tag reports a warning (or error if "Abort Run" is selected) if Trading Partner does not exist. The tag and field descriptions are described in the following table:

Field Label	FieldName/Tag	Req'd	Max. Len.	Validation	Notes
Trading Partner Name	PartnerName	No	35		
Partner ID	PartnerID	No	35		

TRADACOMS Application Partners (Tag: TRCAppPartners)

TRADACOMS Application Partners (Tag: TRCAppPartners)

Use this tag, when nested within a Trading Partner, to add Application Partners at the same time. Application Partners are optional in TRADACOMS. The tag and field descriptions are:

Field Label	FieldName/Tag	Req'd	Max. Len.	Validation	Notes
N/A	PartnerID	Yes	35		1
Application Partner Name	ApplicationName	Yes	35		2
Application Partner ID	AppPartnerID	Yes	35		3
Alias	AliasName	No	20		4
User Defined 1	UserDefined1	No	20		

Field Label	FieldName/Tag	Req'd	Max. Len.	Validation	Notes
User Defined 2	UserDefined2	No	20		

Notes:

1. ID's of the owner of this Application Partner; equal to the previously specified Trading Partner. Used to confirm that the XML is properly formed (Application Partners nested within the parent Trading Partner).
2. Do not use the name "UNKNOWN, or "ALL", or any variation of the case of these two words. For example, do not use "Unknown", or "All".
3. Typically, an EAN number is used in TRADACOMS to identify partners. However, if an EAN is not available and the partners agree, this field can be a name.
4. Alias Names are unique across all TRADACOMS Application Partners, if present.

TRADACOMS Application Partners Change (Tag: TRCApPartnersChange)

Use this tag to add an TRADACOMS Application Partner to an existing Trading Partner, or change an existing Application Partner. To add, set the tag **AddFlag** to **Y**; leaving **AddFlag** blank, not specified, or set to **N** will change the existing TRADACOMS Application Partner if it exists.

To change the unique identifiers of an TRADACOMS Application Partner, use the fields **ApplicationNameChange** and **AppPartnerIDChange**. Do not specify these fields when the AddFlag is **Y**. The resulting change from these fields cannot already exist in the database. Application Partners can have Contacts [Contacts \(Tag: Contacts\)](#) added at the same time; nest the Contact record within this tag. Use this tag, along with the [Contacts](#) tag, to add Contacts to an existing Application Partner.

The tag and field descriptions are:

Field Label	FieldName/Tag	Req'd	Max. Len.	Validation	Notes
N/A	PartnerID	Yes	35		1
Application Partner Name	ApplicationName	No	35		2
Application Partner ID	AppPartnerID	No	35		3
Alias	AliasName	No	20		4
User Defined 1	UserDefined1	No	20		
User Defined 2	UserDefined2	No	20		
N/A	AddFlag	No	1	Y/N	
N/A	ApplicationNameChange	No	35		
N/A	AppPartnerIDChange	No	35		

Notes:

1. ID's of the owner of this Application Partner (Trading Partner's unique identifier)
2. Do not use the name "UNKNOWN, or "ALL", or any variation of the case of these two words. For example, do not use "Unknown", or "All".
3. Typically, an EAN number is used in TRADACOMS to identify partners. However, if an EAN is not available and the partners agree, this field can be a name.
4. Alias Names are unique across all TRADACOMS Application Partners, if present.

TRADACOMS Application Partners Delete (Tag: TRCApPartnersDelete)

Use this tag to delete a TRADACOMS Application Partner. Supply the Application Partner's Name in **ApplicationName**, and/or the Application Partner's **AppPartnerID**. You must also supply the associated Trading Partner's name in **PartnerName** and/or the Trading Partner's **PartnerID**. This tag will also delete any Contacts associated with the Application Partner. It will not delete if Application Partner is assigned to a trade link (see [TRCTradingPartnersDelete](#) if this functionality is needed). A warning report (or error if "Abort Run" is selected) is issued if Trade Link does not exist. The tag and field descriptions are:

Field Label	FieldName/Tag	Req'd	Max. Len.	Validation	Notes
N/A	PartnerName	No	35		
N/A	PartnerID	No	35		
Application Partner Name	ApplicationName	No	35		
Application Partner ID	AppPartnerID	No	35		

TRADACOMS inbound trade links

Trade links can be added, changed, or deleted. In addition, existing Trade Links can have their individual Documents added, changed, or deleted.

TRADACOMS trade links are defined by two record types: Link and Doc. The Link record defines the trading/application partners and the Doc record defines the documents exchanged. A trade link requires one Link record, followed by 1 or more Doc record(s). There is no support for Unknown TRADACOMS links. Unique to TRADACOMS is how Application Partners are optional. The [TRCIBLink](#) fields **From_TP_or_AP** and **To_TP_Or_AP** are employed so the XML Autoload Utility knows what type of Partner is in use. To add a new TRADACOMS Inbound trade link, use the tags:

- [TRCIBLink](#)
- [TRCIBTLDocs](#)

To change an existing TRADACOMS Inbound Trade Link, use the tags:

- [TRCIBLink](#)
- [TRCIBLinkChange](#)

To delete an entire TRADACOMS Inbound Trade Link, use [TRCIBLinkDelete](#).

The following table, field, and tag descriptions are for TRADACOMS inbound trade links:

- ["TRADACOMS Inbound Trade Link \(Tag: TRCIBLink\)"](#)
- ["TRADACOMS Inbound Trade Link Document \(Tag: TRCIBTLDocs\)"](#)
- ["TRADACOMS Inbound Trade Link Change \(Tag: TRCIBLinkChange\)"](#)
- ["TRADACOMS Inbound Trade Link Document Add, Change, Delete \(Tag: TRCIBTLDocsAddChangeDelete\)"](#)
- ["TRADACOMS Inbound Trade Link Delete \(Tag: TRCIBLinkDelete\)"](#)

- [**TRADACOMS Inbound Trade Link \(Tag: TRCIBLink\)**](#)
- [**TRADACOMS Inbound Trade Link Document \(Tag: TRCIBTLDocs\)**](#)
- [**TRADACOMS Inbound Trade Link Change \(Tag: TRCIBLinkChange\)**](#)
- [**TRADACOMS Inbound Trade Link Document Add, Change, Delete \(Tag: TRCIBTLDocsAddChangeDelete\)**](#)
- [**TRADACOMS Inbound Trade Link Delete \(Tag: TRCIBLinkDelete\)**](#)

TRADACOMS Inbound Trade Link (Tag: TRCIBLink)

Use this tag to add a new TRADACOMS Inbound Trade Link. The tag and field descriptions are:

Field Label	FieldName/Tag	Req'd	Max. Len.	Validation	Notes
N/A	From_TP_Or_AP	Yes	1	T or A	1
N/A	From_Partner_ID	Yes	35		1
N/A	To_TP_Or_AP	Yes	1	T or A	2
N/A	To_Partner_ID	Yes	35		2
Reset	ResetTL_Ind	No	1	T or F	3
STX Control Number	TLControlNoValue	Yes, only if ResetTL_Ind is T	14	must be a number	4

Notes:

1. Set **From_TP_or_AP** to **T** to specify the From Partner as a Trading Partner or set to **A** to specify the From Partner as an Application Partner. When set to **T**, set **From_Partner_ID** to the **PartnerID** found in TRADACOMS Trading Partner screens; otherwise set it to the **AppPartnerID** found in TRADACOMS Application Partner screens.
2. Set **To_TP_or_AP** to **T** to specify the To Partner as a Trading Partner or set to **A** if the To Partner is an Application Partner. When set to **T**, set **To_Partner_ID** to the **PartnerID** found in **TRCTradingPartners**; otherwise set it to the **AppPartnerID** found in **TRCAppPartners**.
3. True/False; reset STX Control Number. Default: F.
4. Only set this value if the **Reset** is set to **T** (True). Setting this field to any value when the **Reset** is **F** (false) will reject this link.

TRADACOMS Inbound Trade Link Document (Tag: TRCIBTLDocs)

The tag and field descriptions are:

Field Label	FieldName/Tag	Req'd	Max. Len.	Validation	Notes
Document Type	DocType	Yes	10		1
From Application Partner SDT/CDT	From_SDT_CDT	No	1	S or C	2
To Application Partner SDT/CDT	To_SDT_CDT	No	1	S or C	3
Get Mail Post Office Nickname	GetMailPO	Yes	3	valid Get PO	
Test (check box)	TestIndicator	No	1	T (True) or F (false)	4
Test Post Office Nickname	TestPutPO	Yes, if Test Indicator =T	3	valid Put Post Office	5
Production (check box)	ProdIndicator	No	1	T (True) or F (false)	6
Production Post Office Nickname	ProdPutPO	Yes, if Prod Indicator = T	3	valid Put Post Office	5
Generate VAT Report	GenerateVATReport	No	1	T (True) or F (false)	7

Notes:

1. Document includes the version appended, for example: INVOIC9. This value must exist as a valid Document Type in the table **TRCDocTypes**, maintainable in Standards Maintenance in Partner Manager.
2. **From_SDT_CDT** is used only if the From Partner is an Application Partner; it is not used when that Partner is a Trading Partner. It indicates that the From Partner appears in either the SDT Segment (set to **S**) or the CDT segment (set to **C**). This field defaults to **S** and must be the opposite value of the next field, **To_SDT_CDT**.
3. **To_SDT_CDT** is used only if the To Partner is an Application Partner; it is not used when that Partner is a Trading Partner. It indicates that the To Partner appears in either the SDT Segment (set to **S**) or the CDT segment (set to **C**). This field defaults to **C** and must be the opposite value of the previous field, **From_SDT_CDT**.
4. Default: **F**. Set to **T** if including a Test Post Office
5. At least one of the Test/Prod Post Offices must be set (both cannot be blank).
6. Default: **T**. Set to **T** if including a Production Post Office
7. **GenerateVATReport** can only be set if the Document Type is CREDIT, INVOIC, or UTLBIL, and will default to **True** if not specified for these 3 document types.

TRADACOMS Inbound Trade Link Change (Tag: TRCIBLinkChange)

Use this tag to change an existing TRADACOMS Inbound Trade Link. This tag is required to identify the Trade Link via the first four fields in the tag. The remaining fields, **ResetTL_Ind** and **TLControlNoValue**, need not be specified if your only changes are to the documents. The tag and field descriptions are:

Field Label	FieldName/Tag	Req'd	Max. Len.	Validation	Notes
N/A	From_TP_Or_AP	Yes	1	T or A	1
N/A	From_Partner_ID	Yes	35		1
N/A	To_TP_Or_AP	Yes	1	T or A	2
N/A	To_Partner_ID	Yes	35		2
Reset	ResetTL_Ind	No	1	T or F	3
STX Control Number	TLControlNoValue	Yes, only if ResetTL_Ind is T	14	must be a number	4

Notes:

1. Set **From_TP_or_AP** to **T** to specify the **From** Partner as a Trading Partner or set to **A** to specify the **From** Partner as an Application Partner. When set to **T**, set **From_Partner_ID** to the **PartnerID** found in TRADACOMS Trading Partner screens; otherwise set it to the **AppPartnerID** found in TRADACOMS Application Partner screens.
2. Set **To_TP_or_AP** to **T** to specify the **To** Partner as a Trading Partner or set to **A** if the **To** Partner is an Application Partner. When set to **T**, set **To_Partner_ID** to the **PartnerID** found in **TRCTradingPartners**; otherwise set it to the **AppPartnerID** found in **TRCAppPartners**.
3. True/False; reset STX Control Number.
4. Only set this value if the **Reset** is set to **T** (True). Setting this field to any value when the **Reset** is **F** (false) will reject this link.

TRADACOMS Inbound Trade Link Document Add, Change, Delete (Tag: TRCIBTLDocsAddChangeDelete)

This tag is used to add, change, or delete a Document in an existing TRADACOM Inbound Trade Link. It must be nested within **TRCIBLinkChange** tag. To add a Document, set the **AddFlag** to **Y**. To delete, set the **DeleteFlag** to **Y**. The tag and field descriptions are:

Field Label	FieldName/Tag	Req'd	Max. Len.	Validation	Notes
Document Type	DocType	Yes	10		1
From Application Partner SDT/CDT	From_SDT_CDT	No	1	S or C	2
To Application Partner SDT/CDT	To_SDT_CDT	No	1	S or C	3
Get Mail Post Office Nickname	GetMailPO	No	3	valid Get PO	
Test (check box)	TestIndicator	No	1	T (True) or F (false) or R (remove)	4
Test Post Office Nickname	TestPutPO	Yes, if TestIndicator=T	3	valid Put Post Office	4
Production (check box)	ProdIndicator	No	1	T (True) or F (false) or R (remove)	4
Production Post Office Nickname	ProdPutPO	Yes, if ProdIndicator = T	3	valid Put Post Office	4
Generate VAT Report	GenerateVATReport	No	1	T (True) or F (false)	5
N/A	AddFlag	No	1	Y or N	
N/A	DeleteFlag	No	1	Y or N	6

Notes:

1. Document includes the version appended, for example: INVOIC9. This value must exist as a valid Document Type in the table **TRCDocTypes**, maintainable in Standards Maintenance in Partner Manager.
2. **From_SDT_CDT** is used only if the From Partner is an Application Partner; it is not used when that Partner is a Trading Partner. It indicates that the From Partner appears in either the SDT Segment (set to **S**) or the CDT segment (set to **C**). This field must be the opposite value of the next field, **To_SDT_CDT**.
3. **To_SDT_CDT** is used only if the To Partner is an Application Partner; it is not used when that Partner is a Trading Partner. It indicates that the To Partner appears in either the SDT Segment (set to **S**) or the CDT segment (set to **C**). This field must be the opposite value of the previous field, **From_SDT_CDT**.
4. After all changes are applied, at least one of the Test/Prod Post Offices must be set (both cannot be blank). If you want to remove an existing Test or Production Post Office, set test Test/Production Indicator field to "R". For example, this is available in the event you want to change an existing group from Test only to Production Only: set the **TestIndicator** to "R", which will remove the Test Post Office, and set the **ProductionIndicator** to **T** and specify the new Production Post Office in **ProdPutPO**. When setting this field to **R**, you must specify the other (test/production) Post Office if one does not already exist on the group.
5. **GenerateVATReport** can only be set if the Document Type is CREDIT, INVOIC, or UTLBL.
6. Set to **Y** if you wish to remove a Document from a Trade Link. Specify only the first field and this flag when deleting a group; all other fields will be ignored.

TRADACOMS Inbound Trade Link Delete (Tag: TRCIBLinkDelete)

Use this tag to delete an existing TRADACOMS Inbound Trade Link. This requires the **From_TP_Or_AP**, **From_Partner_ID**, **To_TP_Or_AP**, and **To_Partner_ID** to identify the Trade Link to delete. A warning report (or error if "Abort Run" is selected) is issued if Trade Link does not exist. The tag and field descriptions are:

Field Label	FieldName/Tag	Req'd	Max. Len.	Validation	Notes
N/A	From_TP_Or_AP	Yes	1	T or A	1
N/A	From_Partner_ID	Yes	35		1
N/A	To_TP_Or_AP	Yes	1	T or A	2
N/A	To_Partner_ID	Yes	35		2

Notes:

1. Set **From_TP_Or_AP** to **T** to specify the **From** Partner as a Trading Partner or set to **A** to specify the **From** Partner as an Application Partner. When set to **T**, set **From_Partner_ID** to the **PartnerID** found in TRADACOMS Trading Partner screens; otherwise set it to the **AppPartnerID** found in TRADACOMS Application Partner screens.
2. Set **To_TP_Or_AP** to **T** to specify the **To** Partner as a Trading Partner or set to "A" if the **To** Partner is an Application Partner. When set to **T**, set **To_Partner_ID** to the **PartnerID** found in **TRCTradingPartners**; otherwise set it to the **AppPartnerID** found in **TRCAppPartners**.

TRADACOMS Outbound Trade Links

The following table, field, and tag descriptions are for TRADACOMS outbound trade links:

- ["TRADACOMS Outbound Trade Link \(Tag: TRCOBLink\)"](#)
- ["TRADACOMS Outbound Trade Link Document: \(Tag: TRCOBTLDocs\)"](#)
- ["TRADACOMS Outbound Trade Link Change \(Tag: TRCOBLinkChange\)"](#)
- ["TRADACOMS Outbound Trade Link Document Add, Change, Delete \(Tag: TRCOBTLDocsAddChangeDelete\)"](#)
- ["TRADACOMS Outbound Trade Link Delete \(Tag: TRCOBLinkDelete\)"](#)
- [TRADACOMS Outbound Trade Link \(Tag: TRCOBLink\)](#)
- [TRADACOMS Outbound Trade Link \(Tag: TRCOBLink\)](#)
- [TRADACOMS Outbound Trade Link Document: \(Tag: TRCOBTLDocs\)](#)
- [TRADACOMS Outbound Trade Link Change \(Tag: TRCOBLinkChange\)](#)
- [TRADACOMS Outbound Trade Link Document Add, Change, Delete \(Tag: TRCOBTLDocsAddChangeDelete\)](#)
- [TRADACOMS Outbound Trade Link Delete \(Tag: TRCOBLinkDelete\)](#)

TRADACOMS Outbound Trade Link (Tag: TRCOBLink)

Use this tag to add a new TRADACOMS Outbound Trade Link. The tag and field descriptions are:

Field Label	FieldName/Tag	Req'd	Max. Len.	Validation	Notes
N/A	From_TP_Or_AP	Yes		1	T or A
N/A	From_Partner_ID	Yes		35	
N/A	To_TP_Or_AP	Yes		1	T or A
N/A	To_Partner_ID	Yes		35	
Scheme	ControlNoScheme	No		1	0,1,2,3,4 Default: 0
Map	ControlNoMap	Yes, if ControlNoScheme is 2 (user defined)		255	
Reset (Control #)	ResetTL_Ind	No		1	T or F
STX Control Number	TLControlNoValue	Yes, only if ResetTL_Ind is T		14	must be a number

Notes:

1. Set **From_TP_Or_AP** to **T** to specify the **From** Partner as a Trading Partner or set to **A** to specify the **From** Partner as an Application Partner. When set to **T**, set **From_Partner_ID** to the **PartnerID** found in the TRADACOMS Trading Partner screens; otherwise set it to the **AppPartnerID** found in the TRADACOMS Application Partner screens.
2. Set **To_TP_Or_AP** to **T** to specify the **To** Partner as a Trading Partner or set to "A" to specify the **To** Partner as an Application Partner. When set to **T**, set **To_Partner_ID** to the **PartnerID** found in **TRCTradingPartners**; otherwise set it to the **AppPartnerID** found in **TRCAppPartners**.
3. Scheme defaults to "0" (Trade Link). Valid values are: 0=TradeLink; 1=Document; 2=User Defined; 3=none. A User Defined Scheme requires a Map Name in the next field, **ControlNoMap**.
4. True/False; reset STX Control #. Default: **F**
5. Only set this value if the **Reset** is set to **T** (True). Setting this field to any value when the **Reset** is **F** (false) will reject this link.

TRADACOMS Outbound Trade Link Document: (Tag: TRCOBTLDocs)

Use this tag to add a Document to a new TRADACOMS Outbound Trade Link. Tag must be nested within TRCOBLink. The tag and field descriptions are:

Field Label	FieldName/Tag	Req'd	Max. Len.	Validation	Notes
Document Type	DocType	Yes	10		1

Field Label	FieldName/Tag	Req'd	Max. Len.	Validation	Notes
From Application Partner SDT/CDT	From_SDT_CDT	No	1	S or C	2
To Application Partner SDT/CDT	To_SDT_CDT	No	1	S or C	3
RSG Expected	RSGExpected	No	1	N, P, or C	4
RSG Version	RSGVersion	No	1	1-9	4
Get Mail Post Office Nickname	GetMailPO	Yes	3	valid Get PO	
Test (check box)	TestIndicator	No	1	T (True) or F (false) Default: F Set to T if including a Test Post Office	
Test Post Office Nickname	TestPutPO	Yes, if Test Indicator=T	3	valid Put Post Office	5
Production (check box)	ProdIndicator	No	1	T (True) or F (false). Default: T Set to T if including a Production Post Office	
Production Post Office Nickname	ProdPutPO	Yes, if Prod Indicator=T	3	valid Put Post Office	5
Generate VAT Report	GenerateVATReport	No	1	T (True) or F (false)	6
Reset (Control #)	ResetDoc_Ind	No	1	T or F	7
Document Control Number	DocControlNoValue	Yes, if Reset Doc_Ind is T	14	must be a positive whole number	8

Notes:

1. Document includes the version appended, for example: INVOIC9. This value must exist as a valid Document Type in the table **TRCDocTypes**, that are maintained in Standards Maintenance in Partner Manager.
2. **From_SDT_CDT** is used only if the From Partner is an Application Partner; it is not used when that Partner is a Trading Partner. It indicates that the **From** Partner appears in either the SDT Segment (set to **S**) or the CDT segment (set to **C**). This field defaults to **S** and must be the opposite value of the next field, **To_SDT_CDT**.
3. **To_SDT_CDT** is used only if the To Partner is an Application Partner; it is not used when that Partner is a Trading Partner. It indicates that the To Partner appears in either the SDT Segment (set to **S**) or the CDT segment (set to **C**). This field defaults to **C** and must be the opposite value of the previous field, **From_SDT_CDT**.
4. **RSG Expected** defaults to **N**(none) and is one of the following values: **N**=None; **P**=Present; **C**=Create. If set to **C** (Create) the next field, **RSGVersion**, can be set to an integer from 1 to 9. The **RSGVersion** will default to "2" if not specified and **RSG Expected** is **C**.
5. At least one of the Test/Prod Post Offices must be set (both cannot be blank).
6. **GenerateVATReport** can only be set if the Document Type is **CREDIT**, **INVOIC**, or **UTLBIL**, and will default to **True** if not specified for these 3 document types.
7. True/False; reset Document-Level Control Number. Default: F.
8. Only set this value if the **Reset** is set to **T** (True). Setting this field to any value when the **Reset** is **F** (false) will reject this link.

TRADACOMS Outbound Trade Link Change (Tag: TRCOBLinkChange)

Use this tag to change an existing TRADACOMS Outbound Trade Link. This tag is required to identify the Trade Link via the first four fields in the tag. The remaining fields need not be specified if your only changes are to the documents. The tag and field descriptions are:

Field Label	FieldName/Tag	Req'd	Max. Len.	Validation	Notes
N/A	From_TP_Or_AP	Yes	1	T or A	1
N/A	From_Partner_ID	Yes	35		1
N/A	To_TP_Or_AP	Yes	1	T or A	2
N/A	To_Partner_ID	Yes	35		2
Scheme	ControlNoScheme	No	1	0,1,2,3,4	3
Map	ControlNoMap	Yes, if ControlNo Scheme is 2 (user defined)	255		
Reset (Control #)	ResetTL_Ind	No	1	T or F	4
STX Control Number	TLControlNoValue	Yes, only if Reset TL_ Ind is T	14	must be a number	5

Notes:

1. Set **From_TP_Or_AP** to **T** to specify the **From** Partner as a Trading Partner or set to **A** to specify the **From** Partner as an Application Partner. When set to **T**, set **From_Partner_ID** to the **PartnerID** found in the TRADACOMS Trading Partner screens; otherwise set it to the **AppPartnerID** found in the TRADACOMS Application Partner screens.
2. Set **To_TP_Or_AP** to **T** to specify the To Partner as a Trading Partner or set to **A** to specify the To Partner as an Application Partner. When set to **T**, set **To_Partner_ID** to the **PartnerID** found in **TRCTradingPartners**; otherwise set it to the **AppPartnerID** found in **TRCApPartners**.
3. Valid values are: **0**=TradeLink; **1**=Document; **2**=User Defined; **3**=none. A User Defined Scheme requires a Map Name in the next field, **ControlNoMap**.
4. True/False; reset STX Control Number.
5. Only set this value if the **Reset** is set to **T** (True). Setting this field to any value when the **Reset** is **F** (false) will reject this link.

TRADACOMS Outbound Trade Link Document Add, Change, Delete (Tag: TRCOBTLDocsAddChangeDelete)

Use this tag to add, change, or delete a Document on an existing TRADACOMS Outbound Trade Link. Tag must be nested within **TRCOBLinkChange**. To add a Document, set the **AddFlag** to **Y**. To delete, set the **DeleteFlag** to **Y**. The tag and field descriptions are:

Field Label	FieldName/Tag	Req'd	Max. Len.	Validation	Notes
Document Type	DocType	Yes	10		1
From Application Partner SDT/CDT	From_SDT_CDT	No	1	S or C	2
To Application Partner SDT/CDT	To_SDT_CDT	No	1	S or C	3
RSG Expected	RSGExpected	No	1	N, P, or C	4
RSG Version	RSGVersion	No	1	1-9	4
Get Mail Post Office Nickname	GetMailPO	No	3	valid Get PO	
Test (check box)	TestIndicator	No	1	T (True) or F (false) or R (remove)	5
Test Post Office Nickname	TestPutPO	Yes, if TestIndicator=T	3	valid Put Post Office	5
Production (check box)	ProdIndicator	No	1	T (True) or F (false).	5
Production Post Office Nickname	ProdPutPO	Yes, if ProdIndicator=T	3	valid Put Post Office	5
Generate VAT Report	GenerateVATReport	No	1	T (True) or F (false)	6
Reset (Control #)	ResetDoc_Ind	No	1	T or F	7
Document Control Number	DocControlNoValue	Yes, only if ResetDoc_Ind is T	14	must be a positive whole number	8
N/A	AddFlag	No	1	Y/N	
N/A	DeleteFlag	No	1	Y/N	

Notes:

1. Document includes the version appended, for example: INVOIC9. This value must exist as a valid Document Type in the table **TRCDocTypes**, maintainable in Standards Maintenance in Partner Manager.
2. **From_SDT_CDT** is used only if the From Partner is an Application Partner; it is not used when that Partner is a Trading Partner. It indicates that the **From** Partner appears in either the SDT Segment (set to **S**) or the CDT segment (set to **C**). This field defaults to **S** and must be the opposite value of the next field, **To_SDT_CDT**.
3. **To_SDT_CDT** is used only if the To Partner is an Application Partner; it is not used when that Partner is a Trading Partner. It indicates that the To Partner appears in either the SDT Segment (set to **S**) or the CDT segment (set to **C**). This field defaults to **C** and must be the opposite value of the previous field, **From_SDT_CDT**.
4. **RSG Expected** valid values are: N=None; P=Present; C=Create. If set to **C** (Create) the next field, **RSGVersion**, can be set to an integer from 1 to 9. The **RSVersion** will default to **2** if not specified and **RSG Expected** is **C**.
5. After all changes are applied, at least one of the Test/Prod Post Offices must be set (both cannot be blank). If you want to remove an existing Test or Production Post Office, set test Test/Production Indicator field to **R**. For example, this is available in the event you want to change an existing group from Test only to Production Only: set the **TestIndicator** to **R**, which will remove the Test Post Office, and set the **ProductionIndicator** to **T** and specify the new Production Post Office in **ProdPutPO**. When setting this field to **R**, you must specify the other (test/production) Post Office if one does not already exist on the group.
6. **GenerateVATReport** can only be set if the Document Type is **CREDIT**, **INVOIC**, or **UTLBIL**.
7. True/False; reset Document-Level Control #. Default: **F**
8. Only set this value if the **Reset** is set to **T** (True). Setting this field to any value when the **Reset** is **F** (False) will reject this link.

TRADACOMS Outbound Trade Link Delete (Tag: TRCOBLinkDelete)

Use this tag to delete an existing TRADACOMS Outbound Trade Link. This tag requires the fields **From_TP_Or_AP**, **From_Partner_ID**, **To_TP_Or_AP**, and **To_Partner_ID** to identify the link to delete. A warning report (or error if **Abort Run** is selected) is issued if Trade Link does not exist. The tag and field descriptions are:

Field Label	FieldName/Tag	Req'd	Max. Len.	Validation	Notes
N/A	From_TP_Or_AP	Yes	1	T or A	1
N/A	From_Partner_ID	Yes	35		1
N/A	To_TP_Or_AP	Yes	1	T or A	2
N/A	To_Partner_ID	Yes	35		2

Notes:

1. Set **From_TP_Or_AP** to **T** to specify the **From** Partner as a Trading Partner or set to **A** to specify the **From** Partner as an Application Partner. When set to "T", set **From_Partner_ID** to the **PartnerID** found in the TRADACOMS Trading Partner screens; otherwise set it to the **AppPartnerID** found in the TRADACOMS Application Partner screens.
2. Set **To_TP_Or_AP** to **T** to specify the To Partner as a Trading Partner or set to "A" to specify the To Partner as an Application Partner. When set to **T**, set **To_Partner_ID** to the **PartnerID** found in **TRCTradingPartners**; otherwise set it to the **AppPartnerID** found in **TRCAppPartners**.

Setting up the tutorial

This chapter describes the tutorial setup. These requirements must be met before you continue with the tutorial.

- [Software requirements](#)
- [Using Partner Manager](#)

Software requirements

Ensure that the following are installed on your PC:

----- IBM WebSphere Transformation Extender with Launcher

_____ IBM WebSphere Transformation Extender with Launcher Windows

_____ IBM WebSphere Transformation Extender Trading Manager

_____ Microsoft Access

_____ Tutorial exercise files located in :

C:\install_dir\<tmgr_vn.n\tutorial

(where *install_dir* is the Trading Manager installation directory and *n.n* is the Trading Manager version number)

The tutorial contains the following folders under:

C:\install_dir\<tmgr_vn.n\tutorial\exercises:

- **data** contains text (.txt), XML or database files
 - **maps** contains IBM WebSphere Transformation Extender map source (.mms) and map executable (.mmc) files
 - **trees** contains IBM WebSphere Transformation Extender type tree (.mtt) files
 - **myfiles** is the student working folder
- You need to create the directory structure (folders) for **myfiles**. This directory structure is used for the Post Offices in Trading Manager. To do this, in the exercises folder, create the folder **myfiles**, then under **myfiles**, create the following structure:
- /in/app850in
 - /out/app810out
 - /out/externalapp

Using Partner Manager

- [Tutorial exercise 1: defining the Partner Manager database](#)
 - [Tutorial exercise 2: defining trading partners](#)
 - [Tutorial exercise 3: defining post offices](#)
 - [Tutorial exercise 4: defining inbound trade links](#)
 - [Tutorial exercise 5: creating trade link type trees](#)
-

Tutorial exercise 1: defining the Partner Manager database

About this task

Trading Manager uses a relational database to store information about trading partners, routing instructions, access privileges, document tracking and trading relationships. A sample Microsoft Access database is included with the Partner Manager installation.

In this exercise you will create an ODBC DataSource for the Trading Manager sample database and define the database in Partner Manager.

The following file is required for this exercise:

tmgrnn.mdb

This is the sample Trading Manager database. You will find this file in: **C:\install_dir\<tmgr_vn.n\pmgr**

Note: Throughout this tutorial *install_dir*, is used to refer to the current install directory of your IBM WebSphere Transformation Extender software and *n.n* refers to the current version of Trading Manager.

- [Task: define the ODBC data source](#)
 - [Task: define the database in Partner Manager](#)
 - [Task: define server directories in Partner Manager](#)
-

Task: define the ODBC data source

About this task

Before you can define the sample database in Partner Manager, you must first define a System Data Source Name (DSN) as described in the following procedures:

Procedure

1. From the Windows Start menu, select Settings > Control Panel > Administrative Tools > Data Sources ODBC.
The ODBC Data Source Administrator dialog opens.
2. Select the System DSN tab, and then click the **Add** button.
The Create New Data Source dialog opens.
3. Select Microsoft Access Driver (*.mdb) from the list box, and then click **Finish**.
The ODBC Microsoft Access Setup dialog opens.
4. In the **Data Source Name** field, enter the following information: **TradingManagerTutorial**

5. Click Select.
The Select Database dialog opens.
6. Navigate to C:\install_dir\<tmgr_vn.n\pmgr and select the tmgrnn.mdb file.
7. Click OK.
The Select Database dialog closes. The new entry, **TradingManagerTutorial**, is now displayed in the ODBC Data Source Administration dialog in the **System Data Sources** list box.
8. Click OK.
The **ODBC Data Source Administrator** dialog closes and the System Data Source Name is defined.

Task: define the database in Partner Manager

About this task

Partner Manager **Utilities** include an option for defining multiple Trading Manager databases such as Test, Development and Production.

To define a Trading Manager database:

Procedure

1. Start Partner Manager. From the Start menu, select Programs > IBM WebSphere Transformation Extender n.n > Trading Manager > Partner Manager.
2. Click on the **Utilities** icon in the Partner Manager tool bar.
3. On the **Utilities** navigator, click Database Settings. The Partner Manager Database Selection dialog appears.
4. Select Microsoft Access from **Database Vendor** drop-down list as shown below.
5. Select TradingManagerTutorial from the DSN drop-down menu.
6. Click OK.
7. Partner Manager will restart to activate the new database.

Task: define server directories in Partner Manager

Server directories direct the maps to the physical files. In Partner Manager, select **Utilities** and then **Server Directory Configuration**. For this tutorial, we will use the local file system for our directories. Make certain that the following settings are entered:

Access Method: Mapped Drive
Archive: C:\install_dir\<tmgr_vn.n\mmgr\archive
Resend: C:\install_dir\<tmgr_vn.n\mmgr\resend
Share: C:\install_dir\<tmgr_vn.n\mmgr\share

Use the **Test Configuration** button to check the configuration. If it is correct, save this data

Tutorial exercise 2: defining trading partners

About this task

In this exercise you will demonstrate your understanding of the Address Book by defining an internal and external Trading Partner Address Book entry and corresponding Application Partners. You will:

- View the file **edi850inmi.edi** and verify that the information for the Internal and External Trading Partners matches the data tables shown in the tasks.
- Define an Internal Trading Partner and Application Partner.
- Define an External Trading Partner and Application Partner.

The **edi850inmi.edi** file is required for this exercise. This test data file is located in:

C:\install_dir\tmgr_vn.n\tutorial\exercises\data

More information about this file is provided in the following section.

- [Task: analyze the EDI file](#)
- [Task: define an internal trading partner](#)
- [Task: define an external trading partner](#)
- [Task: define an administrative contact for an external trading partner](#)

Task: analyze the EDI file

About this task

Using a text editor such as Windows Notepad, view the file **edi850inmi.edi** and confirm internal partner IDs, qualifiers, ISA and GS addresses, interchange control version ID, element separators (*) and terminators (NL) required for setup.

Task: define an internal trading partner

Procedure

1. Open Partner Manager.
2. From the Address Book navigator, select Internal > X12.
The Address Book for Internal X12 window opens.
3. Click Add.
The **Add X12 Trading/Application Partner** window opens.
4. Refer to the following table to enter information for My Internal Company into the **Add X12 Trading/Application Partner In Internal Folder X12** window.

In Field...	Enter...
Trading Partner tab:	
Trading Partner Name	My Internal Company
ISA Address	12-phone (telephone companies)
Interchange ID	9395100193
ISA Options tab:	
Interchange Control Version ID	00306 – Draft Standards for Trial Use Approved for Publication by ASC X12 Procedures Review board through oc
Interchange Standards ID	U – U.S. EDI Community of ASC X12, TDCC, and UCS
Separators Box	
Element	2A
Sub Element	3E >
Segment Terminator	OD OA CR/LF
Application Partner tab:	
Application Partner Name	Sales
GS Address	3959910033
Alias Name	
User Defined 1	
User Defined 2	
- For values not shown, accept the default values.
5. Click Save. The Address Book will display the entry.
6. Close the **Address Book** window.

Task: define an external trading partner

Procedure

1. From the Address Book navigator, select External > X12.
The Address Book for External X12 window opens.
2. Click the **Add** button.
The **Add X12 Trading/Application Partner** window opens.
3. Refer to the following table to enter information for Johnson Office Supplies into the **Add X12 Trading/Application Partner in External Folder X12** window.
Note: For values not shown, accept the default values.

In Field...	Enter...
Trading Partner tab:	
Trading Partner Name	Johnson Office Supplies
ISA Address	12 – phone (telephone companies)
Interchange ID	8473179000
ISA Options tab:	
Interchange Control Version ID	00306 – Draft Standards for Trial Use Approved for Publication by ASC X12 Procedures Review Board through oc
Interchange Standards ID	U – U.S. EDI Community of ASC X12, TDCC, and UCS
Separators Box	
Element	2A*
Sub Element	3E >
Segment Terminator	OD OA CR/LF
Application Partner tab:	
Application Partner Name	Purchasing
GS Address	006250740
Alias Name	
User Defined 1	
User Defined 2	
4. Click Save.
The information that you have entered is displayed in the Address Book for External X12 window.

Task: define an administrative contact for an external trading partner

About this task

The definition for the external trading partner, Johnson Office Supplies, needs to include some contact information.

Procedure

1. In the Address Book for External X12 window, select Johnson Office Supplies.
2. Click **Contacts**. The Contacts for Johnson Office Supplies window is displayed.
3. Click Add.
The Add a Contact for Johnson Office Supplies window opens.
4. Refer to the following table to add contact information into the Add A Contact for Johnson Office Supplies window.

In Field...	Enter...
Last Name	Jordan
First Name	William
Phone	555-555-1000
5. Click Save.
The contact information is now displayed in the Contacts for Johnson Office Supplies window.
6. Close the Contacts window and the **Address Book for X12 External** window.

Tutorial exercise 3: defining post offices

About this task

In this exercise you will demonstrate your understanding of Post Offices by defining several **Get** and **Put** File Post Offices. You will:

- Define the File Get Post Office where EDI data will be retrieved for processing.
- Define the File Put Post Office where EDI data will be routed for processing by an external application.
- Define the File Put Post Office where Functional Acknowledgments (997) will be routed for transmission.
- [Task: add a file get post office](#)
- [Task: add a file put post office](#)
- [Task: add another put post office](#)

Task: add a file get post office

Procedure

1. Click Post Offices from Partner Manager toolbar.
2. Select **All Post Offices**.
3. Click Add.
The Add Post Office window opens.
4. Enter the information in the following table into the fields in the **Add Post Office** (Get Post Office) window:

In field...	Enter...
Name	EDI data ready for processing
Nickname	EDI
Adapter Type	File
Get/Put indicator	Get Post Office
Directory	install_dir\tmgr_vn.n\mmgr\mail
5. Click Save to save the information.
The **Add Post Office** window closes and the information that you entered is displayed in the All Post Offices window as shown:
6. Close the All Post Offices window.

Task: add a file put post office

Procedure

1. Click Post Offices in the Partner Manager toolbar.
2. Select All Post Offices.
3. Click Add.
The Add Post Office window opens.

4. Enter the information in the following table into the fields in the **Add Post Office** (Put Post Office) window:

In Field...	Enter...
Name	EDI data ready for App Map
Nickname	APP
Adapter Type	File
Get/Put Indicator	Put Post Office
Directory	install_dir\mgr_vn.n\tutorial\exercises\myfiles\in\app850in

5. **Save** your changes.

The Put Post Office will be listed in the **All Post Offices** list box.

Task: add another put post office

Procedure

1. Click Post Offices in the Partner Manager toolbar.

2. Select All Post Offices.

3. Click Add.

The Add Post Office window opens.

4. Enter the information in the following table into the fields in the **Add Post Office** (Put Post Office) window:

In Field...	Enter...
Name	externalapp data ready to send
Nickname	XAP
Adapter Type	File
Get/Put Indicator	Put Post Office
Directory	install_dir\mgr_vn.n\tutorial\exercises\myfiles\out\externalapp

5. **Save** your changes.

The Put Post Office will be listed in the **All Post Offices** list box.

Tutorial exercise 4: defining inbound trade links

About this task

In this exercise you will learn how to define an Inbound Trade Link.

In the following tasks, the External Trading Partner, Johnson Office Supplies, will be sending ANSI 850 Purchase Orders to an Internal Trading Partner called "My Internal Company". Johnson Office Supplies requires that the Internal Trading Partner, My Internal Company, send them a 997 Functional Acknowledgement.

- [Task: define the trade link](#)

Task: define the trade link

Procedure

1. Click Trade Links in the Partner Manager tool bar.

2. Select Inbound.

The X12 Inbound Trade Links window opens.

3. In the X12 Inbound Trade Links window click Add Link. This will open the **X12 Inbound Trade Links Add Wizard**.

4. Select the External Trading Partner (the 'from' partner) by clicking **Find All**. This will open up the X12 Folder Look up In Dialog.

5. Choose the X12 Johnson Office Supplies Purchasing line and click Select.

6. Click Next. Select the Internal Trading Partner (the 'to' partner) by clicking **Find All**. This will open up the **X12 Partner Lookup in Folder: Internal** dialog.

7. Choose the X12 My Internal Company Sales line and click Select.

8. Click Next. Select the Get Post Office for the trade link from the drop down list.

9. The EDI data ready for Processing should be the only available selection. Click Next.

10. Click Finish. The **X12 Trade Link: Add Application** dialog will open. Enter 3060 for the **Version**, and select PO Purchase Order (850) for the **Functional ID**.

11. Click OK. The X12 Inbound Trade Links dialog will open. Use the table below to update:

In Field...	Enter or confirm...
Trade Link:	
From Trading Partner	Johnson Trading Supplies
To Trading Partner	My Internal Company
Get Post Office	EDI data ready for Processing
Post Offices/Routing:	
Test and/or Production	Enable Test Disable Production
Test Put Post Office	EDI data ready for App Map

In Field...	Enter or confirm...
Acknowledgments / Validate By tab:	
Ack Level	Transaction Set Level
997 Version	3060
Validate By	Transaction Set

12. Click Save.

The Automatic Outbound Acknowledgement Creation for Version 3060 dialog will open.

13. Select **Edi data ready for processing** from the **Get Post Office** drop down list, and select **externalapp data ready to send** from the **Test Put Post Office** drop down list. Leave the Production post office blank. Click **Create**.

14. Close the X12 Inbound Trade Links window.

Tutorial exercise 5: creating trade link type trees

About this task

In this exercise you will run the EDI Wizard to generate type trees based on Trade Link information.

These type trees will be used by the Message Manager system to validate data against the ANSI X12 standards.

- [Task: run the EDI wizard](#)
- [Using message manager](#)

Task: run the EDI wizard

Procedure

1. Click EDI Wizard on the from the Partner Manager menu bar. The only menu option available is **EDI Wizard**.
The **EDI Wizard** opens.

2. Refer to the following table to specify the paths for the IBM WebSphere Transformation Extender Design Studio installation and type tree locations:
Remember that *install_dir* refers to the current install directory of your IBM WebSphere Transformation Extender software and *n.n* refers to the current version of Trading Manager.

In Field...	Select...
Directory location of Transformation Extender software	<i>install_dir</i>
Directory location of EDI type trees	<i>install_dir\tmgr_vn.n\tutorial\exercises\trees</i>
Directory for your new message manager trees	<i>install_dir\tmgr_vn.n\mmgr</i>

3. Click Next.
The wizard will warn you that you already have an **x12mail.mtt** tree. This is the default tree the installation provides. You will also receive a warning about the **edfmail.mtt** tree, the default tree for EDIFACT, which is not used in this tutorial.
4. Click OK to update the tree.
The wizard creates the **x12mail.mtt** from your existing X12 trees. Upon successful creation, you will see the message **Your customized type tree(s) have been created successfully**.
5. Click Done, and close **Partner Manager**.

Using message manager

This chapter contains the following tutorials that are related to the use of Message Manager:

- ["Tutorial Exercise 6: Deploying Message Manager"](#)
- ["Tutorial Exercise 7: Configuring System Resources and the Launcher"](#)
- ["Tutorial exercise 6: deploying message manager"](#)
- ["Tutorial exercise 7: configuring system resources and the Launcher"](#)

Tutorial exercise 6: deploying message manager

About this task

In this exercise you will demonstrate your understanding of the deployment process by deploying the Message Manager system of maps.

- Identify the Trading Manager database
- Deploy the **Message Manager** system maps.
- Deploy the **Runmaps** system of maps.

- [Task: identify the trading manager database](#)
 - [Task: deploy the message manager system](#)
 - [Task: deploy the RunMaps system](#)
-

Task: identify the trading manager database

Procedure

1. Open the **update.mdq** file, located in the *install_dir\<tmgr_vn.n\mmgr* directory, in the IBM WebSphere Transformation Extender Database Interface Designer.
 2. In the navigator, right-click on **TmgrDB** and select Edit.
The Database Definition dialog opens.
 3. Expand the Adapter setting.
 4. Select ODBC as the Adapter **Type** and **Microsoft Windows** as the **Platform**.
Note: It is recommended that you select a native adapter type where possible.
 5. Under the **Data Source** options, select TradingManagerTutorial for both **Database Interface Designer** and **Runtime** databases.
 6. Click OK.
 7. Verify the connection by right-clicking on the database in the Database Interface Designer Navigator and selecting Generate Type Tree from Tables.
Note: If a list of tables is displayed, the database connection was successfully established.
 8. Click Close, and then **Save** in the Database Interface Designer.
-

Task: deploy the message manager system

Before you begin

Note: If you have multiple versions of WebSphere Transformation Extender installed, check the "About" box (using Help...About) in the Integration Flow Designer before continuing with this procedure. The major and minor version displayed (for example, 8.2) must match *exactly* the version you are using for this tutorial. If the version does not match, run the Integration Flow Designer manually and select **msg_mgr.msd** in the *<tmgr_vn.n>* directory.

Procedure

1. Start Message Manager by going to the Start Menu and selecting IBM Transformation Extender n.n...>Trading Manager...>Message Manager.
 2. In the Navigator click the Composition tab to expand **msg_mgr**.
 3. Select Message Manager system in the Navigator, then from the menu bar, select System...>Deploy...>Definitions. The **Define Deploy Scripts for Message Manager** dialog opens.
 4. Ensure that **DeployMessageManager** is selected and then select Generate and Transfer Launcher Control File. Make certain that **Build and Transfer Maps** is also selected.
 5. Click Details.
 6. In the Launcher control file path on server dialog box, verify that the **msg_mrg.msl** will be saved in IBM Transformation Extender Design Studio \systems subdirectory.
 7. Run the **DeployMessageManager** deploy script by selecting Message Manager system in the Navigator, then from the menu bar, select System...>Deploy...>Deploy_MRN_Update.
 8. Verify that the system was deployed successfully.
Note: When you deploy the system, the maps will have warnings. This is expected behavior.
-

Task: deploy the RunMaps system

Procedure

1. Select RunMaps system and run the **BuildRunMaps** deploy script.
 2. Verify that the system was deployed successfully.
 3. Close **msg_mgr.msd** file and save the changes.
-

Tutorial exercise 7: configuring system resources and the Launcher

About this task

In this exercise you will demonstrate your understanding of the Resource Registry and Launcher tools by configuring the Resource Registry to work with the Message Manager system and monitoring process activity with the Launcher Monitoring tools.

In this exercise you will:

Procedure

1. Configure the Resource Registry to create the **Resource.mrc** file used by Message Manager to define the Message Manager system to the Resource Registry.
 2. Configure Launcher Administration and Management Console and start Launcher Service.
-

3. Open Launcher Service Monitor.
 4. Drop test file to trigger the system to run.
 5. Verify data validated and routed in Snapshot Viewer.
 6. Record any errors, your solutions and check corrections
- [Task: create resource configuration files](#)
 - [Task: configure the IBM WebSphere Transformation Extender with Launcher](#)
 - [Task: define the IBM WebSphere Transformation Extender with Launcher in the management Console](#)
 - [Task: run the IBM WebSphere Transformation Extender with Launcher](#)
 - [Task: drop the test file](#)
 - [Task: verify that data was validated and routed in Snapshot Viewer](#)
 - [Task: record any errors and your solutions and check corrections](#)
 - [Integrating external systems](#)
-

Task: create resource configuration files

Procedure

1. Start the Resource Registry from the IBM WebSphere Transformation Extender Start menu link. This may be on the **Launcher** submenu, depending upon how the product was installed.
 2. Right-click on **Configuration Files** and select New
 3. Save new Resource Configuration file as **Resource.mrc** under the **C:\install_dir\<tmgr_vn.n\mmgr** directory.
 4. Expand **Resource**.
 5. Right-click on **Launcher** then select Add.
 6. In the **Add New System** dialog, click the browse button and select the **Msg_Mgr.msl** file located in the deploy directory **C:\install_dir\systems**
 7. Click Select then click **OK**. Under Launcher you should have a new entry named **msg_mgr**.
 8. Right-click on **Msg_Mgr** and select Edit. Click Add in the **Edit Resources** dialog.
 9. Browse and select the **mmgr.mrn** file located in the **C:\install_dir\<tmgr_v>n.n\mmgr** directory and click Select, then click OK.
 10. Right-click on Command Server and select Edit.
 11. Add the **mmgr.mrn** file to the Command Server.
 12. Using the same technique you used for changing the Command Server resource, change the **Global** resource to **mmgr.mrn**.
 13. Select File->Exit and click **Yes** to save changes.
-

Task: configure the IBM WebSphere Transformation Extender with Launcher

Procedure

1. The Launcher Administration application defines the Launcher startup parameters, valid deployment directories and access privileges. Access the Launcher Administration by selecting Launcher Administration from the IBM WebSphere Transformation Extender Start menu link. This may be on the Launcher submenu, depending upon how the product was installed.
 2. On the General tab, disable **Automatic Startup** to allow the system to be started manually in the Management Console. Make sure that **Separate Launcher Process** is unchecked. Checking this box will cause the tutorial to fail.
 3. Identify the Resource Configuration file for the Launcher by selecting Resource.mrc under **C:\install_dir\tmgr_vn.n\mmgr** as the **Resource Configuration File**.
 4. On the Deployment Directories tab, ensure that **C:\install_dir\systems** (where you deployed the Message Manager system) is specified as a deployment directory.
 5. Create an account that will allow you to monitor and control the Launcher in the **Access** tab by clicking **Add** then entering a **User Name** (your name), **Login Name** (your initials), and **Password** (*letmein*)
 6. Change the **Access Rights** from **Revoke** to **Grant** for all actions.
 7. Click OK to exit out of the **Launcher Administration** tool.
-

Task: define the IBM WebSphere Transformation Extender with Launcher in the management Console

About this task

The IBM WebSphere Transformation Extender Management Console controls and monitors Launcher systems.

Procedure

1. Start the Management Console by selecting Management Console from the IBM WebSphere Transformation Extender Start menu link or the IBM WebSphere Transformation Extender Design Studio Start menu link. This may be on the **Launcher** submenu, depending upon how the product was installed.
2. Select Launcher >New.
3. Enter a server **Name of MessageManager**.
4. Enter the Launcher **Host Name of localhost**.
5. Enter **User Name** (your initials) and **Password** (*letmein*).
6. Click OK. The **MessageManager** system will now appear in the console window. Leave the Management Console open and continue.

Task: run the IBM WebSphere Transformation Extender with Launcher

Procedure

1. Start the Launcher service in the Windows **Services**, which can be accessed through the Windows Control Panel > Administrative Tools Control Panel.
2. The Launcher Monitor will allow you to monitor the Message Manager system graphically. Access the Launcher Monitor by selecting by selecting Launcher Monitor from the IBM WebSphere Transformation Extender Start menu link or the IBM WebSphere Transformation Extender Design Studio Start menu link.
3. Select Options > Select Launcher and click Add to add a new server.
4. Enter the Launcher Name as **MessageManager**.
5. Enter User Login (**your initials**) and Password (**letmein**).
6. Enter **LocalHost** for **Server**.
7. Click Connect.
8. Select the Message Manager system then click OK and click OK again.
9. In the **Management Console**, right-click the MessageManager Launcher and select Connect.
10. Right-click CompoundSystem and select Start.
11. The Export map will run as soon as the Launcher is started. Check the Management Console to verify it completed successfully before proceeding to the next step. There will be 2 completed maps under History Successes, and the History Total will also be 2.
12. The first time that you run the event server, you may receive a History Function Failure. This is because the export map is looking for a non-existent backup file in the mmgr\share directory. The second and subsequent times the event server is launched, this error will not repeat. To test this, stop the event server and restart it by right clicking on the **CompoundSystem** icon, selecting Stop, then right click again and select Start.
Note: You may see a history failure: **failed at run function** for the argument **checkhipaafiles** in the export component. This failure can be safely ignored if you are not planning to run Health Insurance Portability and Accountability Act (HIPAA) data through the system. If you plan to use HIPAA data, search the Pack for HIPAA EDI documentation for the file **hipaa_x12_type_6_value.dat** for instructions on how to add this support.

Task: drop the test file

Procedure

1. Copy the file **edi850inmi.edi**, located in:
C:\install_dir\tmgr_vn.n\tutorial\exercises\data,
to:
C:\install_dir\tmgr_vn.n\mmgr\mail
The appearance of this file should trigger the system to start.
Note: This file is automatically deleted a few seconds after you copy the file. This is done by Message Manager, indicating that the file was processed.
2. Click the **Snapshot** button in the **Launcher Monitor** as the activity begins to create a snapshot (.mss file).

Task: verify that data was validated and routed in Snapshot Viewer

About this task

The IBM WebSphere Transformation Extender Snapshot Viewer allows you to view the snapshots of Launcher activity taken in the Launcher Monitor. Snapshots are saved in the IBM WebSphere Transformation Extender install directory as **.mss** file.

Procedure

1. Start the Snapshot Viewer, by selecting by selecting Snapshot Viewer from the IBM WebSphere Transformation Extender with Launcher Start menu link or the IBM WebSphere Transformation Extender Design Studio Start menu link
2. The snapshot file name is the date and time the snapshot was taken. Open the snapshot you just created.
3. Right-click on the map time lines to view the detailed processing information.

Task: record any errors and your solutions and check corrections

About this task

A validated 850 PO and 997 Ack should appear in the Put Post Office locations you specified in the Partner Manager exercise.

Procedure

1. If necessary, make any corrections and re-run the system until it completes successfully.
2. When successful, delete *.app file from **Put Directory** location of **C:\install_dir\tmgr_vn.n\tutorial\exercises\myfiles\In\app850in**.

Integrating external systems

This chapter contains the following tutorials that are related to integrating external systems:

- ["Tutorial Exercise 8: Deploying an Inbound Application System"](#)
 - ["Tutorial Exercise 9: Deploying an Outbound Application System"](#)
 - [Tutorial exercise 8: deploying an inbound application system](#)
 - [Tutorial exercise 9: deploying an outbound application system](#)
-

Tutorial exercise 8: deploying an inbound application system

About this task

Your company (My Internal Company) is using Trading Manager and all its components to begin e-commerce activity with an additional trading partner Acme Corp. Follow the exercises to setup and begin this e-commerce activity with this new Partner.

You have already set up My Internal Company as the internal X12 trading partner. All you need to do is set up Acme Corp as an X12 external trading partner.

To your test database, Acme Corp will begin sending purchase orders (850). You will also need to define an AppMap system to handle processing of the Purchase Orders. The AppMap **EDI850IN** is provided.

In this exercise you will:

- Setup Acme Corp as a new trading partner in Partner Manager
 - Deploy Message Manager and an inbound application system (Purchase Order)
 - Deploy outbound application system (Invoice)
 - Modify the system trigger
 - Deploy and test the system
 - [Assumptions](#)
 - [Task: Define a new external trading partner](#)
 - [Task: define a get post office](#)
 - [Task: define a trade link and select acknowledgement level](#)
 - [Task: define an application subsystem](#)
 - [Task: restart the Launcher and run the system](#)
-

Assumptions

About this task

It should be noted that **PO850mi.DAT** is an input data file located in the following directory :

C:\install_dir\tmgr_vn.n\tutorial\exercises\data

The following files are required to complete this exercise.

- **EDI850IN.mms**
An application map source file located in the following directory :
C:\install_dir\tmgr_vn.n\tutorial\exercises\maps
 - **App810ot.mms**
A source map used to convert data for an outbound invoice is located in the following directory :
C:\install_dir\tmgr_vn.n\tutorial\exercises\maps
-

Task: Define a new external trading partner

Procedure

1. Using a text editor such as Microsoft Notepad open the data file **PO850mi.DAT** to confirm partner IDs, element separators(*) and terminators (NL) required for the setup.
2. Review the information in the following Trading Partner Information table about the External Trading Partner, Acme Corp.
3. Referring to the following table, define an External Trading Partner Address Book entry.

In Field...	Enter...
Trading Partner Tab:	
Trading Partner Name:	ACME Corp.
ISA Address:	ZZ - Mutually Defined
Interchange ID:	ACMECORP
ISA Options Tab:	
Interchange Control Version ID:	00306 - Draft Standards for Trial Use Approved for Publication by ASC X12 Procedures Review Board through oc
Interchange Standards ID:	U - U.S. EDI Community of ASC X12, TDCC, and UCS
Separators Box	
Element	2A*

In Field...	Enter...
Sub Element	3E >
Segment Terminator	OD OA CR/LF
Application Partner tab:	
Application Partner Name	PURCHASING
GS Address	ACMECORP

Task: define a get post office

About this task

Referring to the following information, define a new GET Post Office.

Add Post Office	(Get Post Office)
Name:	PO data ready for Processing
Nickname	DAT
Adapter Type	File
Get /Put Indicator	<tmgr directory>\mmgr\mail

Task: define a trade link and select acknowledgement level

About this task

Referring to the information below, define an **Inbound** Trade Link to "**My Internal Company**" from "**ACME Corp**".

Add Trade Link	(Inbound)
From Trading/Application Partner:	ACME Corp.
To Trading/Application Partner:	My Internal Company
Get Post Office:	PO data ready for Processing
Version:	3060
Functional ID:	PO Purchase Order (850)
Post Offices/Routing tab:	
Test/Production Indicators:	Enable Test; Disable Production
Test Post Office	EDI data ready for App Map
Acknowledgements/Validate By Tab:	
Ack Level:	Transaction Set
Validate By:	Transaction Set
997 Version:	3060
	Save the link, then in Automatic Outbound Assistant
Put Post Office (FA)	externalapp data ready to send
Get Post Office (FA)	PO data ready for processing

Task: define an application subsystem

About this task

In this task you will define an application subsystem to take the inbound purchase order and transform it so it can be processed by your internal purchasing application.

Use the **EDI850IN.mms** map to convert the data in the subsystem. You want this map to execute based on the arrival of data in the Put Post Office directory that you defined for Acme Corp. with an extension of ***.app**

Procedure

1. Start the Integration Flow Designer and open a new system definition file.
2. Save the new system definition file in **C:\install_dir\tmgr_vn.n\tutorial\exercises\myfiles** as **AppMap**.
3. Rename **System1** to **Inbound**.
4. Add the map in the **EDI850IN.mms** source map to the Inbound system.
5. Right click on the **EDI850IN** map in the Inbound system and select **Edit Launcher Settings** to set the input trigger and override the source and destination file paths as follows:
 - Input card
Source Event is On
Source file: **C:\install_dir\tmgr_vn.n\tutorial\exercises\myfiles\in\app850in*.app**
 - Output card #1
Target file: **C:\install_dir\tutorial\exercises\myfiles\in\app850in*.ord**

- Output card #2
Target file: **C:\install_dir\tmgr_vn.n\tutorial\exercises\myfiles\in\app850in*.lin**
6. Change the from the Server drop-down list to Local Server.
 7. Create a deploy script named as Inbound with **C:\install_dir\systems** directory as the deploy directory.
Note: Select option Build and Transfer Maps as well.
 8. Deploy the Inbound system.
 9. Save the changes to the system definition file.
 10. Verify that all ***.app** files were deleted from the directory:

Results

C:\install_dir\tmgr_vn.n\tutorial\exercises\MyFiles\In\app850in

Task: restart the Launcher and run the system

Procedure

1. In the Management Console, stop the CompoundSystem.
2. Right-click on MessageManager and select Disconnect.
3. Stop then start the Launcher service.
4. Start the Launcher Monitor and select the Launcher.
5. Open the Management Console and connect to the Launcher (configured in a previous exercise).
6. Start the CompoundSystem.
7. Using Windows Explorer, copy the data file **PO850mi.DAT** in **C:\install_dir\tmgr_vn.n\tutorial\exercises\Data** to the **C:\install_dir\tmgr\mmgr\mail** directory.
This file is the trigger for the **Message Manager** system
8. Take a snapshot of the system in the Launcher Monitor and verify the maps completed successfully.
A validated 850 PO and 997 Ack should appear in the Put Post Office locations:
C:\install_dir\tmgr_vn.n\tutorial\exercises\MyFiles\In\app850in*.app and **C:\install_dir\tmgr_vn.n\tutorial\exercises\MyFiles\Out\externalapp*.xap**
Note: You will also see **lin** and **ord** files created in the **app850in** directory. These were created by the application map.
9. Upon successful completion delete all files under:
C:\install_dir\tmgr_vn.n\tutorial\exercises\MyFiles\In\App850in\.
10. In the Management Console, stop the **CompoundSystem** and disconnect from **MessageManager**.
11. Stop the Launcher service.

Tutorial exercise 9: deploying an outbound application system

About this task

Now that the inbound application system is completed and tested, you must deploy an outbound application system that takes the inbound Purchase Order (850) created in the Message Manager exercise and formats an outbound Invoice (810). The system will be executed based on triggers.

Use the **App810ot.mms** map for your outbound system.

In this exercise you will:

- Use Partner Manager to:
 - Define a new application for My Internal Company and Acme Corp
 - Define a **PUT Post Office**
 - Define an outbound **TradeLink**
- Use the Integration Flow Designer to:
 - Define an outbound application system
 - Define wild card input triggers and wild card output file names
 - Deploy the outbound system
- Use the IBM WebSphere Transformation Extender to:
 - Run the system with the Launcher
 - Monitor the system using the Management Console and Launcher Monitor
- **Files for this exercise:**
- [**Task: define the application partner**](#)
- [**Task: define a post office for outgoing invoice**](#)
- [**Task: define a new trade link for the outgoing 810**](#)
- [**Task: run EDI type tree wizard**](#)
- [**Task: deploy message manager system and RunMaps system**](#)
- [**Task: define new application map in the AppMap system**](#)
- [**Task: start the Launcher and monitor the system**](#)
- [**Partner Manager monitoring**](#)
- [**Trading Manager maintenance**](#)

Files for this exercise:

About this task

- **PO850mi.dat**
Data file located in **C:\install_dir\tmgr_vn.n\tutorial\exercises\data**
 - **App810ot.mms**
Map source file located in **C:\install_dir\tmgr_vn.n\tutorial\exercises\maps**
 - **EDI type trees** (trimmed for class)
Type trees required by EDI Wizard located in the **C:\install_dir\tmgr_vn.n\tutorial\exercises\trees**
 - **itemxref.txt**
Input file for **App810ot.mms** located in **C:\install_dir\tmgr_vn.n\tutorial\exercises\data**
 - **profile2.txt**
Input file for **App810ot.mms** located in **C:\install_dir\tmgr_vn.n\tutorial\exercises\data**
-

Task: define the application partner

Procedure

1. For internal Trading Partner My Internal Company enter a new Application Partner as follows:
 2. **Application Partner Name:** Invoice
 3. **GS Address:** 3959910032
 4. For external Trading Partner Acme Corp enter a new application Partner as follows:
 5. **Application Partner Name:** Invoice
 6. **GS Address:** ACMECO
-

Task: define a post office for outgoing invoice

About this task

Referring to the following information, define a new **PUT Post Office**.

- **PO Name:** Invoices ready to send
 - **PO Nickname:** INV
 - **Adapter Type:** File
 - **PO Path** **C:\install_dir\tmgr_vn.n\tutorial\exercises\myfiles\out\app810out**
-

Task: define a new trade link for the outgoing 810

About this task

Referring to the information below, define an X12 Outbound Trade Link from "My Internal Company" to "Acme".

Add Trade Link	(Outbound)
From Trading/Application Partner:	
Trading Partner	My Internal Company
Application Partner	Invoice
To Trading/Application Partner:	
Trading Partner	ACME Corp.
Application Partner	Invoice
Get Post Office	EDI data ready for Processing
Interchange Control# Scheme	Increment
Group Control# Scheme	Relative
Version	3060
Functional ID	IN Invoice Information (810, 819)
PO/Validation/Ack:	
Put Post Office	Invoices ready to send

Task: run EDI type tree wizard

About this task

Refer to Exercise 5: Creating Trade Link Type Trees exercise if you require step by step instructions.

Task: deploy message manager system and RunMaps system

About this task

Refer to Message Manager Exercise 6 - Task: Deploy the Message Manager System if you require step by step instructions.

Task: define new application map in the AppMap system

Procedure

1. Open the AppMap system definition file that was created in the task titled: "[Task: Define an application subsystem](#)".
2. Create a new system named **Outbound**.
3. Add the source map **App810ot.mms** located in **C:\install_dir\tmgr_vn.n\tutorial\exercises\maps** to this system.
4. Click the Launcher Settings button on **App810ot** and set the input trigger and override the destination file paths as follows:
 - Input card #1: Source Event **ON**
Source file: **C:\install_dir\tmgr_vn.n\tutorial\exercises\myfiles\In\App850in*.ord**
 - Input card #2: Source Event **ON**
Source file: **C:\install_dir\tmgr_vn.n\tutorial\exercises\myfiles\In\App850in*.lin**
 - Input card #3:
Source file: **C:\install_dir\tmgr_vn.n\tutorial\exercises\data\ITEMXREF.TXT**
 - Input card #4:
Source file: **C:\install_dir\tmgr_vn.n\tutorial\exercises\data\profile2.txt**
 - Output card#1:
Target file: **C:\install_dir\tmgr_vn.n\mmgr\mail*mo.edi**
 - Output card #2:
Target file: **C:\install_dir\tmgr_vn.n\tutorial\exercises\data\profile2.txt**
5. Change **Server** to **Local Server**.
6. Create a deploy script for the Outbound system that deploys the Launcher file (**.msl**) to the IBM WebSphere Transformation Extender\systems directory.
7. Deploy the Outbound system.
8. Save your changes.

Task: start the Launcher and monitor the system

Procedure

1. Start **Launcher** service.
2. Start the Launcher Monitor and connect to the Launcher.
3. Open the Management Console and **Connect** to the Launcher and start the CompoundSystem.
4. Using Windows Explorer, locate data file **PO850mi.DAT** in **C:\install_dir\tmgr_vn.n\tutorial\exercises\data**. Copy **PO850mi.DAT** to the **install_dir\tmgr_vn.n\mmgr\mail** directory.
The appearance of this file in the directory will trigger the Message Manager system. The files produced by Message Manager will trigger the AppMap **Inbound** system.
5. Record any errors and your solutions and check corrections
6. Take a snapshot of the system in the Launcher Monitor and verify the maps completed successfully.
A validated 850 PO, 810 INV and 997 Ack should appear in the Put Post Office locations you specified in the Partner Manager exercise.
7. After making any corrections to the dropped test file and running **Launcher** to verify success, delete all files under **C:\install_dir\tmgr_vn.n\tutorial\exercises\myfiles\In\App850in**.
8. In the Management Console, stop the **CompoundSystem** and disconnect from the Launcher.
9. Stop the Launcher service.

Partner Manager monitoring

This chapter contains the following tutorials that are related to the monitoring of Partner Manager :

- "[Tutorial Exercise 10: Running Traffic Reports](#)"
- "[Tutorial Exercise 11: Running Additional Reports](#)"
- [**Tutorial exercise 10: running traffic reports**](#)
- [**Tutorial exercise 11: running additional reports**](#)

Tutorial exercise 10: running traffic reports

About this task

In this exercise you will demonstrate your understanding of monitoring traffic activity by running and exporting a transmission level report for previously processed e-commerce data.

In this exercise you will:

- Run a report for previous exercise activity.
 - Select Display Detail option to view transmission detail.
 - Select Report Info option to view report detail
 - Select drilldown option to query the details of the transmission.
 - Make note of any errors
 - Export a traffic report
- [Task: run inbound transmission traffic report](#)**
- [Task: display transmission detail](#)**
- [Task: export transaction set detail report](#)**
-

Task: run inbound transmission traffic report

Procedure

1. Click on **Reports** in the **Partner Manager** toolbar.
 2. Create a Traffic report that will report Inbound Transmissions for All Dates
The **Inbound Transmission Report View** panel should be displayed.
 3. Right-click on a transmission entry and select Display Detail to review transmission detail information.
 4. Click Close.
 5. Right-click on a transmission entry and select Report Info. Review **Report Information**.
-

Task: display transmission detail

Procedure

1. Right-click on a transmission entry and select Drilldown. Review displayed transmission information. Note that drilldown can also be seen by double-clicking on a traffic report entry.
 2. Right-click on a transmission entry and select Drilldown. Review displayed interchange information.
 3. Right-click on an interchange entry and select Drilldown. Review displayed Functional Group information.
 4. Right-click on functional group entry and select Drilldown. Review displayed Transaction Set information.
 5. Right-click on a transaction set entry and select Display Detail. Review transaction set detail information.
 6. Click Back.
-

Task: export transaction set detail report

Procedure

1. Right-click on a transaction set entry and select Export in the Traffic Report Export dialog.
 2. In the Traffic Report Export dialog, click **Browse** for export file name.
 3. Save the file to: **C:\install_dir\tnmgr_vn.n\tutorial\exercises\myfiles** (where *n.n* indicates current version number), and accept the default file name. Click Open.
The file name and location appear in the **Export File Name** field.
 4. Click Begin **Export**. Your Internet browser is automatically launched.
 5. Review the exported report.
-

Tutorial exercise 11: running additional reports

About this task

In this exercise you will demonstrate your understanding of reviewing Trading Partner setup information by running, reviewing, printing and Trading Partner information.

In this exercise you will:

- Run an Additional Report
 - Review Additional Report information
 - Print Additional Report
 - Export Additional Report
- [Task: run inbound trade link report](#)**
- [Task: export additional report](#)**

- [Task: Print additional report](#)

Task: run inbound trade link report

Procedure

1. Click on **Reports** in the **Partner Manager** toolbar. The Reports navigator appears.
2. Select Report Type: Additional Reports
3. Select Report: X12 Trade Link Inbound Application
4. Click Run Report. The X12 Trade Link Inbound Application Report panel appears.
5. Accept the report default run parameters, click Run Report.
6. Review report information.

Task: export additional report

Procedure

1. Click the **Export** icon within Report Viewer.
 - Select location: **C:\install_dir\tmgr_vn.n\tutorial\exercises\myfiles**
 - Enter File name: **X12_Inbound_Links**
 - File Type: **HTML**
2. In the Traffic Report Export dialog, click **Browse** for export file name.
3. Save the file to: **C:\install_dir\tmgr_vn.n\tutorial\exercises\myfiles** (where *n.n* indicates current version number), and accept the default file name. Click Open. The file name and location appear in the **Export File Name** field.
4. Click Begin Export. Your Internet browser is automatically launched.
5. Review the exported report.

Task: Print additional report

Procedure

1. In the browser window, select **File > Print**.
2. Review the report.

Trading Manager maintenance

This chapter contains the following tutorials that are related to the maintenance of Trading Manager:

- ["Tutorial Exercise 12: Copying the Partner Manager Database"](#)
- ["Tutorial Exercise 13: Security Setup"](#)
- ["Tutorial exercise 12: copying the Partner Manager database"](#)
- ["Tutorial exercise 13: security setup"](#)

Tutorial exercise 12: copying the Partner Manager database

About this task

In exercise you will demonstrate your understanding of Partner Manager database utilities by using the Database Copy utility to create a copy of the current Partner Manager database.

- [Task: create DSN for empty database](#)
- [Task: copy Partner Manager database](#)

Task: create DSN for empty database

Procedure

1. Using Windows Explorer, browse to the **C:\install_dir\<tmgr_vn.n\pmgr** directory.
2. Copy **m4EC_empty.mdb** to your working directory (**C:\install_dir\<tmgr_vn.n\tutorial\exercises\myfiles**) as **CM_Test.mdb**.
3. Start the **Data Sources (ODBC)** application from the Windows **Administrative** Tools menu.

4. On the System DSN tab, click Add.
 5. Select Microsoft Access Driver (*.mdb) and click Finish.
 6. Enter a Data Source Name of **CM_DB_TEST** and a description of **CM_DB_TEST Database**
 7. Click Select. The **Select Database Dialog** appears.
 8. Browse to **C:\install_dir\<tmgr_vn.\tutorial\exercises\myfiles** and select CM_test.mdb
 9. Click OK to save the changes and click OK again to exit the ODBC Data Sources application
-

Task: copy Partner Manager database

Procedure

1. Start the Partner Manager Database Copy utility from the start menu.
 2. Select the DSN of the database you are currently using (**TradingManagerTutorial**)
 3. Select Source Database Vendor: **Microsoft Access**.
 4. Select the new DSN you created in the previous task as the Destination DSN.
 5. Select Destination Database Vendor: **Microsoft Access**.
 6. Click Begin Copy.
-

Tutorial exercise 13: security setup

About this task

In exercise you will demonstrate your understanding of Partner Security by setting secure access to Partner Manager.

In this exercise you will:

- Define a security group.
 - Create a User ID.
 - Restart Partner Manager and sign-on with a User ID.
 - [Task: create security group](#)
 - [Task: create user ID and assign to security group](#)
 - [Task: restart Partner Manager and log on](#)
-

Task: create security group

Procedure

1. Select Security Groups from the **Utilities** navigator. The **Group Information** panel appears.
 2. Enter Security Groups Name of **EDI Admin** and press **ENTER**.
 3. Click **Yes** to create.
 4. Select Security Profile of **Administrator**.
 5. Disable the **EDI Wizard** option.
 6. Save the Security Group.
-

Task: create user ID and assign to security group

Procedure

1. Select User Maintenance from the **Utilities** navigator.
 2. Click Add.
The User Maintenance: Add dialog opens.
 3. Enter your user information in the **User ID** and **User Name** fields.
 4. In the **Group ID** drop down menu, select EDI Admin.
 5. Enter a **Password**.
 6. Enable **Password Expires Every** and enter **30** days.
 7. Save the new user account.
-

Task: restart Partner Manager and log on

Procedure

1. Restart Partner Manager.

2. Enter the User ID and Password you created in the previous task to logon.
3. Verify that the EDI Wizard option is now grayed out.
4. Delete the User ID created above to complete this tutorial. Note the warning shown when deleting the last user from Partner Manager.

Glossary

Look up terms and definitions that you find in IBM® WebSphere® Transformation Extender products and documentation.

This glossary includes terms and definitions for IBM WebSphere Transformation Extender.

The following cross-references are used in this glossary:

- See refers you from a term to a preferred synonym, or from an acronym or abbreviation to the defined full form.
- See also refers you to a related or contrasting term.

[A](#) [B](#) [C](#) [D](#) [E](#) [F](#) [G](#) [H](#) [I](#) [J](#) [L](#) [M](#) [O](#) [P](#) [R](#) [S](#) [T](#) [U](#) [V](#) [W](#)

A

A2A

See [application to application](#).

activity

An action designed to achieve a particular business process. An activity is performed on a set of targets on a specific schedule.

adapter

An intermediary software component that allows two other software components to communicate with one another.

adapter object

An object used in the TX Programming Interface that represents a resource adapter.

API

See [application programming interface](#).

Applicability Statement 2 (AS2)

An EDI protocol for securely exchanging data over the Internet, by using HTTP as a transport.

application programming interface (API)

An interface that allows an application program that is written in a high-level language to use specific data or functions of the operating system or another program.

application to application (A2A)

A data transformation from the output of one application to the input of another application.

AS2

See [Applicability Statement 2](#).

audit log

A log file containing a record of system events and responses.

B

BAPI

See [Business Application Programming Interface](#).

big endian

A format for storage or transmission of binary data in which the most significant value is placed first. See also [little endian](#).

Boolean

Characteristic of an expression or variable that can only have a value of true or false.

bound component

In the Type Designer, a component for which each occurrence of the data can be identified without considering the context in which that occurrence is placed.

bound type

In the Type Designer, a type whose data object can be identified without considering the context in which that data object is placed.

BPML

See [Business Process Modeling Language](#).

BPML activity

A step in a business process that provides directions for how data should be handled.

Business Application Programming Interface (BAPI)

A programming interface that is used to access SAP databases from with SAP or other development platforms. BAPI is used to achieve integration between the R/3 System and external applications and legacy systems.

Business Process Modeling Language (BPML)

An XML-based language that describes business processes designed by the Business Process Management Initiative (www.bpmi.org).

C

card

In the Map Designer, a data object. There are two types of map cards: input and output.

card object

An object used in the TX Programming Interface that represents an input or output card of a map in program memory.

category

A type class that is used to organize types in a type tree in the Type Designer. Categories organize types that have common properties.

certificate

In computer security, a digital document that binds a public key to the identity of the certificate owner, thereby enabling the certificate owner to be authenticated. A certificate is issued by a certificate authority and is digitally signed by that authority.

choice type

A group type with a subclass equal to choice that is used to define a selection from a set of components. A choice type defines a choice group, which is valid when the data matches one of the components in the choice group.

class
A basic unit of the classification hierarchy used in the Type Designer. There are three classes: item, group, and category.

classification hierarchy
The hierarchy of a type tree in the Type Designer. The deeper the subtype, the more specific the data characteristics are. See also [compositional hierarchy](#).

cloud
See [cloud computing](#).

cloud computing (cloud)
A computing platform where users can have access to applications or computing resources, as services, from anywhere through their connected devices. A simplified user interface and application programming interface (API) makes the infrastructure supporting such services transparent to users.

code list
One or many dynamic pairs of code values that contains sender code and receiver code. Each code pair has one description and up to four additional codes relating to the pair.

code list table
A repository for lists of codes that can further define fields.

compiled map component
An Integration Flow Designer object that references an executable map in compiled file format.

complete type name
The name of a type that represents its hierarchical structure within a type tree, which includes the names of all the types in the path from the root type down.

component rule
An expression about one or more components, which is defined in the Type Designer. A component rule is used for validating data and specifies what must be true for the data that is defined by that component to be valid.

composite
A group of related data elements used in EDI transactions.

compositional hierarchy
A hierarchy in which the composition of the data is reflected in the structure of the group type in the group window. See also [classification hierarchy](#).

contracted component
In the Integration Flow Designer, a component that does not display the sources and targets associated with it. See also [expanded component](#).

correlation
Data that enables a user to record document-specific correlation parameters generated during translation, by the correlation service, or by document tracking functions.

D

data element
A unit of data that cannot be divided. An example is the data element "age of a person" with values consisting of all 3-decimal digit combinations.

data object
A portion of data in a data stream that can be recognized as belonging to a specific type.

data structure
The composition of the data, including repeating sub-structures, nested groupings, sequences, and choices.

decode
To convert data by reversing the effect of some previous encoding.

de-enveloping
The process of removing one or more envelopes from a document or a set of documents.

delimited format
Data that has data objects that are separated by delimiters.

delimiter
1. A character, such as comma or tab, used to group or separate units of text by marking the boundary between them.
2. A flag that is formed by a character or a sequence of characters to group or separate items of data by marking the beginning and end of a unit of data. The delimiter is not a part of the flagged unit of data.

dequeue
To remove items from a queue. See also [enqueue](#).

distinguishable types
Types that do not have common data objects.

document envelope
A structure that is applied to a document to prepare it for exchange between trading partners.

document type definition (DTD)
The rules that specify the structure for a particular class of SGML or XML documents. The DTD defines the structure with elements, attributes, and notations, and it establishes constraints for how each element, attribute, and notation can be used within the particular class of documents.

DTD
See [document type definition](#).

E

EDI
See [electronic data interchange](#).

EDI loop
A group of consecutive EDI segments that repeat together in an EDI document definition. There is no object type in Data Interchange Services that defines an EDI loop on its own. EDI loops are logically defined within an EDI document definition.

EDI message
See [EDI transaction](#).

EDI transaction
In X12 EDI Standards, a group of logically related data that makes up an electronic business document, such as an invoice. The layout of an EDI transaction is described by an EDI Document Definition in Data Interchange Services.

electronic data interchange (EDI)
The exchange of structured electronic data between computer systems according to predefined message standards.

element
A component of a document, such as an EDI, XML, or ROD record. An element can be a simple element or a compound element.

encode
To convert data by the use of a code in such a manner that reconversion to the original form is possible.

enqueue
To put a message or item in a queue. See also [dequeue](#).

envelope
A combination of header, trailer, and control segments that define the start and end of an individual EDI message. Each envelope contains a header segment and a trailer segment, which separate the envelope from other envelopes and provide information about the contents of the envelope.

executable map
A compiled map.

execution settings
Settings that influence how a component behaves at execution time. These settings are compiled into the map file or system file. Many of these settings compiled into the map can be overridden (or partially overridden) using execution commands and options.

expanded component
A component that displays the sources and targets that are associated with it in the Integration Flow Designer. See also [contracted component](#).

explicit format
A format that relies upon syntax to separate data objects. Each data object can be identified by its position or by a delimiter in the data. Delimiters will also appear for missing data objects. See also [implicit format](#).

expression
A statement about data objects. Expressions are a combination of literals, object names, operators, functions, and map names. Component rules are expressions that evaluate to either TRUE or FALSE. Map rules are expressions that evaluate to data to produce the desired output.

external link
In the Integration Flow Designer, solid lines displayed in a system definition diagram that visually represent the data flow between two map components.

F

fixed syntax
A group whose components have a fixed size. Each component is padded to a fixed size or its minimum and maximum content size values are equal.

functional acknowledgment
An electronic acknowledgment returned to the sender to indicate acceptance or rejection of EDI documents.

G

global transaction management (GTX)
The monitoring of transactions that can include operations on two or more different data sources. This feature enables databases or servers to be returned to a pre-transaction state if an error occurs. Either all databases and servers are updated or none are. The advantage of this strategy is that databases and servers remain synchronized and data remains consistent.

GPM
See [Graphical Process Modeler](#).

Graphical Process Modeler (GPM)
A stand-alone graphical interface tool that is used in Sterling B2B Integrator to create and modify business processes. The GPM converts the graphical representation of business processes to well-formed BPMN (source code) and saves the effort of writing code.

group
A complex data object that consists of components.

GTX
See [global transaction management](#).

H

hypervisor
Software or a physical device that enables multiple instances of operating systems to run simultaneously on the same hardware.

I

identifier attribute
An attribute that can be assigned to one component to identify a collection of components, when creating type trees and defining components of a group. An identifier attribute is used during data validation to determine whether a data object exists.

image
See [virtual image](#).

implicit format
A format that defines a group type whose data objects are distinguishable by content, not syntax. Implicit format relies on the properties of the component types. Unlike explicit format, if delimiters separate data objects, they do not appear for missing data objects. See also [explicit format](#).

initiator
1. A syntax object in a data stream that signals the beginning of a data object. For example, if a record begins with an asterisk (*), the asterisk would be the record's initiator.
2. In distributed queueing, a program that requests network connections on another system.

input card
In the Map Designer, a component that contains the complete definition of input for the map, including information such as source identification, retrieval specifics, and the behavior that should occur during processing.

interchange
The exchange of information between trading partners. Also a set of documents grouped together, such as EDI documents enclosed within an EDI envelope.

internal link
In the Integration Flow Designer, a solid line displayed by an expanded map component that visually represents the source and target of the map.

item
A simple data object that does not consist of other objects. An item type is represented by a blue dot next to the type name in the type tree.

J

- Java**
An object-oriented programming language for portable interpretive code that supports interaction among remote objects. Java was developed and specified by Sun Microsystems, Incorporated.
- Java Database Connectivity (JDBC)**
An industry standard for database-independent connectivity between the Java platform and a wide range of databases. The JDBC interface provides a call level interface for SQL-based and XQuery-based database access.
- Java EE**
See [Java Platform, Enterprise Edition](#).
- Java Message Service (JMS)**
An application programming interface that provides Java language functions for handling messages.
- Java Platform, Enterprise Edition (Java EE)**
An environment for developing and deploying enterprise applications, defined by Oracle. The Java EE platform consists of a set of services, application programming interfaces (APIs), and protocols that provide the functionality for developing multitiered, web-based applications. (Oracle)
- Java runtime environment (JRE)**
A subset of a Java developer kit that contains the core executable programs and files that constitute the standard Java platform. The JRE includes the Java virtual machine (JVM), core classes, and supporting files.
- Java Secure Socket Extension (JSSE)**
A Java package that enables secure Internet communications. It implements a Java version of the Secure Sockets Layer (SSL) and Transport Layer Security (TSL) protocols and supports data encryption, server authentication, message integrity, and optionally client authentication.
- Java SE Development Kit**
The name of the software development kit that Sun Microsystems provides for the Java platform.
- Java virtual machine (JVM)**
A software implementation of a processor that runs compiled Java code (applets and applications).
- JDBC**
See [Java Database Connectivity](#).
- JMS**
See [Java Message Service](#).
- JRE**
See [Java runtime environment](#).
- JSSE**
See [Java Secure Socket Extension](#).
- JVM**
See [Java virtual machine](#).

L

- literal**
A symbol or a quantity in a source program that is itself data, rather than a reference to data.
- little endian**
A format for storage or transmission of binary data in which the least significant value is placed first. See also [big endian](#).
- local server**
A predefined server that designates the current computer to run the Integration Flow Designer.

M

- map component**
An Integration Flow Designer object that encapsulates a reference to an executable map, along with its execution settings. There are three types of map components: source, compiled, and pseudo.
- map object**
An object used in the TX Programming Interface that represents an instance of a map in the program memory.
- map rule**
An expression that evaluates to data and produces the required output. A map rule is entered on an output card in the Map Designer and cannot be longer than 32KB.
- member**
In the Type Designer, a single occurrence of a component in a group in a type tree. If a component has a range, each occurrence of that component might be referred to as a member of a series.
- message header**
The part of a message that contains control information such as a unique message ID, the sender and receiver of the message, the message priority, and the type of message.
- message queue**
A named destination to which messages can be sent until they are retrieved by programs that service the queue.
- messaging API**
A programming interface that enables an application to send and receive messages and attached files over a messaging system.
- messaging middleware**
Software that provides an interface between applications, allowing them to send data back and forth to each other asynchronously. Data sent by one program can be stored and then forwarded to the receiving program when it becomes available to process it.
- messaging system**
Software used to deliver electronic messages.
- metadata**
Data that describes the characteristics of data; descriptive data.

O

- optional component**

Within a group type, a component that can be defined to represent a data object that is not required to be present in the data. The component range maximum specifies how many occurrences of the data object might optionally exist.

output card

In the Map Designer, a card that contains the complete definition of an output for the map including information such as target identification, destination specifics and the behavior that should occur during processing.

override

An execution setting that overrides default source and target settings of a map.

P

pad character

A character used to fill empty space. For example, in a database application, a field that is ten characters in length that has the word "file" in it contains four text characters and six pad characters.

partition

To divide a type into subtypes that are mutually exclusive.

partitioned type

A type whose subtypes are distinguishable or mutually exclusive.

pattern

A reusable solution that encapsulates a tested approach to solving a common architecture, design, or deployment task in a particular context.

persistence level

A level that determines the degree of detail written to the database as a business process runs. Decreasing the persistence level increases the business process performance at the cost of full tracking for each step of the business process.

predefined business process

A business process that is ready to use upon installation of Sterling B2B Integrator.

primary document

A document that the services in a business process act on or in relation to. A primary document is usually the document passed to a business process by the initiating adapter.

process control information

Map component settings that can be changed at run time by specifying overrides at the command line, in a command file, or by configuring the Launcher.

process data

Data that is accumulated in an XML document about a business process during the life of the process. Activities in the process add elements to the process data and use components of the process data to complete configured processing tasks.

propagation

The point at which the properties of a type are inherited by its subtypes.

property

A characteristic of an object that describes the object. A property can be changed or modified. Properties can describe an object name, type, value, or behavior, among other things.

pseudolink

In the Integration Flow Designer, dotted lines manually drawn in a system definition diagram that visually represent a data flow relationship between two map components that has not yet been determined precisely.

pseudomap component

An Integration Flow Designer object that is a placeholder for an executable map that has not yet been implemented.

R

range

1. The number of consecutive occurrences of the component in the data stream. The range is composed of two numbers separated by a colon.
2. The categorization of an attribute into different segments.

relative type name

The name of a type relative to another type. Relative type names are used when defining components, syntax items, and comment types.

release character

The character that indicates that a separator or delimiter is to be used as text data instead of as a separator or delimiter. The release character must immediately precede the delimiter.

required component

A component that can be defined within a group type to represent a data object that must be present in the data. The component range minimum specifies how many occurrences of the data object are required.

resource adapter

Map input and output data sources that are used to retrieve and route data. Resource adapters provide access to databases, files, messaging systems, and other data sources and targets. Each adapter includes a set of adapter commands that can be used to customize its operation.

restart attribute

An attribute that specifies that processing of the input data should continue even though a data object of the component is invalid. The restart attribute provides instructions for handling errors encountered in a data stream and can be assigned to a component within a group type.

role-based security

Security that provides access rights to certain files, business processes, web templates, and features, according to the permissions associated with the user account.

rollback

The process of restoring data that was changed by an application program or user.

root type

The type from which all other types stem. The root type represents the data objects of all the types in the tree.

row

The horizontal component of a table, consisting of a sequence of values, one for each column of the table.

run map

An executable map that is called using the RUN function.

S

SDK

See [software development kit](#).

Secure FTP

An FTP protocol that uses Secure Sockets Layer (SSL) protocol.

segment

1. A portion of a profile. The format of each segment is defined by a template.
2. An EDI logical unit of information. EDI segments are made up of data elements and composites. Segments are delimited; their components are separated by a delimiter.

series

The consecutive occurrences of a component. In map rules, the [] characters denote an indexed member of a series.

server definition

A definition for a computer that hosts a command server, to which systems under development in the Integration Flow Designer can be assigned as the intended execution server.

service-oriented architecture (SOA)

A conceptual description of the structure of a software system in terms of its components and the services they provide, without regard for the underlying implementation of these components, services and connections between components.

simple type name

The type name that appears next to the type icon in the type tree.

sized attribute

An attribute that can be assigned to one or more components within a group type, whose value specifies the size, in bytes, of the component immediately following it.

SOA

See [service-oriented architecture](#).

SOAP

A lightweight, XML-based protocol for exchanging information in a decentralized, distributed environment. SOAP can be used to query and return information and invoke services across the Internet. See also [web service](#).

software development kit (SDK)

A set of tools, APIs, and documentation to assist with the development of software in a specific computer language or for a particular operating environment.

source map component

An object that references an executable map within a source map file.

stream object

An object used in the TX Programming Interface that permits overrides to the loaded map input and output specifications.

subflow

A sequence of processing steps, implemented using message flow nodes, that is designed to be embedded in a message flow or in another subflow. A subflow must include at least one Input or Output node. A subflow can be executed by a broker only as part of the message flow in which it is embedded, and therefore it cannot be deployed.

subsystem component

An Integration Flow Designer object that references another system which a user has defined.

subtree

A branch of a type tree that includes a type and all of the subtypes that stem underneath it.

subtype

A type that extends or implements another type; the supertype.

supertype

In a type hierarchy, a type that subtypes inherit attributes from.

syntax object

One or more characters used as separators between portions of data. A syntax object can be a number separator, a delimiter, a terminator, an initiator, or a release character.

system

A collection of referenced executable maps that are organized into a unit.

system definition diagram

The graphical representation of a system viewed within a system window in the Integration Flow Designer. A user can interact with system definition diagrams to design systems.

system window

A window in the Integration Flow Designer in which system definition diagrams are created, maintained, and displayed.

T**terminator**

A syntax object that signifies the end of a data object. For example, a carriage return or line feed at the end of a record might be the record's terminator.

transaction set

The basic business document in EDI data. Transaction sets are enclosed in an envelope that separates one transaction set from another. Groups of transaction sets that are functionally related are enclosed in a functional group envelope.

translation object

A source map that has been compiled to provide instructions for translating from one format to another in a way that can be interpreted by the translator.

transmission type

The largest object in an EDI type tree. A transmission might include many interchanges from or to many trading partners.

transport adapter

An adapter (such as an HTTP Adapter) that is used with an encoding/decoding adapter to support various protocols (for example, SOAP) in a transport-independent way. The transport adapter is used to transport the data either from the source or to the destination.

transporting

A method of conveying data using a specified adapter following either an encode or decode command.

trigger

A mechanism that detects an occurrence and can cause additional processing in response.

tuple

See [row](#).

type

The definition of a data object or set of data objects that is graphically represented in a type tree in the Type Designer.

type tree

In the Type Designer, the graphical representation of the definition and organization of data objects.

U

unbound set

The set of all possible types of data that might be listed last in a group.

Unicode

A character encoding standard that supports the interchange, processing, and display of text that is written in the common languages around the world, plus many classical and historical texts.

Uniform Resource Indicator (URI)

A unique address that is used to identify content on the web, such as a page of text, a video or sound clip, a still or animated image, or a program. The most common form of URI is the web page address, which is a particular form or subset of URI called a Uniform Resource Locator (URL). A URI typically describes how to access the resource, the computer that contains the resource, and the name of the resource (a file name) on the computer.

URI

See [Uniform Resource Indicator](#).

user account

The login directory and other information that gives a user access to the system.

UTF-8

Unicode Transformation Format, 8-bit encoding form, which is designed for ease of use with existing ASCII-based systems. The CCSID value for data in UTF-8 format is 1208.

V

variable component name

A component of a group type that includes the literal at the end of the name because it represents more than one type. The literal ANY acts like a wild card, which represents any type whose name could appear in that place.

virtual image (image)

The operating system and product binary files that are required to create a virtual system pattern.

virtual machine

An abstract specification for a computing device that can be implemented in different ways in software and hardware.

W

watch

A map, including the set of events that initiate it, as defined from the Integration Flow Designer.

web service

A self-contained, self-describing modular application that can be published, discovered, and invoked over a network using standard network protocols. Typically, XML is used to tag the data, SOAP is used to transfer the data, WSDL is used for describing the services available, and UDDI is used for listing what services are available. See also [SOAP](#), [Web Services Description Language](#).

Web Services Description Language (WSDL)

An XML-based specification for describing networked services as a set of endpoints operating on messages containing either document-oriented or procedure-oriented information. See also [web service](#).

workbench

The user interface and integrated development environment (IDE) in Eclipse and Eclipse-based tools such as IBM Rational Application Developer.

WSDL

See [Web Services Description Language](#).